

**Design Considerations**  
**for Knowledge Boundary Resources**  
Onboarding App Developers in Platform Ecosystems

A Design Science Research Study

Gwendolyn Borchsenius & Alexander Fife



Thesis submitted for the degree  
Master of Informatics  
60 credits

Department of Informatics  
The faculty of Mathematics and Natural Sciences  
University of Oslo

Spring 2022



**Design Considerations**  
**for Knowledge Boundary Resources**  
Onboarding App Developers in Platform Ecosystems

A Design Science Research Study

Gwendolyn Borchsenius & Alexander Fife  
2022

© Gwendolyn Borchsenius & Alexander Fife

2022

Design Considerations for Knowledge Boundary Resources

<http://www.duo.uio.no>

Printed: Representeren, University of Oslo



# Abstract

Many of today's most successful software companies have grown to global prominence through a platform strategy. A platform strategy involves attracting a versatile community of third-party complementors to develop applications on their platform. However, when the development of software is moved to third-parties, the platform owner must transfer the necessary knowledge to the complementors, enabling them to develop applications on the platform. This is particularly challenging in software platforms because the necessary knowledge is often highly technical and complementors are heterogeneous, have varied aims and are often geographically dispersed. Therefore, creating an effective and scalable onboarding process is important to maximise the potential for third-party contribution in a platform ecosystem. Prior research conceptualises knowledge boundary resources (KBRs) as the means for platform owners to transfer knowledge to complementors. However, there is limited knowledge on how to design KBRs, particularly in the important context of onboarding.

This thesis extends current research on software platforms and KBRs by addressing the following research question: *How can KBRs be designed to onboard complementors in a software platform ecosystem?* The question is examined through a 1,5 year-long engaged design science research study conducted in collaboration with the platform owner of the software platform DHIS2. The study involved the development of a comprehensive online course which was used by 137 students as part of their onboarding to the DHIS2 platform. Through the design, development and evaluation of the course and other DHIS2 KBRs, we identify five design considerations that can guide platform owners when designing KBRs for onboarding complementors: 1) Designing KBRs for comprehensiveness and specificity 2) Broadcasting tutorials, guides, references and explanations, 3) Performing boundary spanning activities, 4) Provisioning interactive broadcasting KBRs and 5) Providing non-platform specific knowledge. These design considerations contribute to practise aiming to guide software platform owners in the design of KBRs to onboard complementors. We also contribute to research on software platforms by extending current knowledge on KBRs.

**Keywords:** software platform ecosystems, software platforms, boundary resources, knowledge boundary resources, knowledge transfer, onboarding

# Acknowledgements

First and foremost, we would like to express our deepest gratitude to our supervisor, Magnus Li, for helping us with this project from start to finish. Without you, the thesis would not be near its final results. Your encouragement, constructive feedback and our endless discussions helped us stay motivated while enjoying the research process.

Second, we would like to thank all participants in our research project. The discussions with you provided us with valuable insights that were essential for the thesis. We would like to thank members of the DHIS2 core team for participating in our research project and taking time for us despite their busy schedule. In addition, we would like to thank the DHIS2 design lab and its members for engaging conversations and lunches with lots of laughter throughout the last two years.

Finally, we would like to thank our family and friends for their love and support. Gwendolyn möchte gerne ihrer Familie, ihren Freunden und Aidan danken für die Aufmunterungen und die Unterstützung. Alexander vil takke familien sin, som har alltid vært der for han, og gutta.

# Table of Contents

<b>1. Introduction</b>	<b>1</b>
<b>2. Literature Review</b>	<b>5</b>
2.1 Platform theory	5
2.2 Kernel theory	11
2.2.1 Diátaxis	11
2.2.2 Comprehensive and specific KBRs	15
2.3 Summary	17
<b>3. Research Approach</b>	<b>18</b>
3.1 Case description	18
3.1.1 HISP and DHIS2	18
3.1.2 DHIS2 application development	20
3.1.3 DHIS2 Design Lab	21
3.1.4. Development in Platform Ecosystem	22
3.1.5 Using a university course as a laboratory	23
3.2 Research paradigm	24
3.3 Methodology: Engaged Design Science Research	25
3.4 Research process	26
3.5 Data collection	28
3.5.1 Problem identification and motivation	29
3.5.2 Demonstration	30
3.5.3 Evaluation	31
3.5.4 Summary of data collected	33
3.6 Data analysis	34
3.6.1 Preliminary study	35
3.6.2 Evaluation of the artefact	36
3.6.3 Developing design considerations	37
3.7 Ethical considerations	39
<b>4. Existing Artefact and Challenges</b>	<b>40</b>
4.1 DHIS2 boundary resources	40
4.1.1 DHIS2 API	41
4.1.2 Data Queries	42
4.1.3 UI component library	43
4.1.4 Datastore and Datastore Manager	44
4.2 DHIS2 knowledge boundary resources	45
4.2.1 DHIS2 App Course	45
4.2.2 DHIS2 references	47

4.2.3 Storybook	48
4.2.4 Data Query Playground	49
4.3 Challenges identified in preliminary study	50
4.3.1 Limited non-platform specific knowledge	51
4.3.2 Complicated platform	51
4.3.3 Insufficient DHIS2 KBRs	52
4.3.4 Lack of assistance	52
<b>5. Artefact Description</b>	<b>54</b>
5.1 Curriculum design	54
5.2 Non-platform specific KBRs	55
5.2.1 React module	56
5.2.2 React and API Assignment	57
5.3 Platform-specific KBRs	58
5.3.1 DHIS2 Tutorial	59
5.3.2 DHIS2 How-to-guides	62
5.4 Boundary spanning activities	63
5.5 Chapter summary	64
<b>6. Artefact Evaluation</b>	<b>65</b>
6.1 Students behaviour	66
6.2 Curriculum design	69
6.3 Non-platform specific KBRs	70
6.4 Platform specific KBRs	71
6.4.1 DHIS2 Tutorial	72
6.4.2 DHIS2 How-to-guides	74
6.5 Boundary spanning activities	76
6.6 Interactive broadcasted KBRs	77
<b>7. Design considerations</b>	<b>80</b>
7.1 Designing KBRs for comprehensiveness and specificity	82
7.2 Broadcasting tutorials, guides, references and explanations	84
7.3 Performing boundary spanning activities	86
7.4 Provisioning interactive broadcasting KBRs	88
7.5 Providing non-platform specific knowledge	90
<b>8. Contributions and discussion</b>	<b>92</b>
8.1 Contributions to practice	92
8.2 Contributions to research	93
8.2.1 Comprehensiveness and specificity	93
8.2.2 Boundary spanning as mechanism for improving broadcasting KBRs	95
8.2.3 Interactive broadcasting KBRs	96
8.3 Limitations	97

8.4 Future research	99
<b>9. Conclusion</b>	<b>100</b>
<b>10. References</b>	<b>101</b>

# List of Images

3.1	DHIS2 Dashboard	19
3.2	Countries that implemented DHIS2	19
3.3	DHIS2 App Hub	20
3.4	Example survey question	30
3.5	Thematic analysis	37
4.1	Comparison between an API query and a Data Query	42
4.2	Examples of DHIS2 UI components	43
4.3	Datastore Manager	44
4.4	DHIS2 App Course modules	46
4.5	Example of DHIS2 App course	46
4.6	DHIS2 References	47
4.7	A button component in Storybook	48
4.8	Data Query and response from the DHIS2 API	49
5.1	React section on DHIS2 App Course	56
5.2	Example solution of React and API assignment	57
5.3	Tutorial and How-to guides sections	58
5.4	The DHIS2 tutorial on DHIS2 App Course	59
5.5	DHIS2 Tutorial browse component	61
5.6	DHIS2 Tutorial insert component	61
5.7	Datastore how-to-guide	63

## List of Tables

2.1	Summary the Diátaxis documentation quadrants	14
2.2	Summary of concepts	17
3.1	Evaluation criteria	31
3.2	Summary of all data collected throughout the project	33
3.3	Example of theme from preliminary study	35
4.1	Summary of identified challenges	50
5.1	Proposed solutions to challenges	64
7.1	Design considerations	81

## List of Figures

2.1	Complementors usage of KBRs on a platform	9
2.2	The Diataxis framework.	12
3.1	The onboarding of students	23
3.2	Design science research process	26
3.3	A timeline of our project	28
3.4	Deductive and inductive elements of data analysis	35
4.1	A subset of the DHIS2 data model	41
6.1	The relationship between the KBRs and boundary resources	65

# Acronyms

**API** Application Programming Interface

**BR** Boundary Resources

**CSS** Cascading Style Sheets

**DHIS2** District Health Information System

**DQP** Data Query Playground

**DSR** Design Science Research

**HISP** Health Information Systems Programme

**HTML** Hypertext Markup Language

**HTTP** Hypertext Transfer Protocol

**KBR** Knowledge Boundary Resource

**KBRs** Knowledge Boundary Resources

**PDF** Portable Document Format

**SDK** Software Development Kit

**UI** User Interface

**UiO** University of Oslo

**UNIX** Uniplexed Information and Computing System



# 1. Introduction

Platform ecosystems can be described as multi-sided networks where a platform owner “opens” the platform and encourages third-parties to develop complementary products and services on top of the platform (Tiwana, 2013; Ghazawneh & Henfridsson, 2013; Cozzolino, Corbo & Aversa, 2021). Such a platform strategy underpins the idea that a platform’s value can benefit from attracting a versatile community of third-party developers (Constantinides et al., 2018). Since platform ecosystems are subject to network effects, every additional complementor brings value to the existing network of complementors and end-users (Tiwana, 2013). Many of today’s largest platforms such as Microsoft, Oracle, Google and Amazon have benefitted from this approach. This thesis focuses on software platform ecosystems specifically. A software platform is “an extensible software-based system that provides the core functionality shared by “apps” that interoperate with it, and the interfaces through which they interoperate” (Tiwana, 2013, p. 5). Third-party developers use these interfaces to develop applications and extend the platform’s functionality. Since third-party developers complement the existing core functionality with add-on functionality, we refer to them as complementors. The platform interfaces are often referred to as boundary resources in platform literature (Ghazawneh & Henfridsson, 2013). Examples of platform boundary resources include APIs, SDKs and end-user licence agreements.

Because a platform strategy moves the application development from a single firm to multiple external complementors, the platform owners need to provide complementors with the required knowledge about the platform to develop applications (Kauschinger et al., 2021). Specifically, platform owners need to furnish complementors with knowledge about how to “access, combine and extend platform functionality in order to develop add-on products” (Foerderer et al., 2019, p. 120). Existing research has conceptualised knowledge boundary resources (KBRs) as the “objects and activities employed by platform owners to overcome knowledge boundaries and enable effective product development outcomes” (Foerderer et al., 2019, p. 125). Through provisioning KBRs, platform owners can transfer the required knowledge to complementors. If the complementors are not provided with sufficient knowledge on how to use the boundary resources, the successful complementor contribution is likely to be endangered (Foerderer et al., 2019). KBRs are particularly important when new complementors join a platform ecosystem as they have no prior experience or knowledge

about the platform. We refer to the process complementors encounter when they acquire platform-specific knowledge to build a custom application on a platform as onboarding. Once they successfully completed their application by utilising different BRs and KBRs, they have been onboarded to a platform. Because a platform relies on acquiring an ecosystem of complementors who develop complements, the onboarding of new complementors is essential (Engert, 2022), especially in the early stages of the platform lifecycle.

Therefore, creating an effective onboarding process is important for software platforms in order to maximise the potential for third-party contribution in a platform ecosystem. However, onboarding new complementors in a platform ecosystem can be challenging for platform owners for several reasons. First, platform ecosystems commonly include a large network of participating complementors (Foerderer et al., 2019) and the onboarding must scale to many, often geographically dispersed, complementors. It is hence not feasible for the platform owner to interact with every individual complementor through personal relationships (Huber et al., 2017). Second, these complementors are heterogeneous with different interests, actions, competencies and goals (Eaton et al., 2015, Yoo et al., 2010). Finally, the technical knowledge required for developing applications is often specialised and complex, increasing the difficulty of transferring knowledge to complementors. (Foerderer et al., 2019; Foerderer et al., 2014).

While existing research has pointed out the importance of facilitating complementors in their development work (Sarker et al., 2012) by transferring required knowledge (Foerderer et al., 2014.), the question of how platform owners can furnish the complementors with the required application development-related knowledge remains largely unaddressed. Although Foerderer et al. (2019) conceptualised KBRs as a way to reduce knowledge boundaries between the platform owner and complementors, existing research has not yet examined how such KBRs should be designed. In this thesis, we extend existing research on KBRs by examining how KBRs can be designed to onboard complementors and answer the following research question: *How can KBRs be designed to onboard complementors in a software platform ecosystem?*

To answer this research question we conducted an engaged design science research study in collaboration with the platform owners of the health management information system platform DHIS2 (District Health Information Software). Informed by the practitioner's framework "Diátaxis" (Procida, 2017) for structuring technical documentation, we design and develop the comprehensive online course "DHIS2 App Course" that aims to bring a complementor with no experience in web development to being able to build an application on DHIS2. We introduce the artefact to the university course "Development in Platform ecosystem" and evaluate how it and the other DHIS2 KBRs onboarded 137 students to application development on DHIS2. Through the design, development and evaluation of the course and other DHIS2 KBRs, we identify five design considerations; 1) Designing KBRs for comprehensiveness and specificity 2) Broadcasting tutorials, guides, references and explanations, 3) Performing boundary spanning activities, 4) Provisioning interactive broadcasted KBRs and 5) Providing non-platform specific knowledge. These design considerations contribute to practise by guiding software platform owners in the design of KBRs to onboard complementors. We also contribute to academic research by extending current knowledge on KBRs. Concretely, we identify boundary spanning as a mechanism for improving KBRs and develop the concepts of comprehensiveness, specificity and interactive broadcasted KBRs. We also outline a set of avenues for further research.

The rest of this thesis is structured as follows; In the **Literature Review** we describe relevant existing academic literature and theoretical constructs that our thesis builds off. We also introduce the kernel theory Diátaxis which informed the design of our innovative artefact.

Then, in the **Research Approach** chapter, we provide a detailed case description of the enterprise software ecosystem DHIS2 that was the focus of this thesis. We present the selected methodology, including the used methods for data collection and data analysis.

In the chapter **Existing artefact and challenges** we give a detailed description of the existing artefact before our intervention, including the boundary resources and KBRs which are necessary for understanding the artefact. We then present some challenges that previously hindered the successful onboarding of complementors to the DHIS2 platform.

This is followed by the chapter **Artefact description** where we explain the changes made to the artefact. We discuss how these changes address the challenges identified in our preliminary study and how our kernel theory has informed the design of the artefact.

In the **Artefact Evaluation** chapter, we present our findings from evaluating our artefact based on a set of predefined evaluation criterias. The presented findings emerge both from observing the artefact in the field and evaluating if and how well the artefact solved the earlier identified challenges.

Thereafter, in the chapter **Design considerations** we present five design considerations that can guide platform owners in designing KBRs for onboarding complementors in a software platform ecosystem.

We conclude with a discussion of limitations, directions for future research and a summary of the study's key contribution in the **Discussion** chapter

## **2. Literature Review**

In this chapter, we draw upon existing research to position our work in relation to other research and introduce the theoretical foundation we expand upon in this thesis. Following this, we introduce the practitioner's framework Diátaxis which guided the design and development of the “DHIS2 App Course” online course. We also introduce the concepts of comprehensiveness and specificity as two important attributes of KBRs.

### **2.1 Platform theory**

In recent years, there has been a significant expansion of digital platforms. Many of today's largest and most successful companies, such as Apple, Facebook, Amazon and Google, have benefitted from a platform strategy. A digital platform describes a technological system that acts as a foundation upon which other firms can develop complementary products, technologies or services (Yoo et al., 2012). Digital platforms can be seen as multi-sided markets because they enable different platform participants to interact with each other. Further, digital platforms are exposed to network effects that describe how the value of the market increases as the number of actors increases (Tiwana, 2013). For example, the 1 billion iOS users benefit strongly from the 2.22 million apps available on the iOS App Store. A great variety of complementary apps can satisfy a large user base and attract more users, attracting even more complementors to develop apps for the ecosystem. These self-reinforcing processes, once triggered, can help platform owners to grow their platform ecosystem. Therefore, motivating complementors to join the platform ecosystem, should therefore be a high priority for platform owners to benefit from these network effects. There are different types of digital platforms, we focus, however, on innovation platforms, also called software platforms. Software platforms are emerging as a dominant model for software development (Tiwana, 2013). A software platform is “an extensible software-based system that provides the core functionality shared by “apps” that interoperate with it, and the interfaces through which they interoperate” (Tiwana, 2013, p. 5). Thus, the software platform serves as the foundation upon which outside parties (complementors) can build complementary products or services.

In software platforms, value and innovation are derived by co-creating products and services with complementors (Evans & Gawer, 2016, Hein et al., 2019, Pershina et al., 2019). Such a platform strategy underlies the idea that a platform's value can benefit from a versatile community of complementors that develop software on top of the core functionality (Tiwana, 2013, Tiwana et al., 2010).

Software platforms do not stand alone but are instead embedded into their surrounding ecosystem which includes platform participants and their interactions with each other (Constantinides et. al, 2018). Typically, an ecosystem has a relatively stable platform core and a set of complementary subsystems such as third-party applications (Baldwin and Woodard, 2008). The platform core provides the core functionality and the interfaces through which the subsystems operate with the platform (Tiwana, 2013). Interfaces give access to the platform by allowing the apps to interact, operate and communicate with the platform (Tiwana, 2013). Apps are add-on software that extends the platform's functionality (Tiwana, 2013). A broad variety of different apps make a platform ecosystem functionally more desirable to its end-users. By separating these different subsystems, complementors can develop and integrate modules without extensive knowledge of the other subsystems in the ecosystem allowing for versatility and scalability of new modules (Tiwana et al., 2010). In addition to the different components, there are several platform participants that serve their roles in the platform ecosystem. The platform owners are primarily responsible for the platform governance and maintenance of the core functionality. Complementors build complementary software on top of the platform core. Finally, end-users interact with the platform by using the apps. Summed up, platform ecosystems can be seen as a complex socio-technical information system that facilitates interaction between various subsystems and platform participants. It also allows the exchange and distribution of information, functional resources and/or services between the platform owner, complementors and end-users. Compared to traditional companies, there is no single controlling authority and no clear hierarchical structures in platform ecosystems. Consequently, the platform owner only has partial control over the app-development processes. This has implications for the governance of the platform ecosystem. The goal of governance should therefore be to orchestrate complementors rather than controlling them in a hierarchical command-and-control structure found in traditional organisations (Evans & Gawer, 2016; Tiwana et al., 2010).

In order for platform owners to cultivate a software platform ecosystem, their focus must shift from developing applications directly to providing resources that third-party developers use to build applications (Ghazawneh & Henfridsson, 2013; Tiwana, 2013). These platform resources are shared with complementors through interfaces known as boundary resources i.e. “software tools and regulations that serve as the interface for the arm’s-length relationship between the platform owner and the application developer” (Ghazawneh & Henfridsson, 2013, p. 174). Bianco et. al. (2014) describe technical boundary resources as boundary resources that are used by the applications directly or assist in the development of them. In software platforms, technical boundary resources typically consist of a software development kit (SDK) and a multitude of related APIs (Bianco et. al., 2014; Ghazawneh & Henfridsson, 2013). Technical boundary resources create design affordances that enable complementors to develop complements in order to serve their target market (von Hippel & Katz, 2002; Hein, 2019). For instance, when Apple added GPS to the iPhone, complementors could build new applications by embedding the GPS into their market context. This enabled a wide variety of innovative applications like Google Maps, Pokemon Go, Uber and other applications. Boundary resources can also take the form of guidelines and regulations that affect the resulting applications, e.g. licensing agreements, design systems and the terms of service (Ghazawneh & Henfridsson, 2013; Engert et. al. 2022). However, these types of boundary resources which are more related to governance are not the focus of this thesis and when we refer to boundary resources we refer to technical boundary resources.

As the locus of application development is moved towards complementors, platform strategies inherently impose knowledge boundaries between the platform owner and platform complementors (Foerderer et al., 2019). Foerderer et al. (2019) identified that there are three technological characteristics of platforms that influence the knowledge boundary between platform owners and complementors; functional extent, interface design, and evolutionary dynamics. Functional extent refers to the “degree and depth of core functionality that a platform offers for reuse and recombination” (Foerderer et al. 2019, p. 129). If there is more functionality, and it is more complex, more knowledge must be transferred to the complementor and the knowledge boundary is broadened. Interface design refers to the specific implementation details of the boundary resources. For instance, if a platform's interfaces are based on widely used standards, complementors may have prior experience or can rely on other sources of knowledge and the resulting knowledge boundaries are lessened.

Alternatively, interfaces can be designed sub-optimally, which increases the amount of knowledge complementors require thus broadening the knowledge boundary. Finally, evolutionary dynamics refers to how much the platform and its boundary resources change over time. When platform boundary resources change, complementors require up-to-date knowledge and knowledge boundaries emerge. Regardless of which cause, if the knowledge boundaries are left unaddressed, they can pose obstacles to successful platform contribution. Overcoming these knowledge boundaries is therefore essential for the success of a platform strategy (Foerderer et al., 2019). Transferring knowledge about how to access and use the boundary resources becomes crucial to ensure a symbiotic relationship between complementor and platform owner.

Existing research has conceptualised knowledge boundary resources (KBRs) as the “objects and activities employed by platform owners in order to overcome knowledge boundaries and enable effective product development outcomes” (Foerderer et al., 2019 p. 125). Importantly, this conceptualisation contains a distinction between boundary objects and boundary spanners. Boundary objects are “artefacts that are plastic enough to adapt to local needs and constraints of the several parties employing them, yet robust enough to maintain a common identity across sites” (Star & Griesemer, 1989, p. 393). In simpler terms, they are technology-based objects which allow communication and coordination between actors across knowledge boundaries, for example textbooks, websites, training videos and whitepapers. Boundary spanners are human resources employed “to gather information from and transmit information to several external domains” (Tushman, 1977, p. 587). Examples of boundary spanners are account managers, customer success managers and help desk employees. Boundary spanners facilitate boundary spanning activities by e.g. holding workshops, answering support tickets, participating in informal conversations, and teaching about the platform. A simplified relationship between complementors and KBRs is illustrated in the figure below (see Figure 2.1). As a complementor seeks to develop applications on top of the platform, they utilise knowledge boundary resources which transfer knowledge about the platforms’ boundary resources. The KBRs can consist of both boundary spanners and boundary objects.



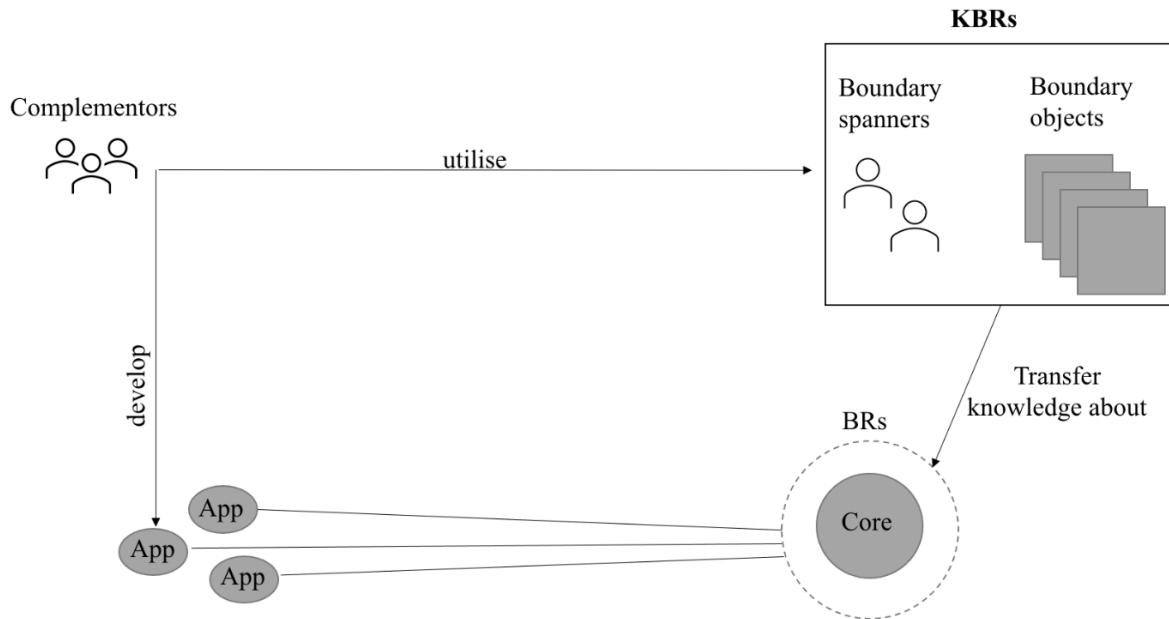


Figure 2.1: Complementors usage of KBRs on a platform

One of the key challenges that platform owners face when provisioning KBRs is that they need to balance the creation of resources that scale, i.e. how many complementors they can address, while simultaneously delivering the necessary scope, i.e. how much of the gap in knowledge they overcome (Foerderer et al., 2019). For instance, an online API reference scales to every complementor in the ecosystem at a low marginal cost. However, its scope is limited because it is not necessarily effective at overcoming knowledge boundaries in all cases. On the other hand, one-on-one assistance can be highly effective at transferring knowledge across boundaries but it scales less and can thus be expensive to provide to many complementors (Huber et al., 2017; Engert et. al, 2022). Foerderer et al. (2019) broadly categorise knowledge boundary resources into three distinct categories; broadcasting, brokering and bridging;

First, broadcasting KBRs are highly standardised boundary objects that complementors can access without interacting directly with platform boundary spanners. Examples of broadcasting KBRs are technical documentation, information portals, video tutorials, sample code, online courses and developer sandboxes. These KBRs scale well, but may have less scope as they may be too standardised to help all complementors in all situations.

Second, brokering KBRs is a semi-formalised type of KBR where boundary spanners refer complementors to other KBRs. Examples of brokering KBRs are help desks and account managers. Brokering KBRs do not scale as well as broadcasting KBRs because they require boundary spanners to mediate the knowledge transfer. They are effective at assisting complementors in finding relevant KBRs, however, they can have a more limited scope if there are no existing KBRs that can be referred to.

Finally, bridging KBRs are the most high-touch type of KBRs where boundary spanners interact with complementors directly to transfer knowledge and problem-solving capabilities. Examples of bridging KBRs are one-on-one assistance, training workshops, and alignment workshops. These types of KBRs scale the least, however, they have a high scope as they can provide highly individualised knowledge and assistance.

To sum up, platform owners face a critical challenge in furnishing complementors with the knowledge required to develop applications on the platform. The underlying technological characteristics of the platform boundary resources influence the resulting knowledge boundaries. When the functional extent of the platform is high, the interface design is unique or the boundary resources change knowledge boundaries to broaden. A platform owner can address these knowledge boundaries by provisioning knowledge boundary resources in the form of boundary objects or boundary spanners. Because platforms can have a high number of complementors, platform owners are faced with the challenge of provisioning KBRs that can scale to the ecosystem while still delivering the necessary scope of knowledge. They do this through strategically provisioning broadcasting, brokering and bridging knowledge boundary resources. These concepts lay the theoretical foundation of our thesis and is the academic literature we expand upon. However, these concepts have not been applied to onboarding complementors to software platforms which we address in the next chapter.

## 2.2 Kernel theory

Following a design science research methodology, we aimed to find a prescriptive design theory that could guide the development of our artefact. We did not find any prescriptive knowledge in the academic literature which could assist with this. However, because our project revolves around creating KBRs that transfer development-related knowledge to complementors, we identified parallels between the development of the course “DHIS2 App Course” and the field of documentation writing. First, we introduce Diátaxis, which is a prescriptive framework for writing technical documentation which heavily influenced the development of the “DHIS2 App Course”. Following this, we introduce two concepts that combine Diátaxis with our theoretical foundation of KBRs.

### 2.2.1 Diátaxis

Diátaxis is a technical documentation framework created by Dianele Procida (2017). It is widely used in industry across software such as Django, NumPy, Ubuntu, Cloudflare, Gatsby and PostgREST (Procida, 2017). Experienced technical writers from among others Google, Redux and WriteTheDocs have highly endorsed its usefulness on an online forum (Basques, 2021; Erikson, 2021; Holscher, 2021). The framework is well-established in the field of technical writing and is encountered by many developers, leading to its use as a kernel theory. The aim of this section is not to reiterate Diátaxis in detail; however, we will cover the essentials for understanding this thesis. The Diátaxis framework is a quick read and practitioners who wish to learn more about it should read the source material on the official website (see [Diataxis.fr](https://diataxis.fr)).

Diátaxis argues that there are four different types of software documentation: tutorials, how-to guides, references and explanations (see Figure 2.2). The framework provides prescriptive design principles for each of these types of documentation. It also states that by structuring documentation according to these distinct categories, technical documentation will become more organised and effective in providing developers with the knowledge required to use the software. By keeping these types separated from each other, technical documentation will improve for both the reader and the writer.

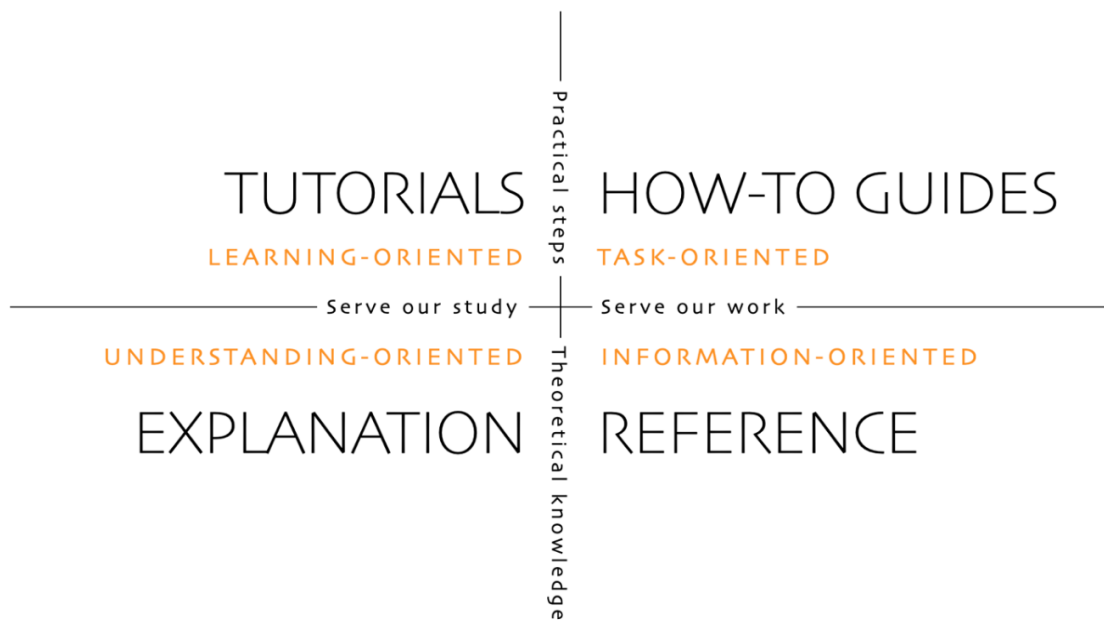


Figure 2.2: The Diátaxis framework

*Note.* From “Diátaxis. A systematic framework for technical documentation authoring.”, by Procida (2017)

Tutorials are the first and arguably the most important type of documentation developers encounter when learning a new system. They aim to introduce the developer to the system and provide an overview of its functionality and other necessary knowledge for using the system. A tutorial commonly leads the reader through a series of steps resulting in a completed project. By following the steps, the developer gains practical experience. After completing the tutorial, the developer is prepared for using the software for their own purposes. Tutorials are learning-oriented and aim to give the developer a foundation of knowledge that prepares them to use the rest of the documentation and the system itself.

How-to guides are a type of documentation that instructs developers on how to solve a specific problem. They guide the developer through a series of steps to achieve a specific goal. Examples of guides are: “How to deactivate your Facebook account” or “How to install Microsoft Windows”. In contrast, “How to develop software” is not a guide because it does not assist with a specific task or problem; it is a skill with no stopping point.

How-to guides also provide an overview of what a developer can achieve with a system. By browsing through a set of how-to guides, the developer can better understand the possibilities of a system. How-to guides are task-oriented and aim to provide step-by-step instructions on how to solve a specific problem.

References are a type of documentation that comprehensively describes the system in a technical, succinct and to-the-point manner. The most common example of references in software systems is the API reference, which describes the interface design of every endpoint and its respective parameters. In contrast to tutorials and how-to guides, references are led by the product it describes, not by the needs of the user. References are supposed to describe the system from a technical point of view without explaining or instructing because this distracts the developer. Rather, it should link to relevant tutorials or how-to guides. References should, however, include examples to illustrate usage. References are information-oriented and aim to provide up-to-date, accurate, comprehensive information about a system.

Explanations are a form of documentation that clarify and illuminate a particular topic. Unlike tutorials or how-to guides, they are not focused on tasks the developer is trying to achieve. Instead, they discuss a topic from higher perspectives and from different angles to increase a developer's understanding of the system. Explanations tie the system together and explain details that are required to understand the system. A minimal explanation is often necessary in tutorials or how-to guides. However because these documentation types should be concise and not share more information than necessary, extensive explanations should reside in a separate section. Explanations are understanding-oriented and aim to give developers a greater understanding of the system and its trade-offs, details, alternatives and other topics of relevance. A summary of the different documentation types and their role can be found in Table 2.1.

	<b>Tutorials</b>	<b>How-to guides</b>	<b>Reference</b>	<b>Explanation</b>
	introduce, educate, lead	guide, demonstrate	states, describes, informs	explains, clarifies, discusses
<i>answers the question</i>	"Can you teach me to...?"	"How do I...?"	"What is...?"	"Why...?"
<i>oriented to</i>	learning	tasks	information	understanding
<i>purpose</i>	to allow the newcomer to get started	to show how to solve a specific problem	to describe the machinery	to explain
<i>form</i>	a lesson	a series of steps	dry description	discursive explanation

Table 2.1: Summary of the Diátaxis documentation quadrants

*Note.* From “Diátaxis. A systematic framework for technical documentation authoring.”, by Procida (2017)

We found Diátaxis promising as a kernel theory for instructing the design of our comprehensive course. The different types of documentation prescribed by Diátaxis captured most of the KBRs the DHIS2 platform provides for application development and provided us with a conceptual lens for describing and evaluating them. Because Diátaxis concerns itself with structuring technical documentation on a website there are parallels to broadcasting KBRs. Additionally, each of the quadrants aims to provide developers with development-related knowledge similar to the processes of knowledge transfer through a KBR. Furthermore, several software platforms like Ubuntu and Django, have stated that they use Diátaxis, validating its applicability to platforms. Diátaxis is promising with regards to the onboarding of complementors as well. The tutorial quadrant is specifically designed around introducing new developers to a software project and the rest of the quadrants are actively used during the development process. This, in combination with wide adoption and endorsements from experienced practitioners, made it a suitable kernel theory for our artefact.

### 2.2.2 Comprehensive and specific KBRs

We theorised how the descriptive concepts put forth by Foerderer et. al. (2019) and the prescriptive design principles from Diátaxis could be combined, with the goal to build a greater understanding of the concept of the scope, i.e. the effectiveness of a KBR at overcoming knowledge boundaries. Because Diátaxis concerns itself with structuring technical documentation on a website, it is not directly applicable to all KBRs. Therefore, we introduce two new concepts; comprehensiveness and specificity. These concepts do not fully encompass the notion of scope, nor do they connect to the entirety of Diátaxis. They are a reconceptualisation of aspects from both Diátaxis and Foerderer (2019) which we found intuitive and prescriptively useful during the project and for theorising afterwards. We posit that the effectiveness of a KBR depends on how comprehensively it covers the platform and its boundary resources. Furthermore, we posit that the effectiveness of a KBR also depends on how specific the knowledge is for a complementor's task and context.

Comprehensive KBRs are oriented towards the platform's boundary resources. Comprehensiveness refers to the degree to which a KBR covers the functional extent and interface design of a boundary resource. For instance, API references have high comprehensiveness because they describe the breadth of the API. A tutorial can be more or less comprehensive depending on how much of the functional extent of the platform to which it introduces the developer. It is important to note that not every KBR should aim to be comprehensive. However, we propose the sum of knowledge provided by all of the KBRs should be comprehensive and cover the entirety of the functional extent and interface design of the platform.

Specific KBRs are oriented towards the tasks complementors perform on the platform. The specificity of a KBRs refers to the degree of how relevant it is to a complementor's task and context. Specific KBRs aim to reduce the amount of extraneous knowledge and give relevant information required for completing a task, for example through step-by-step instruction. For instance, the how-to guide "How to unpack a .zip file on Windows 10" is highly specific to users on Windows 10 who want to unpack .zip files. For a user aiming to unpack .zip files on Windows 7, it will require a certain degree of translation into the Windows 7 context, making it less specific but still relevant. However, for users on UNIX, the guide will not be specific to their context and most likely be unusable. Naturally, for users who want to open a PDF the

guide is unspecific. How-to guides are highly specific KBRs as they aim to assist in completing a specific task. However, tutorials can also be specific. For instance, a good tutorial may aim to cover tasks a complementor is likely to encounter after they have completed the tutorial, making it more specific.

Most KBRs that are used during onboarding have a measure of comprehensiveness towards the platform or specificity towards the tasks complementors are trying to achieve. If we apply these concepts to the Diátaxis framework, references are highly comprehensive, but not very specific. How-to guides are highly specific, but not very comprehensive. Tutorials can vary depending on the platform, and what the tutorial aims to achieve. For instance, a platform with less functional extent could have a single highly comprehensive tutorial that could cover all of the platform's functionality. However, a platform with a high functional extent would likely aim to have multiple tutorials tailored to different use cases each with varying degrees of comprehensiveness. Finally, the comprehensiveness of explanations varies greatly but they do not tend to be oriented towards tasks and are often not specific.

Because Diátaxis was only directly applicable to broadcasting KBRs, we have attempted to extract some of the prescriptive knowledge and generalised it to other KBRs such as boundary spanning activities in an onboarding context. For instance, boundary spanning activities can become highly specific because the boundary spanner can identify precisely what knowledge the complementor requires by communicating with them, and then transfer it to them tailored to their context. However, this depends on how comprehensively the boundary spanner knows the platform. Although, if the boundary spanner knows where to find the knowledge requested, they are positioned to broker comprehensive knowledge. Most types of KBRs related to application development can be assigned these two attributes which we found provided both descriptive and prescriptive potential. These two concepts have guided our conceptualisation of KBRs throughout the project and are used to describe the artefact, for reasoning during the evaluation, and are finally embedded into the design considerations.



## 2.3 Summary

Throughout this chapter, we introduced the academic literature that positions our research and provides the theoretical foundation of this thesis. We introduced our kernel theory, Diátaxis, which provided us with the four concepts; tutorials, references, how-to guides and explanations. These four types of documentation guided the design of the artefact and helped us to describe the broadcasting KBRs we encountered throughout this thesis. Finally, we introduced the concepts of comprehensiveness and specificity, which is a reconceptualisation of Diátaxis and Foerderer et al. (2019) and can be applied to more KBRs used during onboarding. A summary of the concepts we use throughout this thesis brief explanation of them is in the table below:

Concept	Description
Broadcasting KBR	Boundary objects with high scale, typically websites, online courses, documentation and textbooks.
Brokering KBR	Activity where boundary spanners refer a complementor to another KBR that contains the specific knowledge they require.
Bridging KBR	Activity where boundary spanners assist complementors with specific knowledge and problem-solving skills.
Functional extent	The degree and depth of functionality that a platform offers for reuse and recombination. A greater functional extent can increase the knowledge boundaries between complementor and platform owner.
Interface design	Specific implementation details about the functionality. Suboptimal or proprietary interface design can increase knowledge boundaries
Tutorial	Introduction to a system which aims to provide a foundation of knowledge that enables complementors to use the rest of the documentation and platform.
How-to guide	Step-by-step instruction which aims to assist a complementor with knowledge on how to solve a specific task.
References	Comprehensive technical description of the platform boundary resources' functional extent and interface design. Typically used when documenting API endpoints and their respective parameters.
Explanation	Explanations about topics of relevance which aim to aid the complementor with a greater understanding of the platform.
Comprehensiveness	Refers to the degree of which a KBR covers the functional extent and interface design of a boundary resource.
Specificity	Refers to the degree a KBR is relevant to a complementor's task and context.

Table 2.2: Summary of concepts

## **3. Research Approach**

This chapter outlines our research approach. We first provide the important background knowledge about DHIS2 and our empirical study conducted in collaboration with the DHIS2 platform owner. Then, we present the research paradigm that guided our research, followed by a presentation of the chosen methodology and how we deployed it in our project. We explain how our design considerations were developed by presenting the deductive and inductive elements that fed into the development of the design considerations. We present the selected methods for gathering and analysing data. This is followed by a presentation the evaluation means for the artefact. Finally, we reflect upon some ethical considerations.

### **3.1 Case description**

#### **3.1.1 HISP and DHIS2**

This study was conducted as part of the global Health Information Systems Programme (HISP), centred at the University of Oslo (UiO). The program was founded in 1994 and has since promoted health information systems in low- and middle-income countries. HISP's goal is to, “enable and support countries to strengthen their health systems together with regional HISP groups through increased capacity to govern their Health Information Systems in a sustainable way to improve the management and delivery of health services” (UiO, n.d.). To attain their goal, HISP promotes DHIS2 as a global public good. DHIS2 is the world's largest information management system and supports the collection, analysis and management of health data (see Image 3.1). Today, more than 73 countries worldwide have implemented DHIS2 (see Image 3.2). It is fully open-source and was developed through global collaboration, led by the University of Oslo. HISP works in close collaboration with trusted regional partners all around the world to support local implementation of DHIS2 by cooperating with local health authorities, NGOs, donors and consultants.

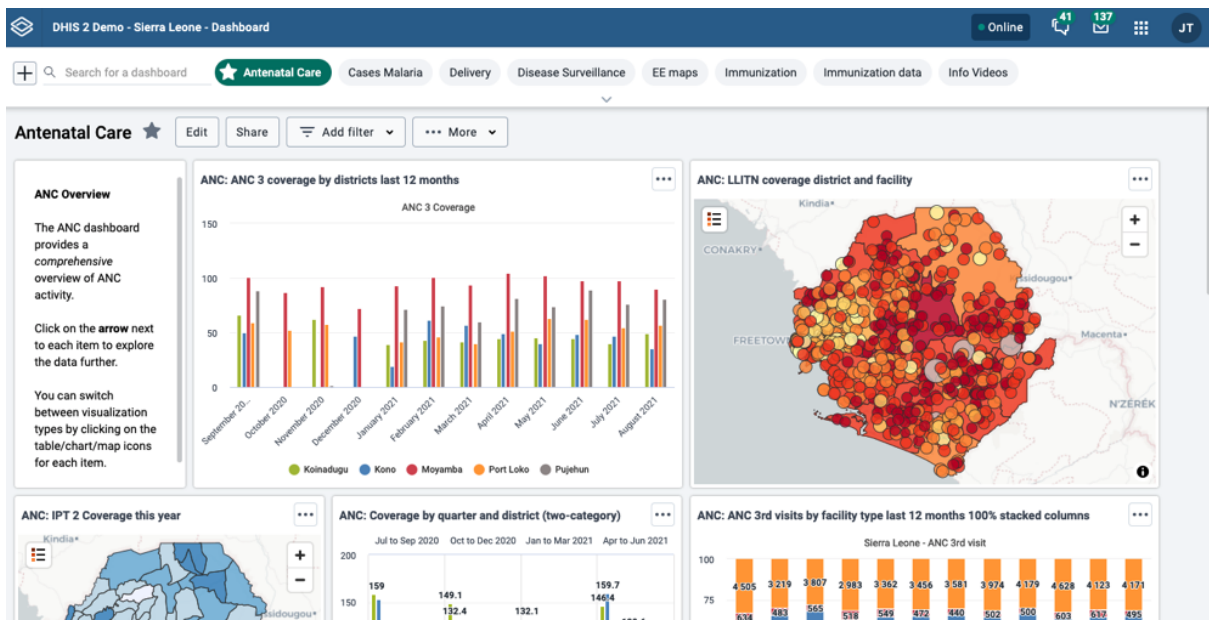


Image 3.1: DHIS2 Dashboard

*Note.* From “Managing dashboards”, by DHIS2 (n.d)



Image 3.2: Countries that implemented DHIS2

*Note.* From “Worldwide Map: DHIS2 in Action”, by DHIS2 (n.d)

### 3.1.2 DHIS2 application development

To support local innovation through application development, the DHIS2 platform owner, also called the DHIS2 core team, has opened their platform for complementors. The DHIS2 platform consists of a generic, stable core that is applicable across countries and relevant for different use cases. The platform core is developed and maintained by the DHIS2 core team, located at the University of Oslo. In addition, the core team provides a set of boundary resources through which DHIS2 complementors can interact with the core. The DHIS2 complementors can then develop customised applications that fit their specific use contexts by extending on the core functionality. This gives the complementors the opportunity and flexibility to adapt application development to local needs. The DHIS2 applications can then be published through the DHIS2 App Hub (see Image 3.3) for worldwide use.

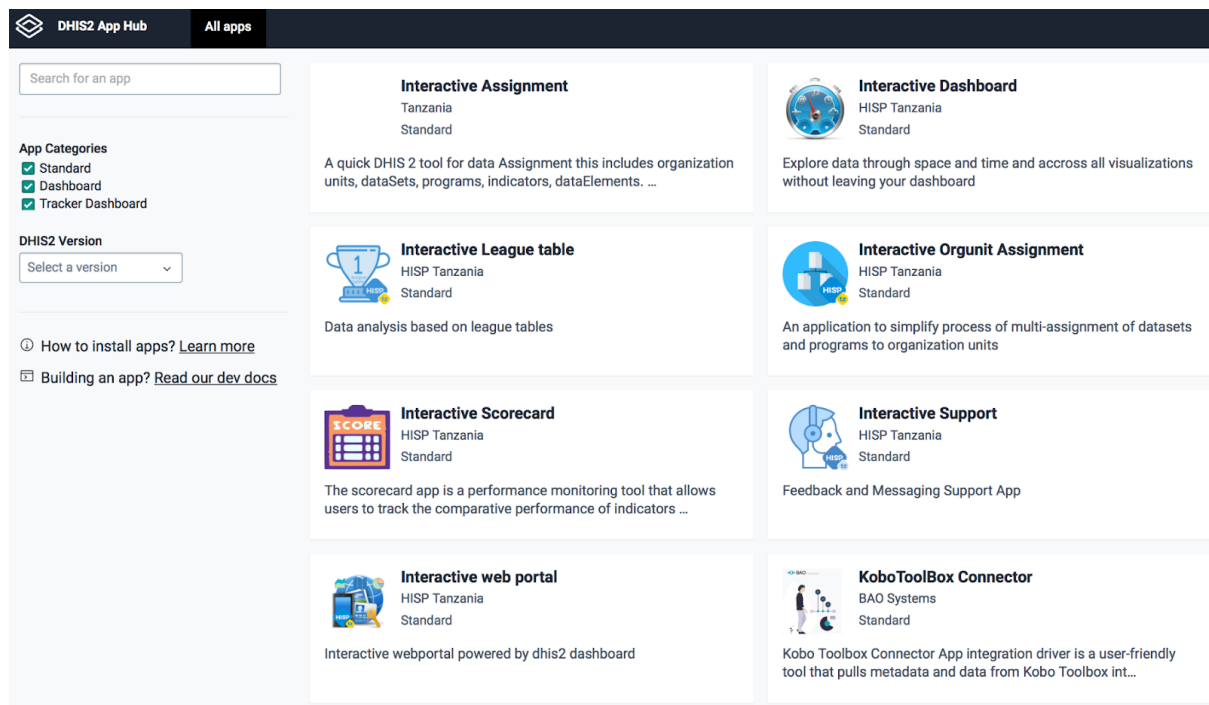


Image 3.3: DHIS2 App Hub  
*Note.* From DHIS2 App Hub (n.d)

### **3.1.3 DHIS2 Design Lab**

Our research project is part of the DHIS2 design lab which aims to promote design and innovation within the DHIS2 platform ecosystem. The design lab involves researchers and post-graduates at the University of Oslo who collaborate with DHIS2 practitioners such as DHIS2 platform owners and HISP groups on various projects (see [DHIS2 Design Lab](#)). The members of the DHIS2 design lab share a common goal of “strengthening usability, communicating using digital tools, and meeting on a frequent basis for discussions around experiences, challenges, ideas, and (possible) interventions” (Li, 2019, p. 6).

The DHIS2 core team collaborates closely with the DHIS2 design lab to further their strategic objectives in promoting design and innovation within the DHIS2 ecosystem. An important research area has been the exploration of “resources that support and promote DHIS2 application development” (UiO, n.d). As such, the DHIS2 core team and design lab have collaborated on how they can build capacity for application development by building online resources, which is the research context of our thesis. Through the design lab, we have been granted access to the university course “Development in Platform Ecosystems”. The course has on multiple occasions been used by the design lab to test DHIS2 platform resources. Through this course, we could research how we can develop capacity building resources with the aim of applying our findings to improve application development in DHIS2 as a whole. We will further elaborate on the course in the next section.

### **3.1.4. Development in Platform Ecosystem**

“Development in Platform Ecosystem” is a master’s level course at the Department of Informatics at the University of Oslo and is held every autumn semester. Essentially, the course aims to bring a student with no experience in front-end development to being able to build an application on DHIS2.

We took the course ourselves in autumn 2020, the first time it introduced the website “DHIS2 App Course” (see [DHIS2 App Course](#)). The website is a self-paced online course that contains all learning resources for the Development in Platform Ecosystems course. However, at the time we took the course, it was incomplete and some sections were unfinished. The continued development of the App Course website is the foundation of our project and is described in Chapter 4 and 5.

When the course period started in the autumn of 2021, we were employed by the University of Oslo as seminar teachers for the course. We held weekly seminars, assisted students at request, graded mandatory assignments and evaluated the final project. This gave us a close relationship with the 137 students who took the course and insight into their usage of KBRs, the DHIS2 platform and their code output. The Development in Platform Ecosystems course is structured in two parts. During the first six weeks of the course, the students individually learn the prerequisite knowledge for developing an application through the DHIS2 App Course website. They have to learn HTML, CSS, Javascript, React and DHIS2 application development. Following this, they are assigned into project groups of three to five students and spend the next five weeks developing a DHIS2 application. Throughout the project, the students utilise DHIS2 KBRs in addition to the DHIS2 App Course website to develop their application. This application is then evaluated by the seminar teachers and the course supervisor, who determine their final grade in the course

### 3.1.5 Using a university course as a laboratory

As mentioned previously, the DHIS2 core team has a long history of using the Development in Platform Ecosystems university course as an arena for testing DHIS2 in order to improve the DHIS2 platform. For instance, during the COVID-19 pandemic, DHIS2 was used globally for contact tracing. During the autumn semester of 2020, the final project was to develop a contact tracing application. After the course, experiences from the applications and the design of them were shared with the DHIS2 core team in order to identify areas of improvement of the core platform.

In a similar manner, we have used the course as a laboratory to examine how DHIS2 as a whole can be improved. To reiterate, we define onboarding as “the process complementors encounter when they acquire platform-specific knowledge to build a custom application on a platform”. In the context of the course, the students are comparable to complementors in the DHIS2 ecosystem. The onboarding process the students go through is to a large degree comparable to onboarding in DHIS2 and other software platforms. The students enter the course with no prior knowledge and complete the course when they have completed a custom application, as shown in Figure 3.1. Additionally, they use the same platform boundary resources and KBRs as DHIS2 complementors do and are highly similar in that regard. This context provides us with a unique research opportunity for exploring how new complementors use a platform's boundary resources and KBRs during their onboarding process.

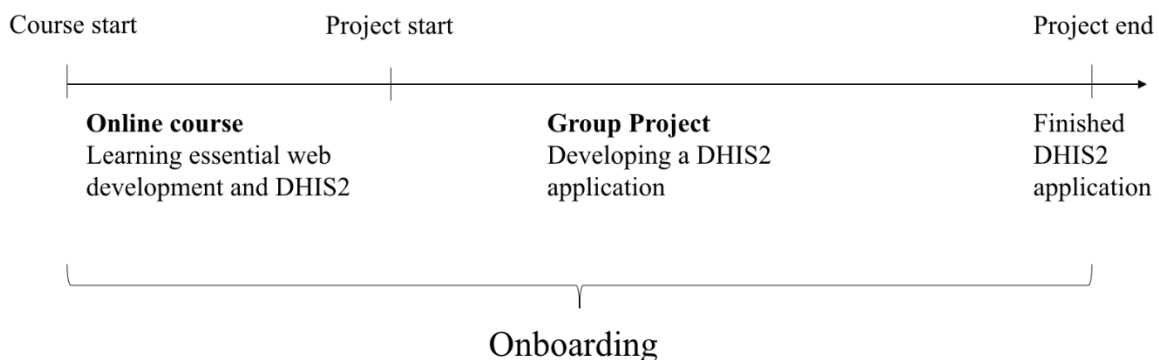


Figure 3.1: The onboarding of students

We have purposefully chosen a research question that we believe is positioned between the DHIS2 core team's research goal and the available empirical context with the aim of finding insights applicable to other software platforms as well. We have avoided generalising from aspects where the research context is too dissimilar to the DHIS2 ecosystem and other software platforms. For instance, we have not focused on KBRs related to the acquisition of complementors or maintenance of applications. However, the case context has some details we would like to address. There are some key differences between complementors in other software platform ecosystems and students in the university course. In contrast to complementors in other ecosystems, the students have no inherent reason, incentive or goal for developing an application outside of passing the course. The course context is reliant on guiding students through assignments and projects. This has implications for KBR design as we can create highly specific resources. Additionally, complementors commonly work in a different organisational structure and the dynamics of a group of students in a project are not comparable. We have addressed this by not emphasising interpersonal aspects and focusing more on the onboarding of individuals themselves. Finally, the context of the university course is more learning-oriented and the students entered into the onboarding process without much prerequisite knowledge. This has influenced this thesis to a degree, however, this more pedagogical approach to developing KBRs is likely valuable for other software platforms as well. Additionally, this has granted us more empirical data about how the students learned other technologies like React and APIs which did not differ much from how they learned DHIS2.

### **3.2 Research paradigm**

Existing literature has presented various research paradigms. Often, researchers distinguish between the interpretive, critical and positivist research paradigms. Goldkuhl (2012) presents the pragmatic research paradigm as another option and highlights its suitability for qualitative research in information systems, in particular for DSR. The pragmatic paradigm is based on the idea that researchers should follow the methodological approach that best suits the specific research problem (Kaushik & Walsh, 2019). Unlike other research paradigms, pragmatism does not focus on establishing one single truth. Instead, it accepts that there can be a single or multiple realities (Kaushik & Walsh, 2019). Fundamental to the pragmatic paradigm is the idea of building constructive knowledge, i.e. knowledge that is useful to the



world. The field of design is an example where constructive knowledge is applied to build effective and user-friendly solutions by combining prescriptive, normative and prospective aspects (Goldkuhl, 2012, p. 141). In our case, we aim to build prescriptive design considerations that can guide platform owners in creating knowledge boundary resources. With its focus on utility and practice at the same time, the pragmatic approach is suitable for our research goal. One research methodology that is based on pragmatism is Design Science Research (DSR) which will be further examined in the next section.

### **3.3 Methodology: Engaged Design Science Research**

The empirical research of this thesis followed an engaged Design Science Research (DSR) approach. DSR supports researchers in developing constructive knowledge through the design and evaluation of artefacts. The idea of DSR as a methodology has gained popularity in IS since the inception of the field (Gregor & Hevner, 2013). Ivari (2014) proposed two common strategies for conducting DSR. With the first strategy, a researcher builds a meta-artefact addressing a broader class of problems. With the second strategy, “a researcher attempts to solve a client’s specific problem by building a concrete IT artefact in that specific context and distils from that experience prescriptive knowledge to be packaged into a general solution concept to address a class of problem.” (Ivari, 2014, p. 107) In other words, this strategy seeks to solve existing organisational problems by creating innovative artefacts and inserting them into a real-world context. Such an approach is also referred to as engaged research because researchers and practitioners collaborate to solve real-world problems while contributing to academic literature (Li, 2021). We adopted the second strategy. In collaboration with the DHIS2 core team, we built an innovative artefact employed in the university course to explore how DHIS2 can improve their platform KBRs. From this experience, we distilled prescriptive design considerations that can guide platform owners in the design of onboarding-related KBRs.

### 3.4 Research process

For carrying out DSR, we followed the model provided by Peffers et al. (2007). It is commonly referenced in academic literature and aids researchers in conducting DSR. The model describes the DSR process as consisting of six activities (see Figure 3.2): 1) Problem identification and motivation, 2) Definition of the objectives for a solution, 3) Design and Development, 4) Demonstration, 5) Evaluation, and 6) Communication. Figure 3.3 shows a timeline of how we followed these six activities throughout our research process.

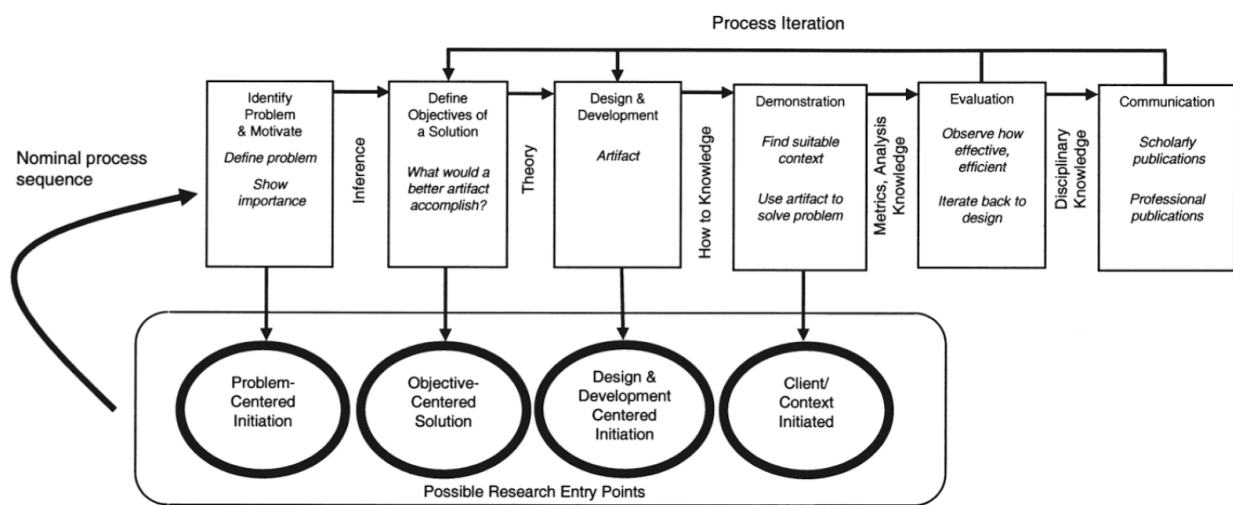


Figure 3.2: Design science research process

*Note.* From “A Design Science Research Methodology for Information Systems Research”, by Peffers et al. (2007, p. 58). Journal of Management Information Systems.

During the “Problem identification and motivation” activity, the research problem is identified and conceptualised to reason about its complexity and potential solutions. This activity started in December 2020. When the university course was previously conducted, the DHIS2 platform KBRs did not always seem to onboard students effectively. To gain an understanding of this problem, we conducted a preliminary study involving several semi-structured interviews. The main goal was to outline frequent challenges that students faced when learning to develop a DHIS2 application. During this phase, we also reviewed the academic literature to leverage existing concepts and insights to frame our research problem accordingly. Specifically, the concepts of boundary resource and knowledge boundary

resource helped us in formulating our research question. For example, we decided that our research question should revolve around the concept of KBRs.

In the “Define objectives of a solution” activity, researchers infer the objectives of a solution from the problem definition and other knowledge sources based on what is possible and feasible. We began this activity in the spring of 2021 by analysing the data from the preliminary study and identifying key challenges. Two of the important aspects that informed the design of our solution were the challenges identified in the preliminary study (see section 4.3) and the kernel theory (see section 2.2). Informed by this, we outlined a proposed solution that addressed the identified challenges.

During the “Design and development” activity, the artefact is designed and developed. We began designing and developing the artefact in April of 2021. The artefact consists of a collection of existing DHIS2 KBRs and a set of complementing KBRs. We describe the artefact in detail in Chapter 5. Much of the development was done during the summer and was finalised in August 2021.

Throughout the “Demonstration” activity, the artefact is used to solve an instance of the problem. The artefact was actively used by 137 students during the course period to develop a DHIS2 application and learn about the DHIS2 boundary resources and KBRs. The demonstration activity ended when the students were onboarded to the DHIS2 platform and completed their custom applications.

In the “Evaluation,” the artefact's effectiveness at solving the research problem is analysed and compared to the solution objectives. In our case, we evaluated the artefact simultaneously as we demonstrated the artefact. By gathering the students' feedback and observing the artefact in use, we could make adjustments to our artefact along the way. The main evaluation, however, took place after the demonstration phase. From a broader perspective, we evaluated the artefact's performance in solving the identified challenges from the preliminary study. This phase was completed in January 2022.

Finally, in the “Communication” activity, the researchers communicate all aspects of their work such as the problem it sought to solve, the artefact’s utility and the artefact’s relevance with regard to the researcher's audience. We will contribute with prescriptive design considerations (see Chapter 7) and discuss our findings in the light of academic literature.

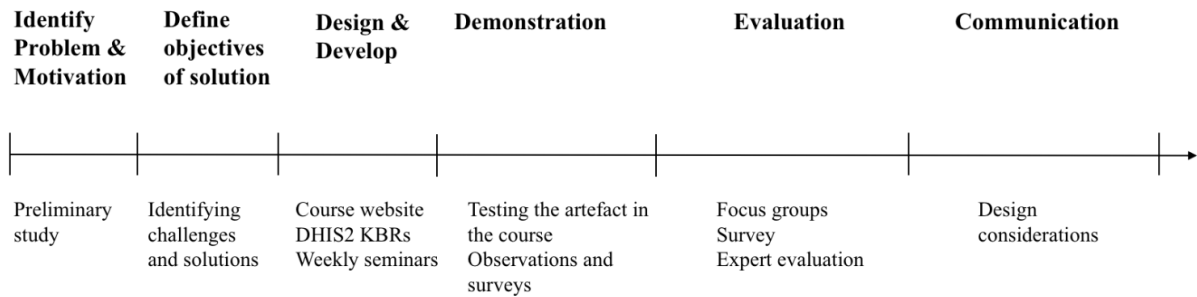


Figure 3.3: A timeline of our project

### 3.5 Data collection

Our empirical data collection mostly follows a qualitative approach. The data is derived from different data collection methods such as interviews, surveys and observations. In the academic literature, combining different data collection methods is referred to as triangulation (Flick, 2004). As different methods have different strengths, researchers can gain a greater understanding by combining different data collection methods. Therefore, we decided to triangulate different data collection methods to gain a greater understanding of the students' onboarding process. In this section, we describe the different methods that have been used to collect data with regard to the following DSR activities (see Figure 3.2): 1) Problem identification and motivation, 2) Demonstration and 3) Evaluation.

### **3.5.1 Problem identification and motivation**

During the problem identification and motivation activity, we conducted nine semi-structured interviews as part of a preliminary study. Semi-structured interviews allow a certain degree of flexibility to deviate from interview questions (Edwards & Holland, 2013). The flexibility of this approach allowed us to explore topics that the interviewees found important but we had not previously thought of.

The interview objects involved seven previous students of the course, one prior seminar teacher and the professor of the course. When selecting the interview objects, we tried to recruit students with different backgrounds and skills to capture the diversity of the students in the course. The main goal of this preliminary study was to examine challenges with the existing DHIS2 KBRs during the onboarding process and outline possible solutions for creating new KBRs. Whereas the interviews with the students focused on the experienced challenges with KBRs, the interviews with the course administrators related more to the technical aspects of the course with the goal to explore what would be feasible and useful to implement from their perspective. The leading question during our interviews with the students related to how they experienced their onboarding process to the DHIS2 platform, including the challenges they faced. We also asked questions related to their prior skills and preferred approaches to learning new technology. Some example questions were:

- How did you learn the DHIS2 API?
- Did you find the DHIS2 App Course helpful as a learning resource?

The full interview guide can be found in Appendix 5. Finally, we had several interviews with the DHIS2 core team where we presented and discussed our ideas. We also used this as an opportunity to learn about frequent problems they experienced when onboarding new complementors to the DHIS2 platform. The data collected from these more unstructured interviews also fed into the design of our artefact.

### 3.5.2 Demonstration

During the demonstration activity, a large portion of the data was gathered by observing students in the course. The strength of observation as a data collection method comes from capturing people's actual behaviour (Moen & Middelthon, 2015). The actions people take and the actions they say they take often differ. For instance, social desirability bias could cause students to reply dishonestly in surveys to please the researcher. By observing how students interacted with our artefact through the weekly seminars, we wanted to see how students utilised our artefact as part of their onboarding process, including the challenges they experienced. By holding weekly seminars during the course, we could observe the students' interaction with the artefact throughout the entire semester. Additionally, we gained a lot of insight into the challenges complementors commonly faced through answering their inquiries in the group seminars and via email. Furthermore, as seminar teachers, we corrected individual assignments and the final DHIS2 application. By inspecting the code that students delivered, we acquired detailed insight into what areas the students commonly struggled with. In addition to observations, we conducted three surveys during the semester to receive the students' feedback on our artefact (see Appendix 1, 2 & 3). The questions in the surveys were similar to the questions asked in the interviews and revolved around how students perceived the DHIS2 App course and other learning resources and KBRs used during their onboarding. Compared to the interviews, the surveys provided us with more quantitative insights as we could include a higher sample through the survey than we could by one-and-one interviews. Image 3.4 shows an example question from our survey.

From a scale 1-10 how well do you feel this assignment prepared you for the group project? \*

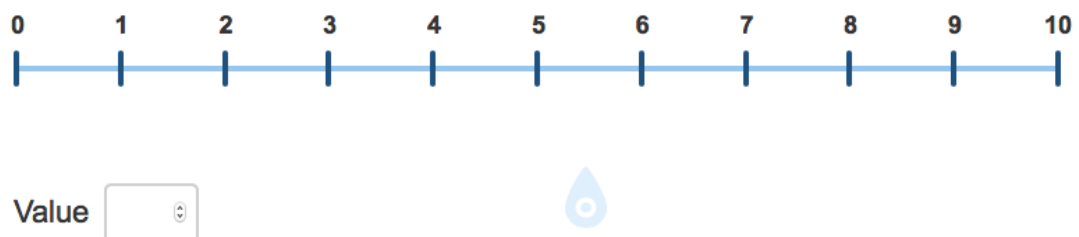


Image 3.4: Example survey question

### 3.5.3 Evaluation

In DSR, the artefact evaluation is an important activity because it demonstrates the artefact's worth by providing evidence of its utility in addressing the defined problem (Gregor & Hevner, 2013). The artefact evaluation helps to establish the artefacts utility in solving the investigated research problem. It can also help to discover potential areas for future research. To evaluate the artefact, we had to develop specific criteria upon which to measure utility. These criteria are described in Table 3.1:

<b>Evaluation criteria</b>	<b>Description</b>
Utility	Artefact has value outside of the development context in a real world context
Validity	Artefact works and does what is originally was designed for
Quality	How well did the artefact perform in achieving what it was originally designed for

Table 3.1: Evaluation criteria

We determined the artefact's utility by studying the students' perceptions and experiences with the artefact throughout the course. Testing the artefacts performance in a real world context was a large focus throughout this study. By testing the artefact in action throughout the course, we observed how students would use the artefact as part of their onboarding process and to develop a DHIS2 application. Another important evaluation criteria is related to whether the artefact met the initial design criteria (validity) as well as how well it achieved these (quality). Through our preliminary study, we found a set of key challenges with the existing DHIS2 KBRs. The artefact seeked to eliminate or reduce these challenges. The validity criteria is therefore assessed by evaluating if the artefact solved the identified challenges from the preliminary study, whereas the quality criteria evaluates how well the artefact performed in solving those challenges. We present the findings from the artefact evaluation in detail in Chapter 6.

The data collection methods we chose for evaluating the artefact were focus groups, a final survey and an expert evaluation. Focus groups share many of the same characteristics as semi-structured interviews with the difference that the researchers can collect data from many participants at once (Gill, Stewart & Chadwick, 2008) which allows the generation of collective perspectives on a phenomenon. The researcher takes the role of a facilitator and guides the discussions among the group members. It is also the researcher's responsibility to ensure that all group members get to share their thoughts and that not one of the group members dominates the entire discussion. A focus group was a natural choice because the university course divided the students into project groups where tasks were delegated between team members. By gathering the project groups, we could collect data about all of the aspects of the application development. In total, we facilitated four focus groups with four different groups of students who completed the final project as part of the university course. The questions in the focus groups revolved around collecting feedback on our artefact, how the students experienced the application development process, and what challenges they encountered. In essence, we focused on three aspects of the students onboarding process:

- What worked well?
- What did not work well?
- What could be improved?

Our surveys revolved around many of the same aspects with a slightly more quantitative approach. The final survey and detailed focus group tasks can be found in Appendix 4 and Appendix 6.

Finally, we conducted an expert evaluation with a DHIS2 core team member. The expert has previous experience developing KBRs to onboard developers from other software platforms. Due to his previous experience with creating KBRs on a platform, we considered his feedback useful to reflect on the artefact in a summative manner. The expert comprehensively reviewed our artefact and created a list of changes and ideas he had about improving it further. Afterwards, we held an unstructured interview consisting of a cognitive walkthrough of our artefact and his thoughts on them using the "think aloud method". If we came across topics of interest we would explore those further by asking further clarifying questions and discussing.



### 3.5.4 Summary of data collected

Table 3.2 provides a summary of all of the data collected throughout the project. We want to note that the observations are estimated. Further, it does not contain data collected through instant messaging, email, one-on-one assistance and weekly meetings between seminar teachers and the course professor.

Activity	Method	Who	#
Problem Identification	Interview	Platform owner	3
	Interview	Previous seminar teacher	1
	Interview	Students	7
Demonstration	Observation: Weekly seminars	Students	24 seminars 2 hours each 2 researchers ~ 96 hours
	Observation: Assignment grading	Students	3 assignments 30 deliveries 2 researchers ~ 180 assignments
	Observation: Application evaluation	Student groups	8 applications 2 researchers ~ 16 applications
	Survey: HTML, CSS, JS	Students	29 respondents
	Survey: React	Students	17 respondents
	Survey: DHIS2	Students	13 respondents
Evaluation	Survey: Final	Students	15 respondents
	Focus groups	Student groups	4
	Expert evaluation	Expert	1

Table 3.2: Summary of all data collected throughout the project

### **3.6 Data analysis**

We analysed our collected data with three major data analysis processes. The first one after the preliminary study had the goal to inform the design of the artefact whereas the second had the goal to evaluate the artefact. Finally, as part of the third analysis, we combined our empirical findings with academic literature to formulate our design considerations for onboarding complementors in a software platform ecosystem.

The analysis of our preliminary study and the analysis of our artefact evaluation were based on thematic analysis. Thematic analysis is a widely used analytic method for qualitative research that includes searching for repeated patterns and identifying themes across a data set (Braun & Clarke, 2006). A theme captures an important aspect about the data with regards to the research question. The data set consisted of interviews, observations, surveys, focus groups and an expert evaluation. Braun and Clarke (2006) describe thematic analysis as comprising six phases: 1) Familiarising yourself with your data, 2) Generating initial codes, 3) Searching for themes, 4) Reviewing themes, 5) Defining and naming themes and 6) Producing the report. Thematic analysis can be conducted in an inductive manner, meaning that the researchers do not have any preexisting coding categories and the themes develop from the data itself. The findings can also evolve deductively, driven by the researcher's theoretical concepts. We included both deductive and inductive elements in our thematic analysis (see Figure 3.4) and will elaborate further on this topic through the different phases of data analysis. Empirical research and academic literature had a rather complementary relationship. For example, the themes that evolved from the data set caused us to specify our research question, arguing for an inductive approach. However, the kernel theory, the academic literature and our own experience with the university course gave us a predefined research interest with the data set, arguing for a deductive approach. Another important aspect of our analysis has been continuous internal discussion. We have benefited greatly from being two researchers throughout the entire research process by discussing findings and complementing each other with different perspectives.

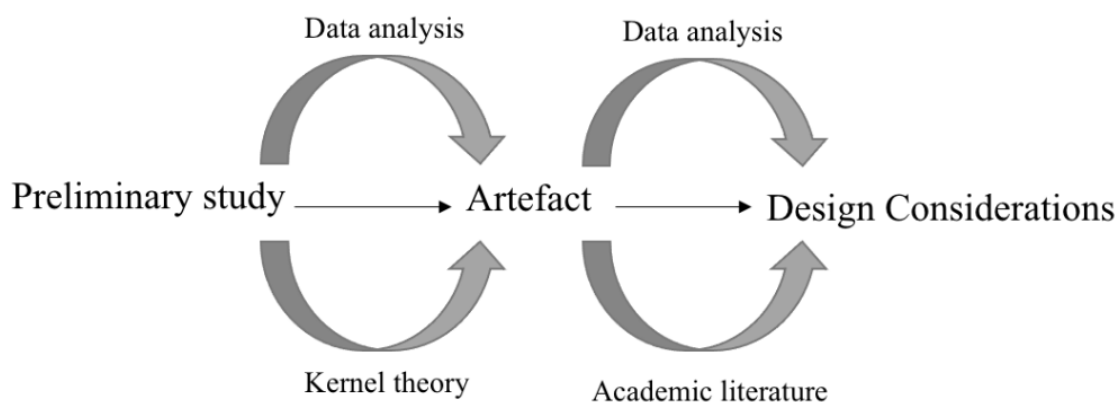


Figure 3.4: Deductive and inductive elements of data analysis

### 3.6.1 Preliminary study

The analysis of our preliminary data was essential for the design of the artefact. From the beginning, we sought to identify any challenges that the students experienced throughout the university course that hindered their onboarding process. We first familiarised ourselves with the situation by reading through our notes from the interviews and discussing them with each other. Following this discussion, we proposed initial codes and began to code the data systematically, looking for interesting or notable features. We used our prior experiences of being enrolled in the course to reason through some of these identified challenges. In the next phase, we discussed the findings and codified these findings on post-it notes. By organising the post-it notes and grouping them together as themes, we identified different categories of challenges. These challenges, together with our kernel theory, motivated the design of our artefact. Table 3.3 shows one of the four main challenges identified throughout our analysis.

Theme	Code	Quote
Lack of assistance	Improvements	“I wish there were practical seminars in the course where you can get a little help from the seminar teachers or fellow students.”
	Challenges	“There was nowhere to go to ask for help other than fellow students.”

Table 3.3: Example of theme from preliminary study

### **3.6.2 Evaluation of the artefact**

Many of the questions that drove the analysis concerned the impact of the DHIS2 App Course on students' onboarding throughout the university course. After our preliminary study, we identified a set of challenges that previously hindered students onboarding. Therefore, the goal of the evaluation was to see if and how well the artefact addressed these challenges. Additionally, we wanted to examine how useful the students perceived it in relation to their application development. We also sought to identify characteristics of KBRs that enabled effective ways of teaching students the necessary knowledge required to develop DHIS2 applications.

We began the evaluation of the artefact by transcribing our records from the focus groups into text. Further, we repeatedly read the transcripts and notes from previous observations, interviews, surveys and focus groups to familiarise ourselves with the breadth and depth of the data. This was a challenging yet important step in our analysis due to the large amount of data that we had gathered. We had to decide which aspects of the data we wanted to focus on and which would be out of scope based on the research question. Once again, the challenges that students experienced while onboarding and any feedback related to the DHIS2 App course were the main focus areas. We also focused on the students themselves. For example, by looking for differences in their preferred learning approaches, we could glean insight into the variety of learning methods which informs our research.

After agreeing on some initial codes, we coded the data systematically to find interesting aspects (see Image 3.5). To prevent groupthink and gain further insight, we coded the transcripts and notes independently from each other. Some examples of the initial coding categories include "Challenges with DHIS2's boundary resources", "Challenges with DHIS2's KBRs" and "Feedback on the DHIS2 App Course". Whereas the first two categories focused on platform resources provided by the core team, the last category focused on our App Course website. This allowed us to evaluate the entirety of KBRs used throughout the course rather than isolating the DHIS2 App Course from other platform resources.

During the next phase, we compared and discussed our findings. As the artefact was designed to solve some of the challenges identified in the preliminary study, we iterated back to evaluate if these challenges were actually resolved after the introduction of our artefact. New findings from the data analysis caused us to rename the themes. For example, we found that we could collapse several of the themes to a common larger theme. We present our findings from our artefact evaluation in detail in Chapter 6. These empirical findings served as the foundation for the development of our design considerations. We describe how we developed the design considerations further in the next section.

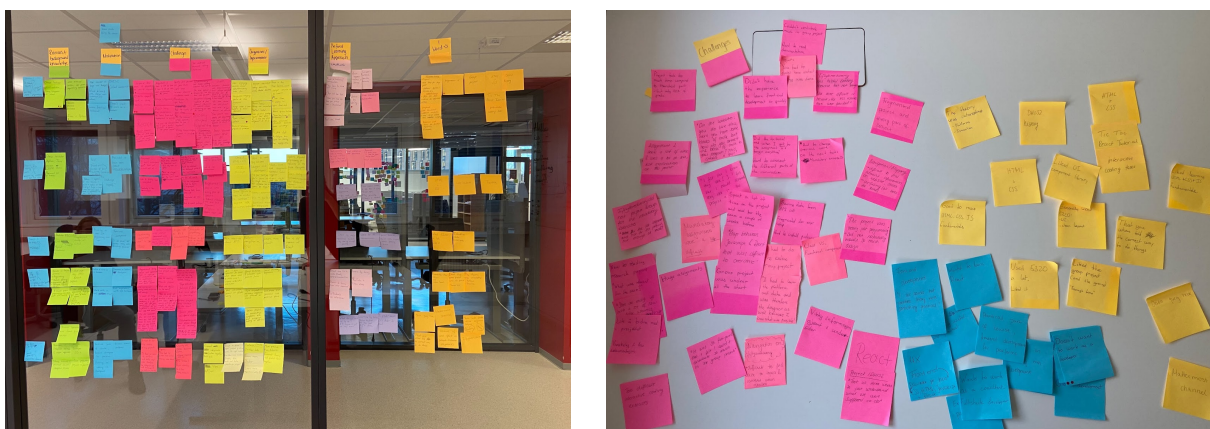


Image 3.5: Thematic analysis

### **3.6.3 Developing design considerations**

The final analysis process of our study involved the development of design considerations for guiding platform owners in designing KBRs for onboarding complementors. Empirically, the design considerations are based on the broader themes identified throughout the data analysis but we discuss them from a theoretic stance by including relevant academic literature. To formulate design considerations, we reflected upon the many interesting insights, data points and existing research we had encountered, much of which was driven by trying to understand how we could use KBRs to onboard students to DHIS2. After numerous interviews, focus groups and observations with students and the core team, we had to process large amounts of data.

Therefore, we had to select which data should go into the last “round of analysis” - the development of design considerations. The selection criteria for the development of design considerations focused on three questions:

- How relevant are the design considerations for other software platform owners?
- How useful are the design considerations for the onboarding of complementors?
- Do the design considerations present novel knowledge?

First, we had to decide on whether our findings could be generalised to other software platforms. In the beginning, we had many ideas on possible design considerations. Because we took the role of seminar teachers and assisted the students throughout the university course, we gained detailed insight into their onboarding process. However, not all of this insight could be generalised to the broader context of onboarding of complementors in software platforms. As mentioned earlier, there were some differences between the students in the university course and complementors in other software platform ecosystems. For example, there may be differences in the interpersonal dynamics of the project groups within a school environment and complementor teams in other software platforms. Findings related to how students collaborated in teams to develop an application may not be entirely applicable to other software platform ecosystems. Further, our design considerations do not focus on the details of equipping complementors with basic software development competencies. Our role as seminar teachers focused strongly on furnishing students with basic software development skills due to the students’ lack of prior experience with platform application development. Complementors in other software platform ecosystems are likely to have basic software development competencies and experiences with application development. Second, we selected only design considerations that we found to be useful in onboarding complementors and believed other software platform owners would benefit from. We strived to create design considerations based on what we had observed to be most impactful in the course and also provided the most effective onboarding experience. Our large data collection and close relationship with the students throughout their application development process enabled us to gain a detailed understanding of what aspects of a KBR are important during an onboarding process. Finally, we selected only design considerations that presented novel prescriptive knowledge that was not yet covered in existing research or extended existing research.

### **3.7 Ethical considerations**

Throughout the project we had to make some ethical considerations. First, researchers are responsible that no harm is done to the research participants. This includes ensuring the integrity and confidentiality of the participants. Therefore, all participants were provided with an informed consent form before conducting interviews and focus groups. In addition, to protect the people included in our study, all data collected has been anonymised. Therefore, we do not use any names when we quote participants of our study.

Additionally, because we were employed as seminar teachers in the university course, we knew many participants personally and gained a closer relationship with them throughout the semester. Throughout the university course, we performed as both researchers and seminar teachers. This raised ethical considerations with regard to the power dynamics between us and the students. For instance, students may get the impression that they have to consent to the data collection, fearing that otherwise they would experience direct or indirect consequences in the classroom. We paid close attention to this during the entire project and repeatedly informed students that all data collection was optional.

## **4. Existing Artefact and Challenges**

Our artefact is a collection of different knowledge boundary resources. Because the students were exposed to many different KBRs, looking at one KBR in isolation does not capture the full picture of onboarding. By regarding the artefact as a collection of KBRs, we can gain greater insight into onboarding as a whole. In this chapter, we first provide an overview of the most essential boundary resources on the DHIS2 platform. Then we present the most significant KBRs used throughout the course. Finally, we highlight the most consequential challenges that students faced when using the KBRs and hindered their onboarding to the DHIS2 platform. These challenges were identified as part of the preliminary study and fed directly into the artefact design.

### **4.1 DHIS2 boundary resources**

It is essential to have a basic understanding of the DHIS2 boundary resources to understand their respective knowledge boundary resources. We briefly cover the most important boundary resources and the necessary details about them to understand this thesis. The boundary resources covered include the DHIS2 API, Data Queries, UI component library, and the DataStore. Further, because DHIS2 contains sensitive health data, the DHIS2 platform is designed to be “self-hosted” on the servers of the implementing organisation, in contrast to DHIS2 centrally managing all DHIS2 servers. We refer to these specific servers hosted by the implementing organisations as a “DHIS2 instance”.



### 4.1.1 DHIS2 API

The primary manner applications interact with a DHIS2 instance is through the DHIS2 API (Application Programming Interface). The API allows applications to access DHIS2 functionality and read, update or delete data on a DHIS2 instance. The DHIS2 API has a large functional extent with over 80 different endpoints. An API endpoint is essentially a specific resource that can be accessed through an API. Similar to how a website returns a different web page when you request [dhis2.org/overview/](https://dhis2.org/overview/) than [dhis2.org/about/](https://dhis2.org/about/), the DHIS2 API returns different data when you query a different endpoint. Because DHIS2 is designed to collect large amounts of varied data, it has a complicated data model (see Figure 4.1). The DHIS2 data model is a representation of the relationships between the data returned by different endpoints. To access the data on a DHIS2 instance, developers must have an understanding of where certain data is located within the data model and query it through the API. Learning the data model and how one can query it is one of the key tasks complementors encounter when developing DHIS2 applications.

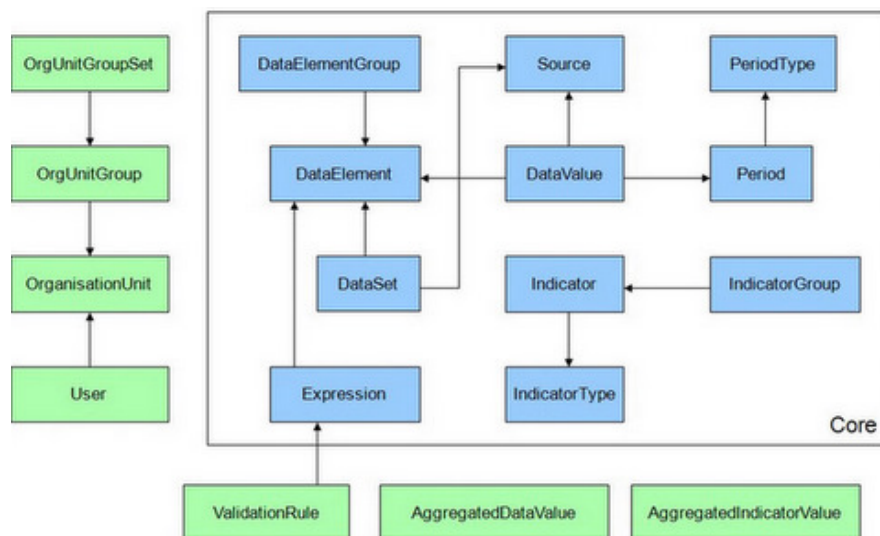


Figure 4.1: A subset of the DHIS2 data model

*Note.* From DHIS2 Documentation (n.d.)

### 4.1.2 Data Queries

Data Queries provide an alternative method of querying the DHIS2 API in a React application. It is possible to query the DHIS2 API directly in a React application; however, using Data Queries enables a more efficient developer workflow. One of the advantages of using Data Queries is that developers do not need to program logic to create HTTP requests or manage network errors. The details around Data Queries are not incredibly important for this thesis. For this thesis, it is sufficient to know that it is an alternative method of querying the DHIS2 API with a slightly different interface design (see Image 4.1). The image below shows implementation differences between a conventional API query and a query performed using Data Query.

```
API Query:  
/api/dataSets?pageSize=5&fields=id,displayName  
  
Data Query:  
{  
  "dataSetsQuery": {  
    "resource": "dataSets",  
    "params": {  
      "pageSize": 5,  
      "fields": ["id, displayName"]  
    }  
  }  
}
```

Image 4.1: Comparison between an API query and a Data Query

### 4.1.3 UI component library

User interface (UI) components are the visual building blocks of a React application. They are small, single-purpose UI elements. Examples include simple components such as buttons, text fields and icons and also more advanced components such as tables. A UI component library is a set of ready-made UI components that are commonly used in applications. These UI components can be arranged together in different ways to create novel and custom application interfaces. The DHIS2 platform provides a UI component library that features over 70 visually consistent UI components. These UI components are specifically designed to fit DHIS2 applications by covering specific functionality that many applications will encounter. Image 4.2 offers some examples of DHIS2 UI components commonly used in applications.

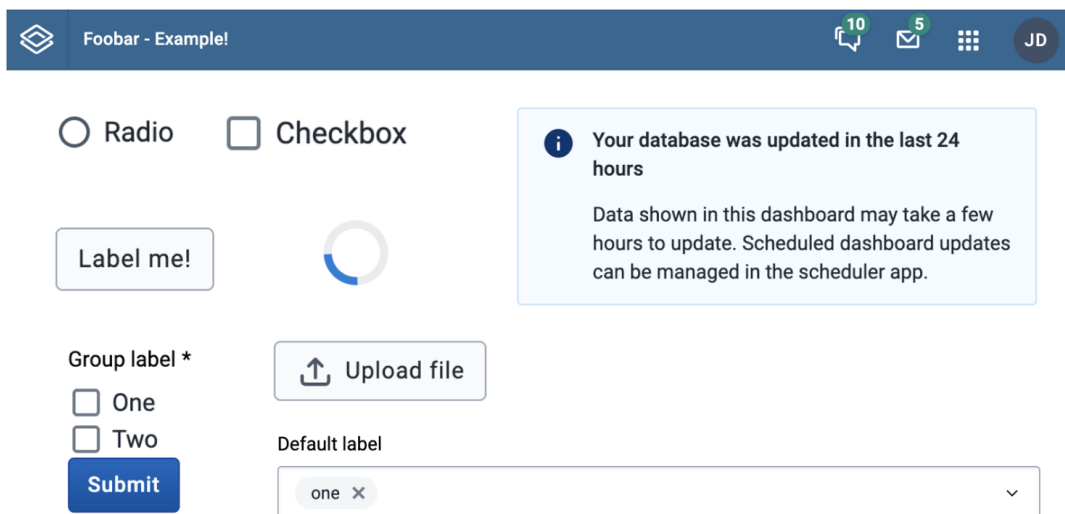


Image 4.2. Examples of DHIS2 UI components.

## 4.1.4 Datastore and Datastore Manager

The Datastore is a key-value database that allows applications to store arbitrary data on a DHIS2 instance. For example, it can be used to save user preferences across computers and sessions. The Datastore can be accessed by applications through the DHIS2 API.

The Datastore Manager (see Image 4.3) is a DHIS2 application developed by the core team and is built on the Datastore API. It provides a graphical user interface where users can access the Datastore API functionality and save, modify and delete data. It is designed to assist with modifying the Datastore without having to interact with the API.

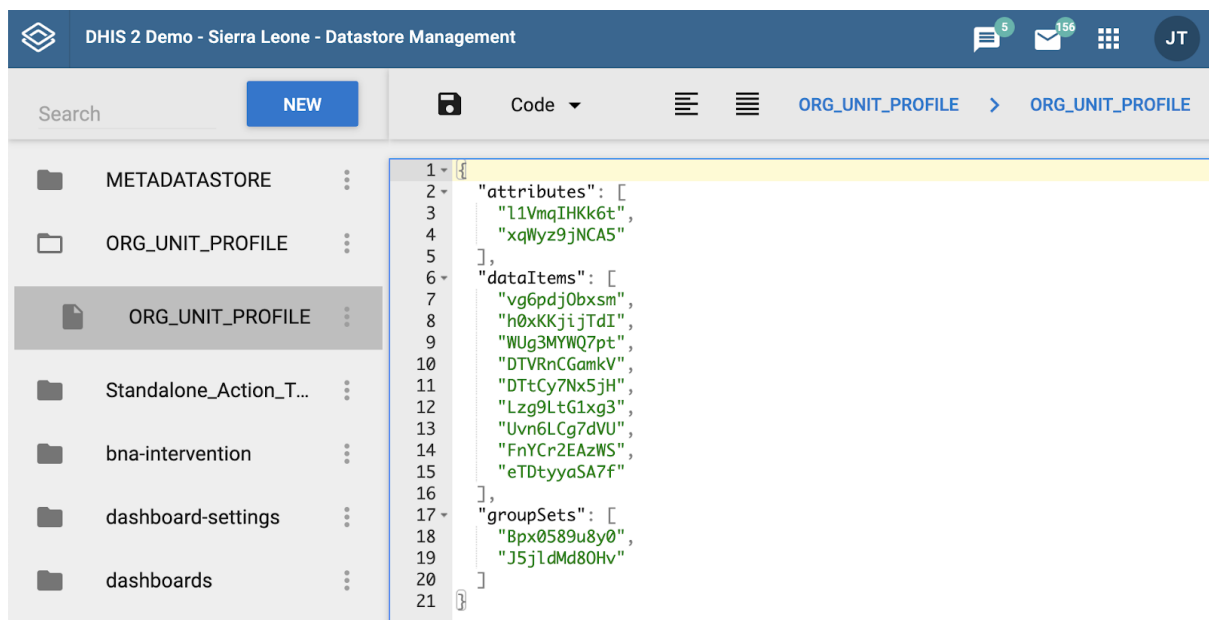


Image 4.3: Datastore Manager

## **4.2 DHIS2 knowledge boundary resources**

During the university course (Development in Platform Ecosystems), the students use many different KBRs during their onboarding to DHIS2. The most important KBR for the first phase of the university course is the comprehensive online course “DHIS2 App Course”. However, during their group project, they use many different KBRs provided by DHIS2 to develop an application. This section provides a brief explanation of the most important KBRs for this thesis.

### **4.2.1 DHIS2 App Course**

The most important KBR for the students enrolled in the university course is the “DHIS2 App Course”, a website we were responsible for developing and maintaining. The website is actively used by students throughout the university course to learn about the DHIS2 platform.

The DHIS2 App Course is designed to support students in developing a DHIS2 application as part of their onboarding to the platform. The website is divided into six modules (see Image 4.4). The first module contains general information about the course and instructions on how to set up a development environment. The second and third modules cover essential web development technologies including HTML, CSS and Javascript (see Image 4.5). The fourth module covers the Javascript library React and the fifth covers DHIS2. The sixth module contains mandatory assignments students are required to pass to progress in the course.

When we began our study, the skeleton of the website already existed. However, it lacked important content. The first three modules of the website were complete, containing comprehensive resources about HTML, CSS and Javascript. However, the React and DHIS2 sections were incomplete, lacking important content. As these modules play an important role in onboarding students to the DHIS2 platform, completing these resources became one of the most important parts of our study. We will explain the details of these modules on the App course website later (see Chapter 5).

## Topics

<h3>Getting started</h3> <p>An introduction to the course as well as a quick development environment setup.</p>	<h3>Essential front-end development</h3> <p>How to make static web sites with HTML and CSS.</p>
<h3>JavaScript</h3> <p>The programming language JavaScript and how we can add interactivity to otherwise static web sites.</p>	<h3>React</h3> <p>The JavaScript library React which is an extremely popular library for building user interfaces.</p>
<h3>DHIS2</h3> <p>The health management information system DHIS2 and how we can create custom apps for it.</p>	<h3>Mandatory Exercises</h3> <p>Details about the mandatory individual exercises.</p>

Image 4.4: DHIS2 App Course modules

The screenshot shows a web page titled "Development in platform ecosystems" with navigation links for "HOME" and "LEARN". A sidebar on the left lists the course structure under "ESSENTIAL FRONT-END DEVELOPMENT", with "HTML (markup)" selected. The main content area is titled "HTML Elements" and includes a "Headings" section. The "Headings" section explains the importance of titles and lists heading tags from `<h1>` to `<h6>`. Below this is a "Code example" section with a code block containing `<h1>My main title</h1>` and `<h2>Subtitle</h2>`, and a "Copy" button. An "Accessibility" section follows, stating that headings should be larger and more distinct than body text to aid users with disabilities.

Development in platform ecosystems HOME LEARN

ESSENTIAL FRONT-END DEVELOPMENT

- HTML (markup)
  - Introduction
  - Elements
    - Headings
    - Paragraphs
    - Images
    - Links
    - Lists
    - Dividers
    - Buttons
    - Text formats
    - Inputs
    - HTML5
  - Project
- CSS (styling)

## HTML Elements

### Headings

Titles are important everywhere; particularly on web pages. They structure documents and help users get a sense of the web page's organization and structure.

We can create headings using the `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` and `<h6>` tags, where `<h1>` is the most important heading and `<h6>` the least important.

#### Code example

```
<h1>My main title</h1>
<h2>Subtitle</h2>
```

Copy

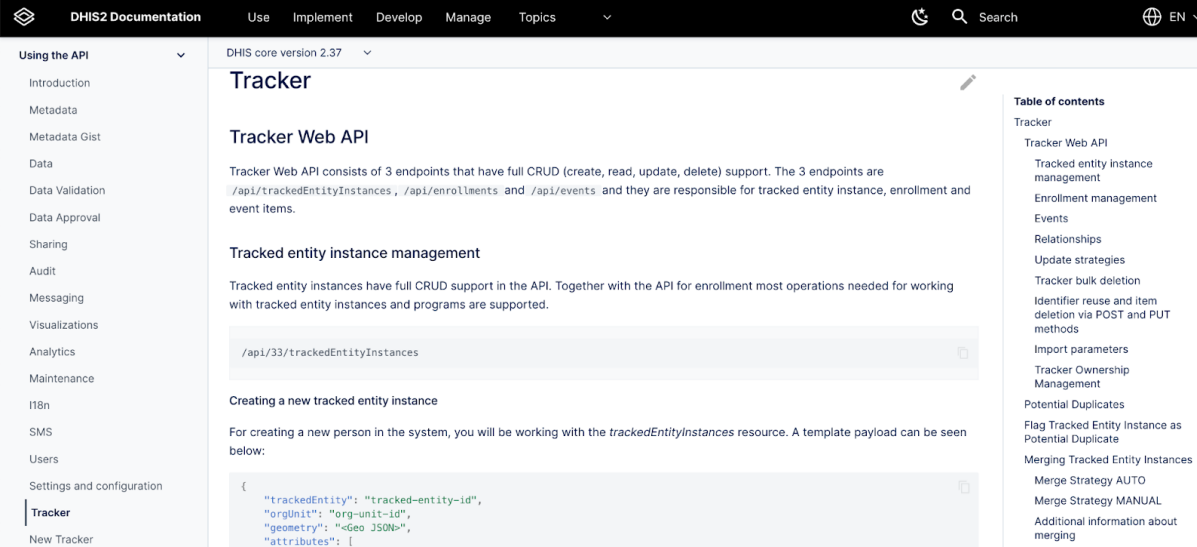
#### Accessibility

Headings on web pages should to be larger and easier to spot than the rest of the body text. We should make an effort in making headings visually distinct to help users with disabilities.

Image 4.5: Example of DHIS2 App course

## 4.2.2 DHIS2 references

The DHIS2 references are the most comprehensive DHIS2 KBR and cover the entire functional extent of the platform. It contains detailed descriptions of the interface design of all API endpoints and includes high-level explanations of different parts of the DHIS2 system (see Image 4.6). Except for the DHIS2 App Course, this is the most frequently used KBR and a large part of our project has been based on the challenges students face while working with it.



The screenshot displays the DHIS2 Documentation website. The top navigation bar includes 'DHIS2 Documentation', 'Use', 'Implement', 'Develop', 'Manage', and 'Topics'. A search bar and a language selector (EN) are also present. The left sidebar lists various sections under 'Using the API', with 'Tracker' selected. The main content area is titled 'Tracker' and 'Tracker Web API'. It explains that the Tracker Web API has full CRUD support and lists three endpoints: `/api/trackedEntityInstances`, `/api/enrollments`, and `/api/events`. A section titled 'Tracked entity instance management' describes the API's support for creating, reading, updating, and deleting instances. Below this, a code block shows a JSON payload for creating a new tracked entity instance: 

```
{  "trackedEntity": "tracked-entity-id",  "orgUnit": "org-unit-id",  "geometry": "<Geo JSON>",  "attributes": {
```

. A 'Table of contents' on the right lists various Tracker-related topics, including 'Tracked entity instance management', 'Enrollment management', 'Events', 'Relationships', 'Update strategies', 'Tracker bulk deletion', 'Identifier reuse and item deletion via POST and PUT methods', 'Import parameters', 'Tracker Ownership Management', 'Potential Duplicates', 'Flag Tracked Entity Instance as Potential Duplicate', 'Merging Tracked Entity Instances', 'Merge Strategy AUTO', 'Merge Strategy MANUAL', and 'Additional information about merging'.

Image 4.6: DHIS2 References

*Note.* From DHIS2 References (n.d)

## 4.2.3 Storybook

Storybook is a website implemented by the core team to display the DHIS2 UI components in a visual and interactive format (see Image 4.7). The website was designed to present UI components from the UI component library. It contains descriptions of all UI components and the functionality they provide. A strength of Storybook is that it lets complementors interactively test and browse UI components in a web interface. After a complementor has found a component and modified it to fit their specific use case, they can simply copy the source code from the web interface and paste it right into their application.

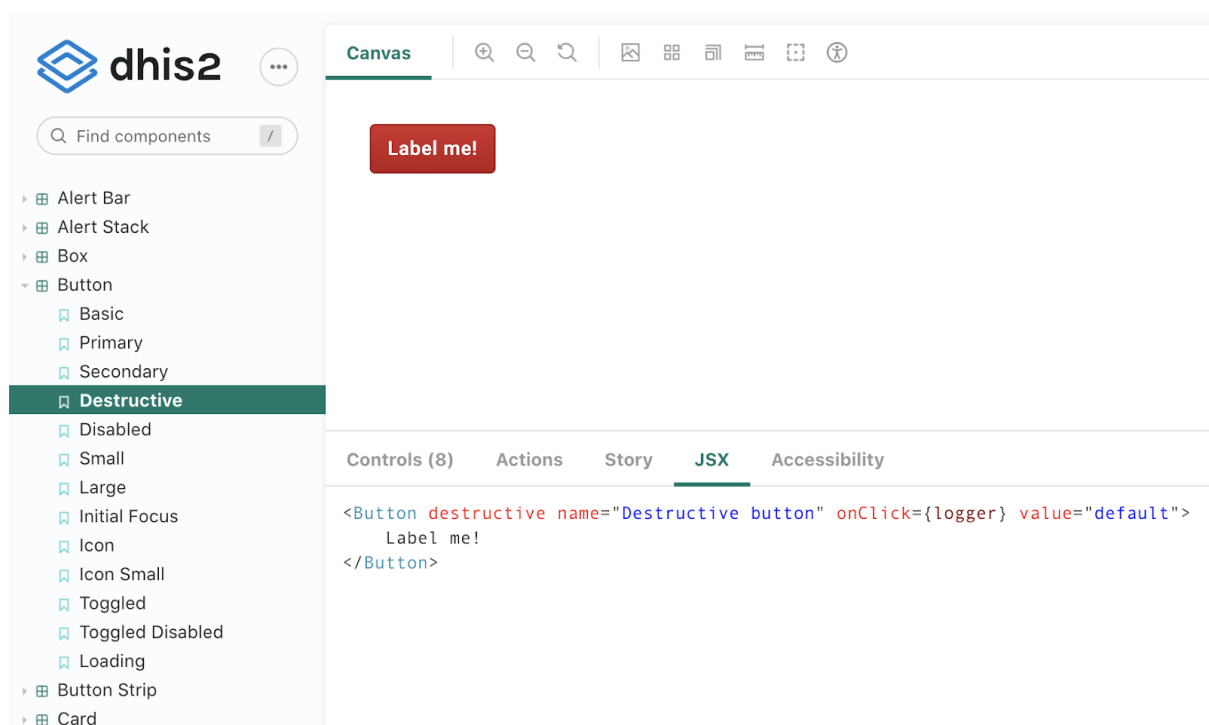


Image 4.7: A button component in Storybook

*Note.* From DHIS2 Storybook (n.d)



## 4.2.4 Data Query Playground

The Data Query Playground (DQP) is a DHIS2 application developed and maintained by the DHIS2 core team. DQP allows users to create and execute Data Queries directly in a web interface instead of querying it from an application (see Image 4.8). It then presents the response or error message from the Data Query that was performed. It is used by complementors to rapidly create and test Data Queries before they are integrated into their application. The image below shows the Data Query on the left side and the retrieved response from the API on the right side.

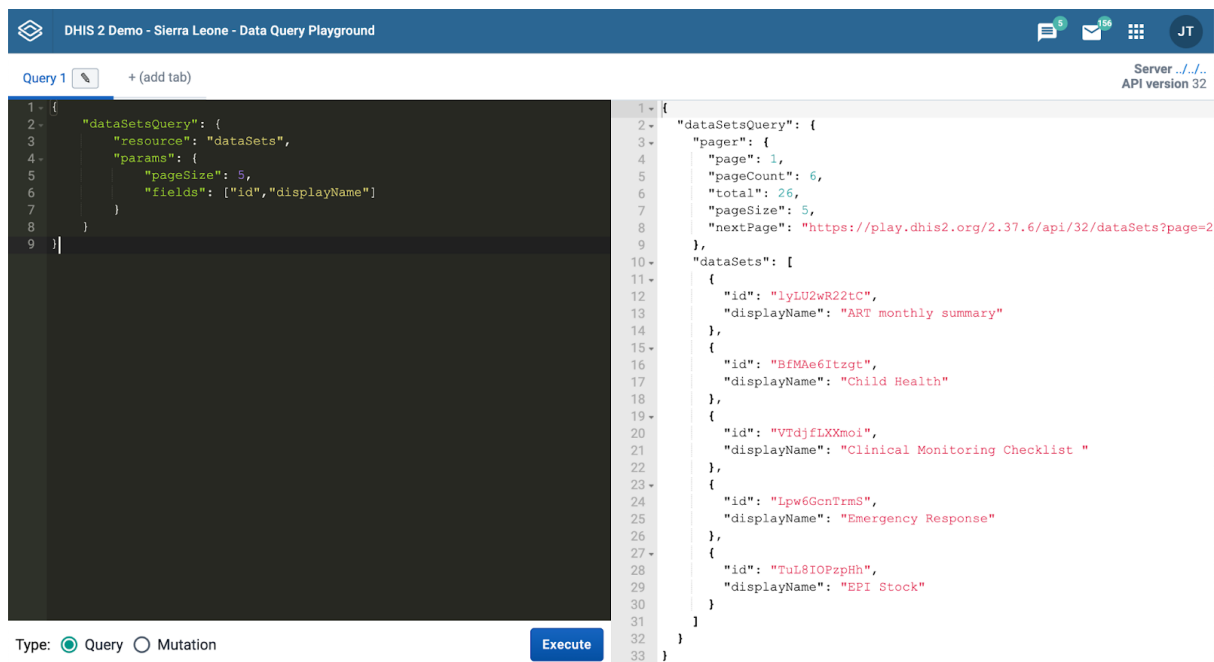


Image 4.8: Data Query and response from the DHIS2 API.

*Note. Screenshot from Data Query Playground*

### 4.3 Challenges identified in preliminary study

As mentioned in section 3.4, we held interviews with seven students, one seminar teacher and the course professor before designing the DHIS2 App Course. Additionally, we had taken the course ourselves and had personal insight into what challenges students commonly faced. From this, we identified a set of key challenges that students faced during the course, hindering their onboarding process (see Table 4.1). Together with the kernel theory, these challenges informed the design of the artefact.

Identified Challenge	Description of Challenge
Limited non-platform specific knowledge	Limited experience with web development and APIs
Complicated platform	High functional extent Challenging interface design Complicated data model
Insufficient DHIS2 KBRs	References are difficult to learn from Need for a smoother learning experience Lack of explanations and examples Large amount of information Fragmentation Confusing documentation
Lack of assistance	No or little opportunity to ask questions when problems arised

Table 4.1: Summary of identified challenges

### **4.3.1 Limited non-platform specific knowledge**

Developers require a set of non-platform specific knowledge, skills and competencies to develop applications on a platform. By non-platform specific knowledge, we specifically refer to skills and knowledge that are prerequisites or beneficial in building an application on a platform. In the case of DHIS2, experience with HTML, CSS, Javascript and React are required non-platform specific skills to build a DHIS2 application. It is also necessary to have some experience querying APIs and working with datasets. Multiple students reported that they had not learned these prerequisite skills well enough. Some reported that they did not understand or learn React well enough and others noted challenges working with APIs. This lack of a foundation of non-platform specific knowledge led to downstream challenges when developing DHIS2 applications.

### **4.3.2 Complicated platform**

DHIS2 is inherently a complicated software platform. It has a large functional extent and at times an inconsistent interface design. New complementors to DHIS2 are essentially required to learn a lot about the platform and its intricacies before being onboarded to the platform. Most students reported difficulties with using the API and learning the data model. As illustrated in Figure 4.1, there is a lot of interconnected data with obscure nomenclature and it is vital that students learn this platform-specific knowledge. By platform specific-knowledge, we mean knowledge that is specific to the platform and not directly applicable elsewhere. In addition, the DHIS2 platform is designed to address an extremely broad range of different use contexts. Several students noted that DHIS2 is their first experience working with an enterprise software system. The size of DHIS2, and how complicated it is, was one of the biggest challenges when onboarding new developers to the platform.

### **4.3.3 Insufficient DHIS2 KBRs**

Every student we interviewed noted challenges when working with the broadcasting DHIS2 KBRs, particularly the DHIS2 references. The DHIS2 references describe the platform boundary resources from a technical point of view. Therefore, it contains many technical terms that new complementors to the platform do not understand fully. In addition, the DHIS2 references are extremely comprehensive in that they comprise all information on all possible use cases of the DHIS2 platform. The specific challenges varied but many reported that some endpoints were poorly documented or lacked critical information required to complete their tasks. Others reported that the references contained a large amount of information, making it difficult to find the specific information they needed. Complementors new to the platform do not necessarily know what specific information they are looking for or what specific questions to ask to find a solution. For example, asking what filters one can apply to the API endpoint requires a student to understand API endpoints to begin with. Another frequently documented issue was that the DHIS2 references lacked explanations, particularly around the data model and the API. Further, the documentation lacked code examples, causing students to struggle in integrating the descriptive knowledge into their code. Finally, students found the DHIS2 KBRs fragmented. There were many different websites, each describing a small part of DHIS2, making it difficult to get an understanding of where one should look for certain information. This posed a challenge for students in finding the correct information with regard to completing specific tasks. The challenges identified in this theme vary; however, they are all related to challenges during the onboarding process caused by KBRs.

### **4.3.4 Lack of assistance**

Finally, students highlighted that they perceived a lack of assistance when they had issues with the implementation of a DHIS2 application relying solely on broadcasting KBRs provided by DHIS2. When they faced friction in development, they had to solve their problems on their own or with the help of other students. Such issues could, for example, include technical difficulties as well as problems with making sense of the DHIS2 documentation. When the university course was held in the autumn semester of 2020, the COVID-19 pandemic caused the weekly seminars to be digital. In addition, there was only one single seminar teacher responsible for holding the weekly seminars. This lack of

assistance when students encountered challenges would halt the onboarding process completely and lead to a lot of frustration.

## 5. Artefact Description

To reiterate, our artefact is a collection of the DHIS2 App Course (see [dhis2-app-course.ifi.uio.no](https://dhis2-app-course.ifi.uio.no)) and other DHIS2 KBRs. In this chapter, we present our interventions to address the identified challenges from the preliminary study. Based on these challenges, we created several KBRs that had the goal to eliminate or reduce these challenges. We also describe the reasoning behind important decisions made during the design and development of the artefact.

### 5.1 Curriculum design

To get an overview of the topics that we needed to cover during the onboarding process, we created a curriculum. We did this by mapping out all technologies and functionality that a student was likely to encounter when developing an application on DHIS2 and their respective prerequisite skills. We reasoned that students required proficiency in a couple of key skills to develop a DHIS2 application. The curriculum consisted of the following topics:

1. Front-end development with HTML, CSS and Javascript.
2. Developing applications with React.
3. Querying REST APIs.
4. A basic understanding of the DHIS2 Data Model.
5. Querying DHIS2 in a React application.
6. Using DHIS2 UI components in a React application.

HTML, CSS, Javascript, React and working with APIs are necessary non-platform specific skills for developing an application on DHIS2. The final three proficiencies are DHIS2 platform-specific skills. The curriculum includes a considerable amount of topics to cover. Covering HTML, CSS, Javascript and React alone could be a full university course. Therefore, we had to heavily prioritise what aspects the curriculum should cover. In other words, we had to increase the specificity of the curriculum to be as relevant to DHIS2 application development as possible. For instance, we did not cover a couple of key concepts in React such as routing because it is not necessary to develop an application on DHIS2. Furthermore, it is possible to reduce the amount students need to learn by providing specific instructions and solutions for tasks they may encounter. For example, we did not expect

students to learn the DHIS2 data model in-depth, leading us to reduce the comprehensiveness of those sections to the bare essentials. We also gave the students code that would fetch the required data for them. Code examples are a highly specific KBR. By choosing between creating more specific resources for certain areas of the curriculum, and more comprehensive resources for others, we could reduce the amount that students needed to learn about less important topics while focusing on the core knowledge required for developing DHIS2 applications. Many less important topics were either mentioned briefly or completely skipped. We did, however, sometimes provide links to where the students could retrieve more information if interested.

This prioritisation and balancing act attempted to address most challenges that students had previously. By providing a step-by-step curriculum, we aimed to introduce students to DHIS2 application development more gradually. Additionally, by prioritising the curriculum towards skills that were specific for application development we could reduce the amount of unnecessary knowledge the students had to learn.

## **5.2 Non-platform specific KBRs**

From the preliminary study, we identified the lack of non-platform specific knowledge as an obstacle to onboarding. Multiple students reported challenges when working with React and APIs specifically. The problems arose partly due to the students varying background knowledge and partly because the existing artefact did not include any non-platform specific KBRs that helped students acquire the relevant skills. By creating non-platform specific KBRs, we aimed to improve the students' capabilities with these technologies. This way, we could also ensure that everyone was on the same skill level before introducing DHIS2 application development.

## 5.2.1 React module

As the React module on the App course website was incomplete, it was a natural KBR to create. React is a required non-platform specific skill that students must acquire to develop a DHIS2 application. This module has three sections. The first section contains an introduction and a set of explanations for fundamental React concepts. By creating this section, we communicated which React concepts students were required to know and provided a brief explanation of them (see Image 5.1). The second section is a guide on how to set up a React project. We wanted all students to follow a standardised way of setting up an application to reduce potential problems. The third section contains links to several high-quality and free tutorials that covered React at a sufficient level for the course. This let the students choose which or how many tutorials they wanted to do to get more practical experience with React.

An important choice that emerged during the development of this section was deciding whether we should create React content ourselves or link to existing external React resources. Because there is a lot of quality React content online, we decided to focus more on curating and linking to the most relevant external resources. By linking we saved resources on content production and could find higher quality resources than we could have made ourselves. This also has the benefit of reducing future maintenance costs because web development is a field that changes rapidly and practices may become outdated.

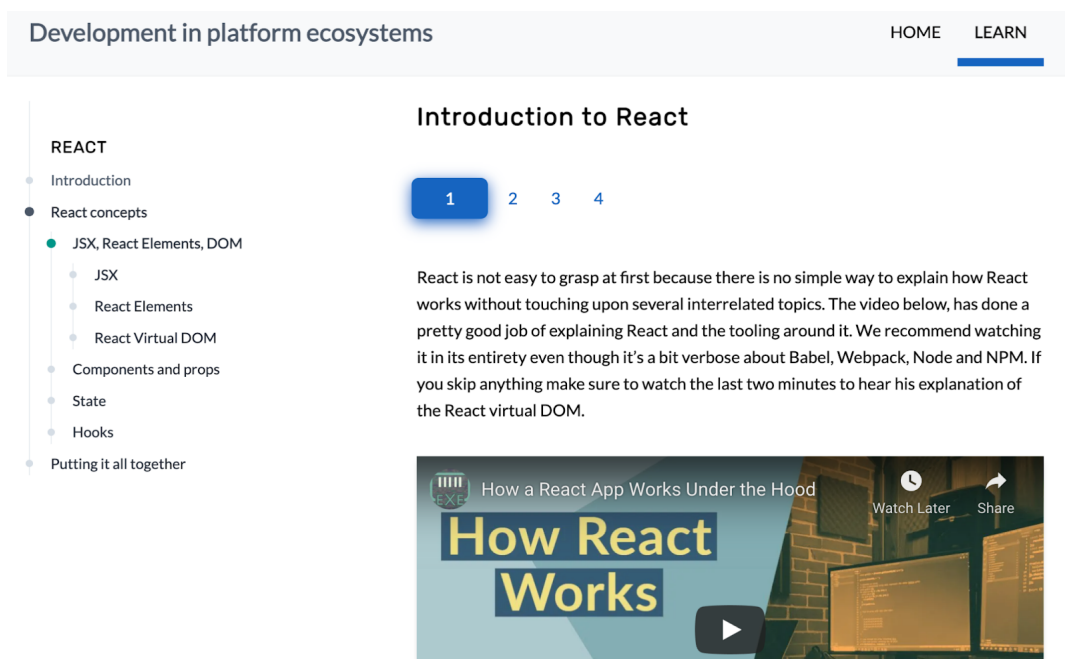


Image 5.1. React section on DHIS2 App Course



## 5.2.2 React and API Assignment

The React and API assignment was designed to give students more practice with the prerequisite skills required to build an application on DHIS2. For this assignment, we designed and implemented a custom API containing population data about countries. Because we wanted to prepare the students for working with DHIS2, we modelled our API to resemble the DHIS2 API. The students were required to query our API in a React application and present it in a table. They would then progressively add more functionality that utilised the APIs functionality such as a search bar, pagination component and sorting by column (see Image 5.2). By designing the assignment to start simple and then progressing to more advanced topics, the students could learn essential React topics more gradually. By providing an assignment that encompassed the specific parts of React required for building a DHIS2 application we could provide them with hands-on experience that aimed to better prepare them for building a DHIS2 application.

**Country Search**

Country	Continent	Population	Population Growth
Nigeria	Africa	200963599	2,56
Ethiopia	Africa	112078730	2,58
Democratic Republic of the Congo	Africa	86790567	3,19
South Africa	Africa	58558267	1,32
Tanzania	Africa	58005463	2,95
Kenya	Africa	52573973	2,27
Uganda	Africa	44269594	3,56
Algeria	Africa	43053054	1,93
Sudan	Africa	42813238	2,39
Angola	Africa	31825295	3,24

Page 1 of 5

Results per page:

Image 5.2: Example solution of React and API assignment

### 5.3 Platform-specific KBRs

Throughout our preliminary study, we found that many students had challenges when working with the DHIS2 KBRs. New complementors had to rely on the DHIS2 references to learn about the platform specifics which caused challenges for many students. The identified challenges varied but we concluded that the KBRs provided by DHIS2 were insufficient to onboard students. Guided by our kernel theory, we created a tutorial and a set of how-to-guides. The tutorial introduced the students to DHIS2 application development, including the most important BRs and KBRs. The how-to-guides assisted them with specific tasks they were likely to encounter that were out of the scope of the tutorial. We also included explanatory materials in the tutorial to aid the students in gaining a better understanding of the DHIS2 API and Data Model. Image 5.3 shows the structure of the tutorial “Getting started with DHIS2 development” on the left side and the how-to guides “App development guides” on the right side.

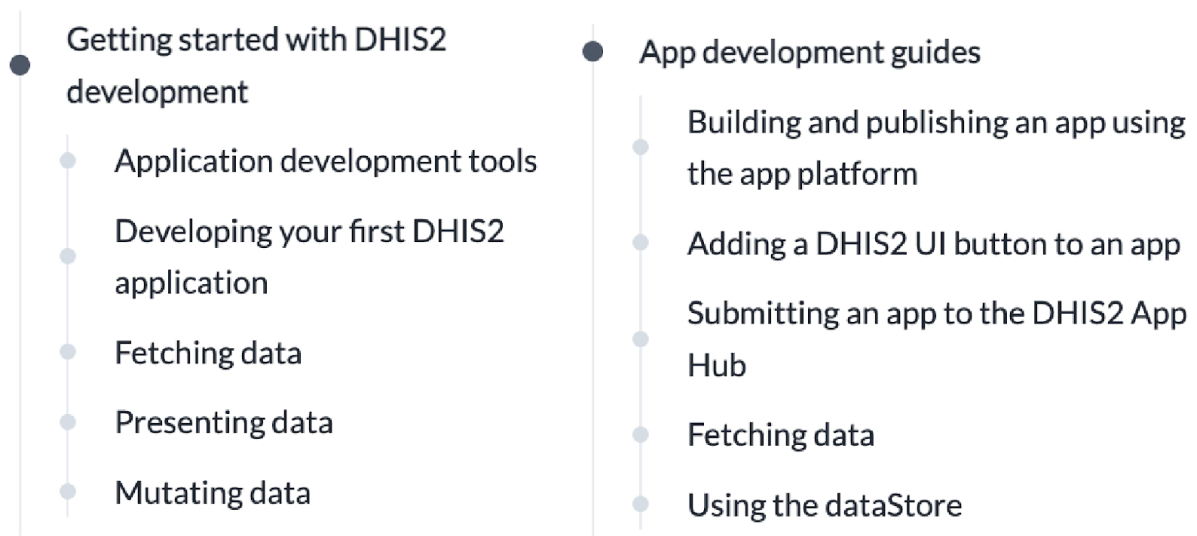


Image 5.3: Tutorial and How-to guides sections

### 5.3.1 DHIS2 Tutorial

We theorised that one of the reasons for why the DHIS2 KBRs were insufficient to onboard students was because they had not been given a thorough enough introduction to DHIS2. The DHIS2 references are highly comprehensive due to the platform’s high functional extent and are not structured in a way conducive to learning. Therefore, we designed the DHIS2 tutorial (see Image 5.4) to gradually introduce the students to DHIS2 and prepare them for working with the other KBRs during the project. Our kernel theory suggested that if the students were made aware of what DHIS2 boundary resources existed, had some practice using them, and were supplied with working code examples, they would be more prepared to develop their own applications.

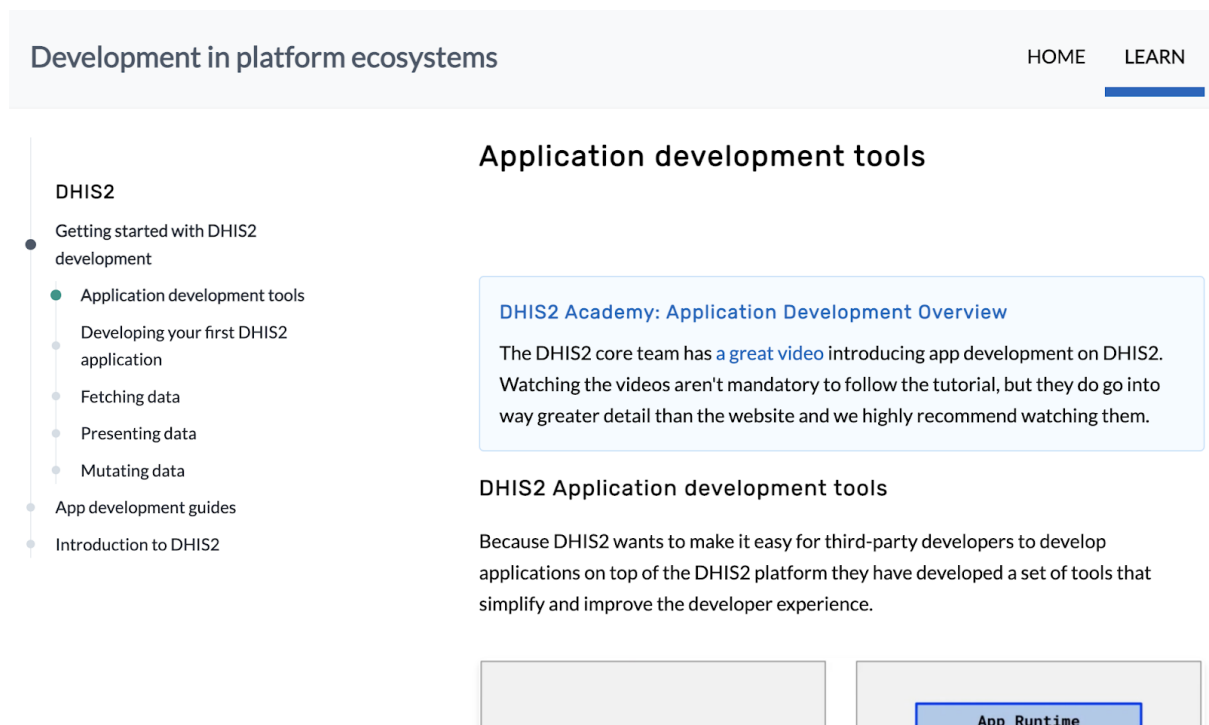


Image 5.4: The DHIS2 tutorial on DHIS2 App Course

Throughout the development of the DHIS2 tutorial, several design decisions emerged. First, we had to consider whether to link to existing DHIS2 KBRs or create content specifically for the university course. We had the option of either utilising existing DHIS2 resources by linking to them or creating new resources that were more course-specific. While we did use both of these strategies, we created many resources from scratch as we felt that the existing KBRs were difficult to use, inadequate or unspecific for the students. Further, we had to decide on the comprehensiveness of the DHIS2 tutorial. Considering that students would be expected to complete this part of the course within two weeks, the comprehensiveness of the tutorial had to be limited. We introduced students to the most important parts of the DHIS2 platform such as the DHIS2 data model, the API, Data Queries and UI components. However, we had to limit how comprehensively we covered the API and the data model in particular. Because the data model is complicated and the platform's functional extent is high, we decided to only cover the essential functionality that all students would be required to use which limited the comprehensiveness and increased the specificity of the tutorial. By identifying what functionality and endpoints the students were likely to encounter, we could supplement the DHIS2 KBRs with highly specific instructions and guidance. By designing the tutorial to exclusively cover functionality that the students most likely had to implement during their group project, we could provide them with specific implementation knowledge, working code examples and experience that would assist them with their project.

Through this tutorial, students would gain hands-on practice and build relevant knowledge about specific parts of the platform, all of which aimed to be relevant to what students were expected to achieve in their final project. The tutorial started off by providing a skeleton of a DHIS2 application that students would progressively add functionality to through the proceeding sections. The tutorial was completed once the student had a finished DHIS2 application that read data from a DHIS2 instance, presented it in a table, and let users modify data (see Image 5.5 and 5.6).

The screenshot shows the DHIS2 Tutorial interface. The top navigation bar includes the DHIS2 logo, the text "DHIS 2 Demo - Sierra Leone - oblig3", and user information "41" and "156" with a profile icon and the initials "JT". On the left, there is a sidebar with "Browse" (selected) and "Insert" buttons. The main content area displays a table with the following data:

Total population < 1 year	Total Population	Total population < 5 years	Expected pregnancies	Total population < 10 years	Population of women of child bearing age (WRA)
820	20500	3342	902		4192

Image 5.5: DHIS2 Tutorial browse component

The screenshot shows the DHIS2 Tutorial interface. The top navigation bar is identical to the previous image. The sidebar shows "Browse" and "Insert" (selected) buttons. The main content area contains a form with the following elements:

- A "Select field" label above a dropdown menu.
- The dropdown menu currently displays "Total Population".
- A "Value" label above a text input field.
- The text input field is currently empty.
- A blue "Submit" button below the text input field.

Image 5.6: DHIS2 Tutorial insert component

We also created a mandatory assignment which tasked the students with extending this application to visualise an additional DHIS2 dataset to encourage more practice. However, this assignment is not relevant for this thesis.

### 5.3.2 DHIS2 How-to-guides

As described in the preliminary study, many students reported that a frequent issue was that the DHIS2 resources document the entire platform and as a result, are very comprehensive.

Whenever students had a specific task to complete, they had to read comprehensive DHIS2 references and filter out information relevant to their specific task. Due to the large amount of information available, it was difficult for them to distinguish between relevant and irrelevant information. We reasoned that this challenge could be solved by creating more specific KBRs, related to tasks and challenges they may encounter. Guided by our kernel theory, we decided to create how-to-guides to assist the students in common tasks they would likely encounter throughout their onboarding experience. This way, students could simply follow a guide and did not have to learn DHIS2 in-depth or search through comprehensive KBRs for specific information. For this section, we want to highlight one specific guide that we found enlightening with regard to our design considerations. The Datastore how-to-guide aimed to assist students in implementing the DHIS2 Datastore in their React application by providing step-by-step instruction. The DHIS2 references cover how to use the Datastore API. Despite this, several students had difficulties implementing Datastore in their application. While there were several reasons why these reference materials were not as usable as they could be, we want to highlight two issues with them. First, the references showed how to query the Datastore using the API. In contrast, students were expected to use the Datastore through Data Queries which has a different interface design (see Image 4.1). As a result, the examples provided could not be directly copied and would have to be modified to fit the students' context making them less specific. In addition, the reference materials did not introduce, nor mention the DHIS2 application “Datastore Manager” (see section 4.1.4)

We created a how-to-guide instructing how to use the DHIS2 Datastore (see Image 5.7). This guide gave students an introduction to the Datastore and the Datastore Manager. It also supplied them with code examples of how to read and write data using the Data Queries that were specific to the course.

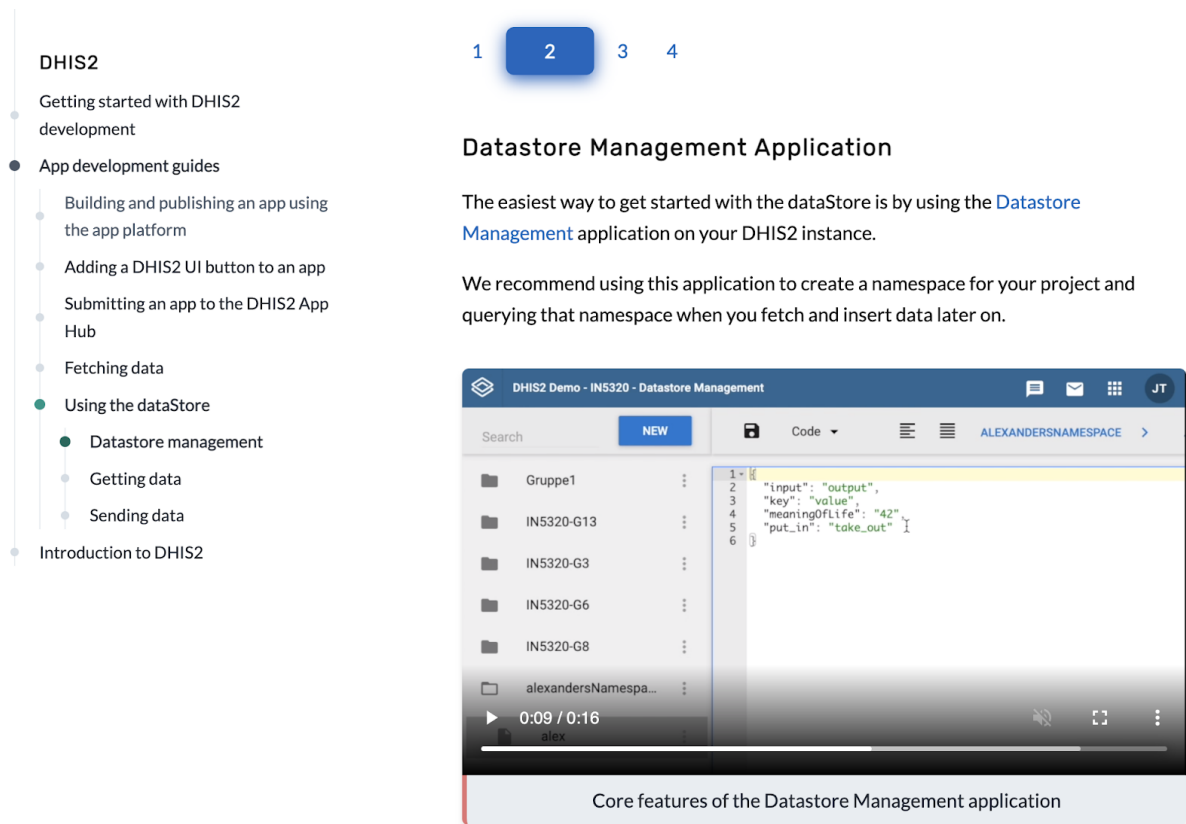


Image 5.7: Datastore how-to-guide

## 5.4 Boundary spanning activities

Another challenge that we identified in our preliminary study was that students lacked assistance when they encountered challenges where the broadcasting KBRs fell short. Students felt that they could not retrieve any help once they experienced difficulties in understanding the KBRs or when they had technical problems with their application. Therefore, we decided to introduce boundary spanning activities in the form of weekly seminars, one-on-one assistance and instant messaging. During the weekly seminars, we would live code, hold lectures about specific topics and assist students through one-and-one help. The course also had a public Mattermost channel, an instant messaging service. Through this channel, the students could discuss with each other and the seminar teachers. The students could also request assistance from us and retrieve individual guidance. Through these boundary spanning activities, we could assist the students when problems arose. The goal was to provide critical knowledge where the broadcasted KBRs turned out to be insufficient and to aid complementors with specific assistance where needed.

## 5.5 Chapter summary

Several aspects informed the design of the artefact. From the preliminary study, we identified the complicatedness of the platform, a lack of non-platform specific knowledge, insufficient DHIS2 KBRs and lack of assistance as the main challenges when onboarding students in the course. In order to address these challenges, we created a curriculum, non-platform specific KBRs, platform specific KBRs and provided boundary spanning activities. Thus, each previously discovered challenge could be addressed accordingly through the design of the artefact. Table 5.1 shows an overview of all challenges and their respective solutions.

Identified challenge	Proposed Solution	Description of proposed solution
Complicated platform	Curriculum design	Limit learning scope Gradual learning experience
Limited non-platform specific knowledge	Non-platform specific KBRs	Create explanations Give practice
Insufficient DHIS2 KBRs	DHIS2 Tutorial	Enable hands-on experience Provide explanations Link to other KBRs Limit learning scope
	DHIS2 Guides	Specific instructions Provide code examples Limit learning scope
Lack of assistance	Boundary spanning activities	One-and-one help Weekly seminars Mattermost channel

Table 5.1: Proposed solutions to challenges



## 6. Artefact Evaluation

During the onboarding process, the students utilised DHIS2 KBRs and KBRs provided by us such as the DHIS2 App course and the boundary spanning activities. The artefact evaluation is therefore not limited to just the KBRs that we created but focuses on the entire collection of KBRs used during the onboarding of students. Figure 6.1 shows the relationship between the KBRs and boundary resources which are a part of this evaluation. The green KBRs at the bottom of the figure are provided through the DHIS2 App Course.

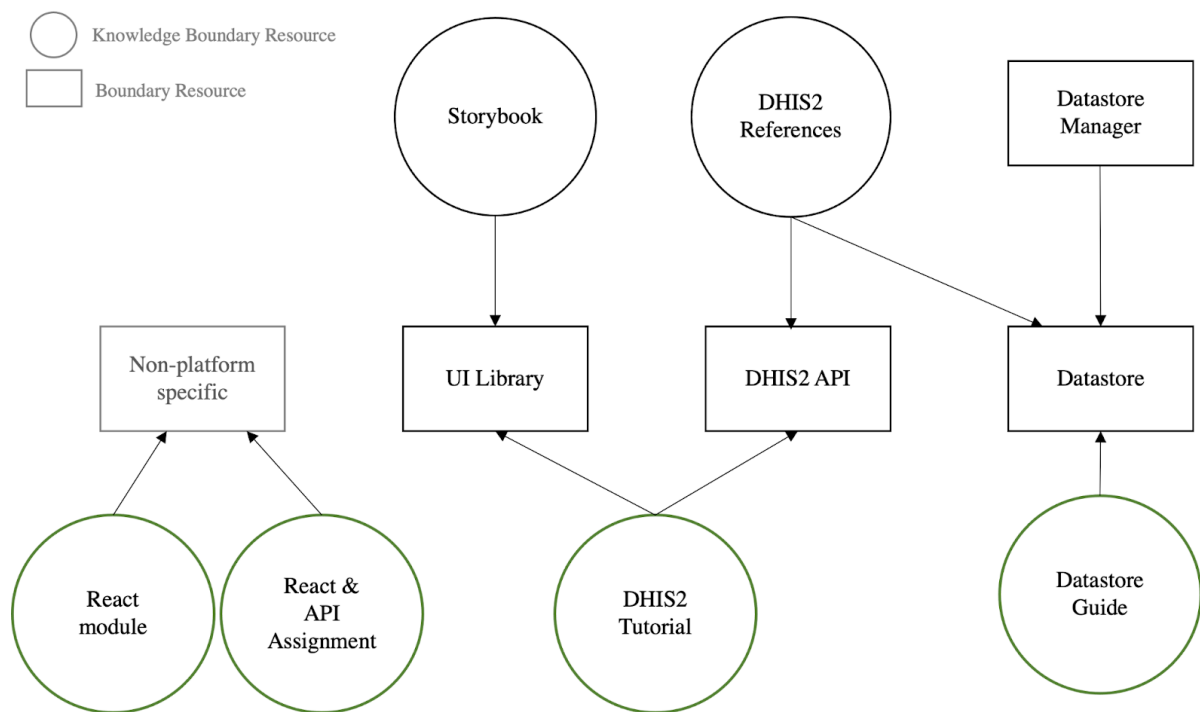


Figure 6.1: The relationship between the KBRs and boundary resources

In Chapter 4, we presented four key challenges with the existing artefact that hindered complementors' onboarding to the DHIS2 platform. Revising from the previous chapter, these challenges were:

- Limited non-platform specific knowledge
- Complicated platform
- Insufficient DHIS2 KBRs
- Lack of assistance

We addressed these by creating a curriculum for the course, non-platform specific KBRs, platform-specific KBRs and boundary spanning activities. To reiterate, we evaluate the artefact on its utility, validity and quality in solving the initial challenges. The empirical findings derive from different focus groups, observations, surveys and the expert evaluation throughout the course. This chapter is structured as follows; We first present our findings on how the artefact was used in practice throughout the course. Thereafter, we discuss how the artefact performed with respect to the initial challenges it sought to solve. Finally, we introduce interactive KBRs, an important concept for onboarding complementors that emerged throughout the artefact evaluation.

## **6.1 Students behaviour**

Because 137 students went through our onboarding, we were able to learn a lot about how they used the provided KBRs as part of their onboarding. The findings were derived from our surveys, focus groups and close relationships with the students. We found that students used many different KBRs throughout their onboarding process and that the KBRs used depended on what the students were trying to achieve at that moment. While there was a lot of variation in how students used our KBRs, we found some commonalities. Most students would primarily rely on using broadcasting KBRs, either provided by us, DHIS2, or by third parties. Image 6.1 comes from our first survey, asking students what resources they used during the first two weeks of the university course when they learned HTML, CSS, and Javascript:

**Did you use any other resources than the app-course website to learn HTML, CSS and JS? \***

If so, what?

Svar	Antall	Prosent
Google	27	93,1 %
Stack overflow	23	79,3 %
W3schools	25	86,2 %
Developer documentation	10	34,5 %
Other online courses / MOOCs	1	3,4 %
YouTube	9	31 %
Mattermost channel	2	6,9 %
Fellow students	14	48,3 %
Group seminars	6	20,7 %
Other	3	10,3 %

Image 6.1: Different KBRs used during the online course

Earlier in the same survey 89.7% of the students stated they used DHIS2 App Course to learn HTML, CSS and Javascript. We find that Google is unsurprisingly a very influential KBR for most developers which leads students to a variety of other external resources. This underscores how developers use a lot of different KBRs when learning a technology. We observed that in general, most students started their learning process by using introductory KBRs such as tutorials that provide an overview of important functionality and attributes of the software or platform. We found that there were differences in how students preferred to approach learning new technology. Some students preferred watching introductory videos while others preferred reading text as it was “more efficient to read”. However, most students stated that they learned the best when “learning by doing” or in other words, learning through practice, usually assisted by KBRs. The extracts below are from three focus groups with students which show some of the variations between students' approaches to learning:

*O1: “I’m a big fan of finding a YouTube video that covers everything and then I’ll just lean back and pay attention.”*

*O2: “I’m more of a trying out things kind of person. I’m a “knock my head against the wall” type of student and programmer.”*

*O3: “You start with the basics and some tutorials afterward, learning by doing is good.”*

Following these introductory KBRs, students would have an idea of what they are trying to achieve with the platform, and how they may implement it into their application. They then usually entered a task-focused implementation process where they tried to implement the boundary resources into their application through trial and error with assistance from KBRs. Here, they would commonly use a multitude of KBRs such as the DHIS2 references, how-to guides, or external resources found through Google. We also observed that some students, especially the more experienced ones, would skip the more introductory KBRs and go straight to attempting to implement the functionality.

*“I just tried things. I like learning in that way, and I guess most people are like that. And when I encountered something that I didn’t understand, I just Googled it. And then there would be something on Stackoverflow about it or something.”*

If a student was skilled enough in application development to use the broadcasting KBRs, the learning approach described above was usually sufficient and they could move on to new tasks. However, if this approach did not solve their task, they would resort to one of the following approaches. First, they intensified their trial and error efforts and would attempt to learn through practice and experimenting directly with the boundary resource. Second, they intensified their information foraging. For instance, they would intensify their Googling efforts or delve deeper into comprehensive KBRs and other sources of knowledge like code repositories. Third, depending on the students' tendencies to ask for assistance, and the environment around them, they would at some point resort to asking others for assistance. For example, by asking a fellow student or boundary spanner. From here they would receive more specific knowledge and guidance and would, in most cases, be able to solve their task. We note that the students' preferred approaches varied; however, most students would address an implementation challenge through a combination of these three approaches. Some students spent days solving a problem themselves rather than asking for help, while others asked for assistance frequently without exerting much effort first.

## 6.2 Curriculum design

The complicated DHIS2 system and the documentation posed another major challenge during the onboarding of new complementors to the DHIS2 ecosystem. We addressed this challenge by designing a curriculum for the course, increasing the specificity of the KBRs towards the project, and enabling a more gradual learning experience for the students. By following the curriculum, the students would first acquire important non-platform specific skills, before learning about the DHIS2 platform essentials. Later during the project, students were required to use the DHIS2 references and find specific information related to their use case on their own. Because the students enrolled in the university course vary every year, the effect of the curriculum design on the course is difficult to compare to the prior course. Various students noted, however, that they enjoyed the step-by-step introduction to DHIS2 application development and how the course was set up in general. We received particularly positive feedback on the React module and the React and API assignment.

However, there were also some enduring challenges with the curriculum design. A student described that there was a big gap between the API calls in the DHIS2 tutorial and the actual DHIS2 API, claiming that it *“sort of only gave me a refresher when it came to the API but it didn't really make me understand all of it and how it relates to the data model.”* As described by the student, our curriculum did not teach about the entire DHIS2 system, nor did it go through all possible use cases. The way the course curriculum was designed, it only addressed a small portion of the API and DHIS2 data model. Due to the time frame of the course and the complicated DHIS2 API, it was infeasible to teach the students everything about the DHIS2 platform. This led to many students feeling that they had not gained a full understanding of DHIS2 when they started developing their applications. After the project was completed we asked the students: “On a scale from one to ten, how well did you feel prepared for developing a DHIS2 application when the [group] project started?”. On average the students reported 5.9 out of 10. It appears that the curriculum only partly prepared the students for DHIS2 application development. This may be explained by the specificity of the tutorial being limited to one use case of the platform. The way the tutorial was designed, the students were not supposed to learn about the entire platform’s functional extent. When asking the students why they did not feel prepared enough for developing an application, 9 out of the 15 respondents answered it was due to the complicated DHIS2 API compounded by a general lack of understanding for how to use the API.

To conclude, the students enjoyed a more gradual curriculum and were able to use the DHIS2 App Course without any large problems. However, there were challenges once they started their application development and had to use comprehensive DHIS2 KBRs. This may be explained by how the curriculum was not comprehensive enough and they did not learn enough about DHIS2 to generalise to their own project. However because DHIS2 is highly complicated this challenge is hard to address in the limited time frame of the course.

### **6.3 Non-platform specific KBRs**

During the preliminary study, we found out that students often lacked the required prerequisite knowledge about web development and APIs which posed an obstacle to their successful onboarding. In order to create a better onboarding process for the students, the artefact included several non platform specific KBRs.

The non-platform specific KBRs include the CSS, HTML, Javascript and React sections on the App course website. Within the context of the non-platform specific KBRs, approximately 72% of respondents answered that the app-course website was either “very” or “extremely” helpful in aiding the refreshment or acquisition of programming language knowledge. In comparison, across the platform specific KBRs, 85% of respondents answered that the course website was either “very” or “extremely” helpful. Comparing these two numbers, it appears that the respondents found the platform specific resources more helpful than the non-specific resources. This could be explained by the fact that there were no alternatives available for acquiring platform specific knowledge other than the official DHIS2 KBRs and our course website. One of the respondents argued that for the non-platform specific tasks such as Javascript and React, the course website helped somehow but was not as critical due to a large number of external resources available. On the contrary, when acquiring DHIS2 knowledge, the course website was “more useful than anything”. It appears that the platform-specific KBRs were perceived as more useful on average. There are also no third-party resources available on platform-specific knowledge. The only way students could gain platform-specific knowledge is through the KBRs provided by platform owners. Not surprisingly, our findings indicate that including platform specific knowledge is more critical than including non-platform specific knowledge when onboarding newcomers to a platform.

*“I think this website needs to be really good because there aren’t really any alternatives. If you are going to learn HTML and Javascript, you can look in other places but not in the case of DHIS2.”*

However, only approximately 21% of the respondents had previous experience with CSS, HTML, Javascript and React. These students relied particularly on the provided non-platform specific KBRs or on external resources. Considering that non-platform specific knowledge was required to build a DHIS2 application, we found it important to include non-platform specific content in our KBRs to ensure that everyone was on the same skill level before introducing DHIS2 app development. We also observed that students who used third-party resources rather than our KBRs sometimes found outdated information without being aware. For example, students who googled to gain knowledge about React sometimes ended up using outdated React versions. Including non-platform specific KBRs can therefore help to foster a better programming practice and decrease the risk of complementors using outdated information by Googling.

## **6.4 Platform specific KBRs**

Another major challenge found throughout the preliminary study was that the DHIS2 KBRs were insufficient when onboarding new complementors to the DHIS2 platform. There were many reasons why students perceived the DHIS2 KBRs as insufficient. Guided by our kernel theory, we identified a missing DHIS2 introduction and missing specific resources as two main problems. To address these challenges, we complemented the DHIS2 KBRs with a tutorial, explanatory material and how-to-guides.

### 6.4.1 DHIS2 Tutorial

Before introducing our artefact, there was no introduction to learning about the DHIS2 UI library or the DHIS2 API and the students had to rely on the DHIS2 references to learn about the platform. As this caused several problems for the students, our artefact introduced a DHIS2 tutorial.

In a survey, 85% of the respondents answered that the tutorial has been “very helpful” or “extremely helpful”. Various students pointed out that the App course website has been a great guide that helped them get started with web development and DHIS2 essentials. For example, one of the respondents explained:

*“The resources provided to us in this course [website] were great for me as a ‘novice’ developer in the DHIS2 ecosystem. I felt like the resources were basic enough at the start for me to have a slow and steady progression with React/CSS/JS, as well as the DHIS2 API, before building on what I had learned later on.”*

Another respondent stated that “the step by step introduction was nicely paced to ease the students into DHIS2 development”. It has also been claimed that the tutorial on the course website was easier to understand than most of the DHIS2 KBRs. For example, according to one student, the provided code examples were one of the reasons why the tutorial was easier to grasp than the DHIS2 references. The DHIS2 references contain detailed information about the DHIS2 platform and is a purely technical description of the system. Before we introduced our tutorial, students struggled using the DHIS2 references because it contains very few examples and explanations. Consequently, it is a difficult place for learning about the platform. It appears that the students found the tutorial very useful as it provided a more gradual introduction to DHIS2. Because the tutorial provided code examples and explanations, many students found it to be easier to understand than the DHIS2 references.

Another essential idea with the tutorial was to limit the learning scope to be more specific to what the students were going to need for the project. As DHIS2 is a large platform with a high functional extent, we decided to limit the scope of the tutorial to only comprise a small part of the DHIS2 data model. The tutorial introduced only a few specific endpoints as well as a few common use cases of the DHIS2 UI library. For example, since tables were very



commonly used to display data, the tutorial illustrated the implementation of the DHIS2 UI table component. We found that students had no or few issues when following the tutorial. There were also a few issues when they reused functionality that was covered in the tutorial later on in the project. However, despite the introduction that they have been given through the tutorial, the students struggled to use the DHIS2 references later. A student described this problem:

*“Your tutorial was really good but a little bit kindergarten. It was a safe space and then when the project started you had to find out about things on your own using the DHIS2 references.”*

Although various students pointed out the importance of the tutorial to be able to use the DHIS2 references, it appears that the tutorial was too specific and did not prepare them for working with the rest of DHIS2. The tutorial was designed to provide students with basic DHIS2 knowledge rather than teaching them everything about the DHIS2 platform. The DHIS2 references are extremely comprehensive in the way that they comprise all information on all possible use cases of the DHIS2 platform. In contrast, the tutorial that we provided was more specific. As a result, students found it difficult to transfer their experiences from the tutorial over to using the DHIS2 references.

We conclude that the tutorial was very useful for getting the students started with DHIS2. By providing a gradual learning curve that introduced the terminology of DHIS2 and encouraged practice, students were better prepared for developing an application afterward. Nonetheless, there were some areas of DHIS2 that were not covered well enough and could be addressed better by the tutorial. However, many of the challenges students faced through their onboarding were more related to the DHIS2 references being difficult to use, and the platform being complicated. We reason that this challenge could not have been addressed by one single tutorial.

## 6.4.2 DHIS2 How-to-guides

In addition to a tutorial, we implemented how-to-guides that instructed students in how to complete common and potentially complicated tasks. This way, students could simply follow a guide instead of studying the DHIS2 references in detail.

During the course, several students noted difficulties implementing the Datastore in their applications and requested assistance from us as boundary spanners. We created a how-to guide to address this challenge in a way that scaled to all of the students. After publishing the how-to-guide on the App course website, we observed that the number of inquiries from students went down dramatically. Thus, the how-to-guides were useful in aiding complementors with completing specific tasks on their own. Further, we could benefit from simply referring to the Datastore guide. In other words, by creating a how-to-guide we could broker knowledge instead of bridging knowledge. As bridging is a resource-intensive activity, the how-to-guides were useful to save time and resources by brokering to existing KBRs. We want to note that creating these task-specific resources required a time and resource investment. First, we had to understand what complementors were having challenges with, then we had to identify an effective solution, and finally, we could create a KBR that broadcasted the solution in a way that worked reliably. Therefore, we only created a how-to guide when many students would face this challenge to make it worth the time investment.

We also noted instances where our task-specific resources were less effective. During the course, many students reported major challenges when working with the advanced functionality of the Data Query (see section 4.1.2). Most groups would have to implement this functionality in their application at some point. The DHIS2 KBRs lacked specific examples and information documenting how to send dynamic parameters to the DHIS2 API, which led to multiple groups spending a large amount of time completing the task. During a focus group, a student pointed out that a how-to-guide would be useful for his project group:

*“We got into a situation where it would have been nice to parameterise the Data Query to get and mutate data. [...] I think [the DHIS2 references] aren’t sufficient in telling how to use these functions and how to parameterise all the options that exist in these functions. [...] So I wish it was more explicit about what you can do. If you want to do this, do that.”*

As clearly stated in this quote, the student wanted KBRs that were more explicit about how you could use certain functionality in a specific context. It seemed like creating a how-to-guide would have been helpful in this specific situation. However, this was challenging because there was a lot of variation in how to use the Data Queries. Essentially, the implementation depended on several factors depending on the application structure and the specific endpoint they were querying. All this variation made it difficult to create one single how-to-guide because there were no one-size-fits all solution in all circumstances. As a result, we decided not to create a specific guide on how to parameterise a data query for all situations. In this circumstance, we resorted to boundary spanning activities and provided code examples that students could copy into their application and modify to their context.

To conclude, we found task-specific resources effective in assisting students with completing specific tasks without having to understand the system comprehensively. For how-to guides to be resource-efficient, they had to address tasks that many students encountered frequently. Following the creation of the guide, we could then broker knowledge rather than bridging it. We also found that creating effective guides was challenging when the underlying boundary resource had a lot of variability, resulting in the guide becoming increasingly comprehensive and less specific.

## 6.5 Boundary spanning activities

The boundary spanning activities were designed to bridge knowledge through one-and-one help, including the group seminars and the Mattermost channel. In a survey, 33% of the respondents answered that they asked boundary spanners for guidance about programming issues and understanding the DHIS2 KBRs. The importance of the boundary spanners was explained by one of the students:

*“It helped that the seminar teachers had the expertise and knew how the system worked. There are a lot of things that we may not be aware of [...]. It’s not certain that the website has all the information needed or that it is as detailed as needed. You have worked as a person between us students and those responsible [for the documentation]. We could ask you questions. That was very helpful. I feel like it was necessary, too. The website doesn’t cover it all.”*

As described by the student, the seminar teacher functioned as mediators between the platform and the students by bridging knowledge where needed. The student also highlighted the utility of boundary spanners due to their platform expertise, specifically when the broadcasted KBRs lacked information.

We found that there were two reasons why a student would experience a broadcasting KBR to be insufficient. The first reason was that the broadcasting KBR was well designed, but the specific student was not sufficiently skilled to comprehend and use it. When this occurred, boundary spanning activities played an important role as we could bridge knowledge to the students and assist them through their onboarding process. Because broadcasting KBRs are designed to scale to many complementors simultaneously, they do not always work for every complementor. The second reason was when the broadcasting KBR was itself poorly designed and difficult to comprehend. Boundary spanning was then helpful because we could bridge knowledge to the student, and afterward improve the broadcasting KBR itself. For instance, in the tutorial, we had a guide on how to set up the DHIS2 development environment. When the course period started, several students mentioned that the guide did not work on their computers. By communicating with the students, we identified the problem, found a solution, and revised the guide to communicate the solution.

Additionally, boundary spanning activities were helpful when students encountered problems during the course and needed specific assistance to solve a task they found challenging. As discussed earlier, we created a Datastore guide because students were frequently asking questions about how to implement it in their application. Without these boundary spanning activities, we would not have identified that students found that specific boundary resource challenging. Thus, boundary spanning activities were an effective method for learning about the students' needs and identifying potential KBRs.

To conclude we found boundary spanning activities for three reasons. First, we could assist students who were less skilled than their peers and needed more support than the broadcasted KBRs provided during the onboarding. Second, we could identify flaws with the broadcasted KBRs and improve them. Third, we could find new ideas for new specific KBRs which assisted the students with challenging problems.

## **6.6 Interactive broadcasted KBRs**

As we observed the artefact in use, we noted a set of KBRs which seemed to transfer knowledge more effectively than other KBRs. We refer to this type of KBR as interactive broadcasted KBRs. We conceptualise them as a type of broadcasted KBR which represents or interacts directly with the platform's boundary resources. By providing an interface built on top of the boundary resource they provide an environment where a complementor can learn or use a boundary resource more effectively than compared to a traditional KBR like a reference or training video. Throughout the project, we identified four interactive broadcasted KBRs; Interactive code exercises, Data Query Playground, Datastore Manager and Storybook. We found that they were useful for two reasons. First, they were effective at aiding the students with attaining hands-on experience and skills. Second, they were effective in assisting students with achieving specific tasks.

The App course website contained many embedded interactive code exercises in the HTML, CSS and Javascript sections. Several students noted that they found them useful for learning these technologies and noted their absence in the later modules in the course. For example, they stated that the instant feedback provided through the interactive code exercises was useful. It informed them if they had understood the concepts or if they had to spend more time learning the materials before advancing to the next topic.

Similarly, the Data Query Playground (see section 4.2.4) was noted as helpful by many students. One student explained during a focus group:

*“It was the opportunity to test things without having to do a lot of work or risk a project. That you could just jump right into it. You could check if this worked, or if that worked. The freedom of how easy it was. It made it so that I didn’t need to be afraid, right? I didn’t have to read a lot to make sure that it works. You just plot it in and if it doesn’t work, who cares, it took 10 seconds. [...] It reduced the barrier to entry, there’s less of a mental block around just trying something”*

This quote captures the benefits of interactive KBRs compared to other broadcasting KBRs such as a reference. Because they are browser-based, students could test their code quickly without having to set up a new project. In other words, it reduced the barrier to entry. Furthermore, they provided an environment where the students could experiment and iterate to a working solution with instant feedback. Additionally, there was no risk to the experimentation. They did not have to modify an existing project, further encouraging them to experiment. All of these attributes are highly conducive to encouraging learning and practical experience.

While the Datastore Manager (see section 4.1.4) is more of a boundary resource, it functioned like an interactive KBR because it was used to gain knowledge, not just use it as a boundary resource. We observed that students found it challenging to use the Datastore through the API, however, the students did not find the Datastore Manager application challenging to use. We reasoned that this was because the Datastore Manager application provided the students with an interface design that they were familiar with from other web

and mobile applications. By using the graphical user interface, the students could learn how the underlying Datastore API worked. After they had experimented with the Datastore Manager, implementing the Datastore API in their application was easier because they had already learned its functionality. In essence, this interactive KBR had the same aforementioned benefits as the other interactive KBRs. However, in addition, it provided a more intuitive interface design and user experience than the underlying boundary resource itself, leading to a better learning experience.

The final interactive KBR in the course was Storybook (see section 4.2.3). We observed that Storybook similarly provided an intuitive, low friction, low-risk, and short feedback loop environment. However, Storybook additionally assisted students with completing common tasks they would encounter when developing an application. Storybook provided a better interface for browsing and discovering UI components compared to a static webpage displaying UI components. After discovering a relevant UI component, students could change its parameters and instantly see how the component changed. After the component had been modified, the student could simply copy the code directly into their application. We observed that Storybook was highly effective and we rarely had to assist students with UI-related matters. In our final survey, 83% of students noted that Storybook was “very helpful” or “extremely helpful” which underscores the effectiveness of this KBR. We note that DHIS2 provided a textual reference of UI components, however, few students used it because Storybook provided a superior experience.

The expert evaluation also highlighted the usefulness of interactive KBRs. When we asked the expert why he thinks that interactivity is such a useful tool, he explained:

*“Because it changes the learning experience from having a lecture towards having a hands-on experience [...] One thing is when you are being thrown something in the face and it's unidirectional. When you have these learning experiences be bidirectional, basically you're learning and you're applying at the same time [...] In general, the more interactive the better because people feel they are in control. When they can try things and they know that they nailed it right?”*

Although creating interactive KBRs may be resource-intensive, interactive experiences give a “*great return on investment*”, the expert added.

To conclude, interactive broadcasted KBRs provided an effective learning and onboarding experience because it let the students learn the boundary resources by using them directly in an interactive environment. Their functions vary, but in general, they provided an intuitive, low friction, low-risk and short feedback loop environment which assisted them with learning or completing tasks on the platform.

## **7. Design considerations**

This chapter presents five design considerations that software platform owners can use as guidance when designing KBRs for onboarding new complementors to their platform ecosystem. These design considerations emerged gradually throughout the entire design and evaluation process of our artefact. Empirically, they are based on the findings from evaluating the artefact in use when over 137 students used it as part of their onboarding process. In total, we define and discuss five design considerations in Table 7.1.

For each design consideration, we first give an overview of what the design consideration entails, followed by providing empirical support from our artefact evaluation. Thereafter, we discuss how platform owners may realise the design consideration and in which situations they can apply the design consideration. Finally, we also discuss trade-offs if any and potential challenges that platform owners may encounter while following these design considerations. While platform owners do not have to follow these design considerations strictly, we suggest that they can use them as guidance for designing their KBRs for onboarding complementors to their platform.



Design Consideration	Description
Designing KBRs for comprehensiveness and specificity	Platform owners face a tradeoff between providing KBRs for comprehensiveness and specificity. While specific KBRs can be easier to follow than comprehensive KBRs for new complementors, comprehensive KBRs are essential because they describe more of the platform's functional extent. Effective onboarding should therefore both include specific and comprehensive KBRs.
Broadcasting tutorials, guides, references and explanations	Platform owners should consider broadcasting references, tutorials, how-to guides and explanations to improve their KBRs. Complementors rely on all four types of KBRs and use them at different times throughout their onboarding process. Due to their scalability, platform owners can effectively onboard many complementors simultaneously.
Performing boundary spanning activities	Platform owners should consider performing boundary spanning activities to assist complementors through their onboarding process where broadcasting KBRs fail to anticipate the complementors needs. Platform owners should also use the learnings from the boundary spanning to improve the broadcasting KBRs for future onboarding of complementors.
Provisioning interactive broadcasting KBRs	Platform owners should consider the provisioning of interactive broadcasting KBRs to aid complementors with a more intuitive, low friction, low risk and instant feedback environment. Such KBRs can also improve the onboarding process by simplifying tasks that complementors encounter, reducing the knowledge required to complete those tasks.
Providing non-platform specific knowledge	Platform owners can consider providing non-platform specific knowledge by linking to external resources or creating custom content. This can improve the onboarding process for less experienced complementors and the platform owners do not have to rely on the quality and suitability of external resources.

Table 7.1: Design considerations

## 7.1 Designing KBRs for comprehensiveness and specificity

By designing KBRs for onboarding complementors to a platform, we found that platform owners face a tradeoff between providing KBRs for comprehensiveness and specificity. While the students in our study found specific KBRs easier to follow than comprehensive KBRs, comprehensive KBRs are essential because they describe more of the platform's functional extent. Effective onboarding should therefore both include specific and comprehensive KBRs.

Comprehensive KBRs are oriented towards the platform's boundary resources. They cover a platform's interface design and functional extent to a high degree. For example, references are highly comprehensive KBRs. Throughout our artefact evaluation, we found that if KBRs did not cover a platform's entire functional extent, the onboarding process was disrupted. To retrieve information about the underlying boundary resource, the students then had to either rely on boundary spanners or trial and error in order to proceed with their onboarding. Additionally, we observed that students found comprehensive KBRs more challenging to use because it requires more effort to learn from. Due to the large amount of information provided, students were often overwhelmed, spent more time learning unnecessary details, or found it hard to identify relevant information required for accomplishing their task.

Therefore, a platform owner should also provide specific KBRs which are more oriented toward the tasks a complementor is trying to achieve with the platform. Specific KBRs aim to reduce the amount of extraneous information through for example step-by-step instructions. We observed that students found specific KBRs easier to use than comprehensive KBRs. They could rapidly accomplish specific tasks, without the need to understand the underlying logic of the task they were attempting to achieve. However, specific resources had drawbacks as well. First, students who were provided highly specific solutions to their tasks did not learn the materials as well, which harmed their ability to generalise to other tasks. Secondly, creating many specific KBRs that covered all the tasks the students may face would require a substantial amount of resources.

Creating an effective onboarding experience requires therefore the provisioning of both specific and comprehensive KBRs. A platform owner should design the sum of KBRs to describe the entire functional extent of the platform. This can be achieved through, for

instance, provisioning a reference, which provides a baseline of comprehensive knowledge. The platform owner should additionally research the most common tasks complementors face and provision specific KBRs to assist in those tasks. Examples of tasks where a highly specific KBR, such as a how-to guide, might be applicable, are tasks that complementors will only need to accomplish once, rarely vary or that the complementors will not need to generalise from at a later point. However, certain tasks that complementors face require more comprehensive resources than just a specific how-to guide. For instance, we found that when the underlying boundary resource had a lot of variability in usage creating a very specific guide was difficult. In these cases, creating a more comprehensive, but still specific, tutorial or explanation can be beneficial. For example, the DHIS2 app course tutorial tries to be comprehensive enough to introduce complementors to application development and allow for generalisation, while still being specific enough towards just application development. Essentially, a platform owner must provision comprehensive and specific KBRs, while striking the right balance between the two.

## 7.2 Broadcasting tutorials, guides, references and explanations

We find that platform owners should consider broadcasting references, tutorials, how-to guides and explanations. Broadcasting KBRs are highly standardised KBRs with high scale, i.e. they address many complementors simultaneously at a low marginal cost. Because platforms often contain many geographically dispersed complementors, broadcasted KBRs are effective for onboarding complementors without the presence of boundary spanners. Diátaxis, which we relied on as part of our kernel theory, argues that technical documentation should not consist of merely referential material but also include tutorials, how-to-guides and explanatory material.

We observed that the DHIS2 references were an essential KBR for application development. They provided detailed information of the entire platform's functional extent and were highly used when implementing the platform boundary resources. Originally, the DHIS2 KBRs consisted primarily of reference materials. However, most students reported difficulties learning about platform-specifics with the DHIS2 references alone. Due to the large amounts of information contained by references, students found it difficult to find specific information relevant for their task. Further, students experienced that references were difficult to use during their onboarding due to unfamiliar technical terminology. Therefore, we found only provisioning references to be insufficient for the onboarding of complementors. For this reason, we implemented a tutorial to introduce them to the DHIS2 platform. The tutorial gave the students hands-on experience with the platform's boundary resources while it explained essential platform-specific knowledge and terminology. In contrast to the DHIS2 references, the tutorial provided the students with a gradual and more specific introduction to application development. Our findings from observing the students while learning new technologies substantiate the need for a tutorial as most students would first use more introductory materials before attempting to implement an application. Furthermore, most students stated that they learn best through practice and learning by doing, underscoring the necessity for a practical introduction. Several students noted that the tutorial helped them to learn about the DHIS2 platform and was easier to understand than other DHIS2 KBRs. Thus, the tutorial turned out to be highly useful for onboarding complementors.

We also observed that a lack of explanations posed an obstacle for onboarding complementors to the platform. By integrating explanations into our artefact, we could

provide the students with a better overview and greater understanding of the DHIS2 platform. Students required, for instance, explanations about the DHIS2 data model and API to make sense of the DHIS2 boundary resources. As new complementors to the platform may not have any previous experience with the platform, explanatory material is important to aid their understanding of the platform, especially with regard to more complicated or complex knowledge. Finally, we found that the DHIS2 KBRs could be improved by adding how-to guides. These step-by-step guides instructed the students in their tasks and were highly effective at assisting students. By providing how-to-guides specific to a task, more students were able to complete more complicated tasks on their own with less effort. Thus, how-to guides are useful for the onboarding process because they reduce how much knowledge complementors need to learn, effectivising how quickly they can get started with developing applications.

We saw that complementors who are being onboarded to a platform relied heavily on these four types of KBRs. Tutorials, references, guides and explanations are used at different times throughout the onboarding process. A good tutorial furnishes complementors with introductory knowledge of the platforms boundary resources and KBRs. References and how-to guides are used extensively when a complementor is developing an application. Explanations are necessary to provide the complementors with understanding about more complicated parts of the platform. Platform owners should consider broadcasting tutorials, references, how-to-guides and explanations due to the important role they play in onboarding new complementors to a platform. Due to their scalability, platform owners can effectively onboard many complementors simultaneously.

### **7.3 Performing boundary spanning activities**

Platform owners should consider performing boundary spanning activities to assist complementors throughout their onboarding process where broadcasting KBRs fail to anticipate the complementors needs. However, they have a high human resource cost. Therefore, these boundary spanners should additionally aim to improve the onboarding process by improving the broadcasting KBRs to better address complementors at scale.

Throughout the onboarding process, we discovered that providing boundary spanning activities was highly useful for three reasons. First, the students sometimes lack the required skills to make use of the platform's KBRs. Since broadcasting KBRs are standardised, they tend to not adapt as well to the individual complementors needs and context. The boundary spanning activities, in contrast, allowed more personalised assistance when the broadcasting KBRs failed to anticipate the complementors' needs. By bridging or brokering knowledge when complementors encounter challenges during their onboarding, platform owners can assist complementors in progressing through the onboarding with less frustration. Second, we found that boundary spanning activities were useful to learn more about common tasks that complementors found challenging. By having a personal relationship with the students throughout the weekly seminars, we could identify common challenges and tasks that many complementors requested assistance with. We could then create broadcasting KBRs such as how-to guides that assisted the students with those specific tasks. Thereafter, we could broker the students to those broadcasted resources instead of bridging, sparing platform resources. Finally, broadcasted KBRs are rarely flawless. We experienced that creating KBRs was challenging, especially without feedback from the complementors. For example, the how-to-guide for setting up a DHIS2 application did not work reliably for all students in all contexts. After being informed of this problem through our boundary spanning activities we could improve the how-to-guide. Thus, boundary spanning activities serve as an important feedback mechanism for improving broadcasting KBRs.

Platform owners should consider providing boundary spanning activities through e.g. one-on-one assistance, instant messaging, workshops or seminars. Boundary spanning activities are highly effective in assisting complementors when broadcasting KBRs fail to onboard complementors. However, because boundary spanning activities do not scale well to all platform complementors they are hard to rely on as the only KBR to onboard new

complementors. Boundary spanners should therefore also simultaneously improve the broadcasting KBRs to better adapt to the complementors needs in the future by identifying common tasks and challenges.

## 7.4 Provisioning interactive broadcasting KBRs

Platform owners may consider provisioning interactive broadcasted KBRs to assist the complementor in their onboarding process. Interactive broadcasted KBRs are a type of broadcasted KBRs which represents or interacts directly with the platform's boundary resources. By providing an interface built on top of the boundary resource they provide an environment where a complementor can use or learn a boundary resource more effectively than compared to a traditional KBR like a reference, training video or.

Our findings show that interactive KBRs were useful for helping students acquire knowledge and skills for several reasons. First, by reducing the barrier to entry with a browser-based environment, students could put their knowledge to practice with less setup and friction compared to using the boundary resource directly. Second, by delivering instant feedback, the students could experiment quickly and validate their solutions more rapidly. Finally, by providing a low-risk environment, students did not have to be concerned about breaking their application and could experiment without risk. All of these attributes combined made interactive KBRs conducive to learning about the platform during the onboarding process. Additionally, we note that an interactive KBR can provide a more intuitive and usable interface than the boundary resource itself which can simplify the usage and learning of it. For instance, the Datastore Manager provided a more intuitive interface through which the student could learn the underlying boundary resource. This reduced the knowledge required to use the boundary resource and let the students experiment with the boundary resource to build understanding which was transferrable to using the boundary resource later on. We also found interactive broadcasting KBRs useful for streamlining specific tasks complementors encounter. For instance, Storybook assisted the complementor with the discovery and implementation of UI components. By effectivising a common task through an interactive interface, students could develop their application more effectively and with less prerequisite knowledge than with conventional KBRs.

We find interactive broadcasted KBRs to be useful with regards to the onboarding process by letting complementors gain knowledge in a more intuitive, low friction, low risk and instant feedback environment. Additionally, our findings show that they can improve the onboarding process by simplifying tasks that complementors encounter, reducing the knowledge required to complete those tasks. Because these interactive broadcasted KBRs scale well, they can



have a large impact on the entire platform ecosystem. Platform owners may therefore consider provisioning interactive broadcasted KBRs to assist complementors with their onboarding to the platform. For instance, platform owners can provide interactive code exercises, hosted developer sandboxes, or developer tools that assist with challenging tasks on the platform.

## 7.5 Providing non-platform specific knowledge

Application development in a platform context often requires competencies in addition to platform specific knowledge. We refer to such prerequisite knowledge that does not relate to the platform directly as non-platform specific knowledge. For example, complementors require web development skills and experience with APIs to develop an application on DHIS2. Because non-platform specifics often do not relate directly to the platform itself, the platform owners are not necessarily responsible for providing respective KBRs. More commonly, complementors rely on third-party resources to acquire these competencies. However, we found that providing non-platform specific knowledge can be beneficial for both the complementors and the platform owner of a platform ecosystem.

We observed that if non-platform specific knowledge is a prerequisite for using the platform boundary resources, a lack of non-platform specific KBRs can pose obstacles to the complementors' onboarding to the platform. In the case of DHIS2, the consequences of students not having enough knowledge about React caused problems for DHIS2 application development. If a platform owner decides not to provide non-platform specific knowledge, complementors will have to rely on third-party resources. However, this will only be effective if the existing resources are of high quality and the complementors are able to find them. Furthermore, third-party resources may not be particularly specific towards the tasks that the complementors are trying to achieve with the platform and they may need to filter through more unnecessary information than required for the task at hand. We also found that the absence of non-platform specific knowledge sometimes leads to bad practices. For instance, when students googled rather than using the provided KBRs, they occasionally found wrong or outdated information. In other cases, third-party resources would at times propose ineffective solutions to their problems.

To address this challenge, platform owners can link to existing third-party resources or integrate non-platform specific knowledge in the platform's KBRs. Linking to high-quality third-party resources is a cheap and effective way for platform owners to provide non-platform specific knowledge which can assist complementors during their onboarding. For example, as there were existing high-quality learning materials available on React, we decided that linking to those would be a better time and resource investment. However, sometimes there are no third-party resources that are specific enough with regards to required

non-platform specific knowledge. In those situations, it can be a better idea to integrate non-platform specific knowledge into the platform's KBRs. This has the advantage of being highly specific to what non-platform knowledge the complementor is required to have. Further, platform owners do not have to rely on the correctness and suitability of third-party resources.

We conclude that platform owners should consider providing non-platform specific knowledge by either linking to relevant resources or integrating it into their KBRs. Platform owners can thus provide complementors with essential non-platform knowledge and do not have to assume that complementors have certain competencies. Platform owners can then provide more tailored non-platform specific knowledge, improving the onboarding process for less experienced complementors.

## 8. Contributions and discussion

This thesis aimed to address the research question: *How can KBRs be designed to onboard complementors in a software platform ecosystem?* We have explored this research question through the design, development and evaluation of the DHIS2 App Course and other DHIS2 KBRs. In this chapter, we discuss our contributions to practice and research in the light of existing research. Following this, we present some limitations and future research avenues.

### 8.1 Contributions to practice

The contribution to practice is twofold. First, we contribute to practice by providing five design considerations that aim to guide software platform owners when designing their KBRs. The study involved the development of a comprehensive online course for onboarding complementors to the DHIS2 platform ecosystem. From the design, development and evaluation of the DHIS2 App course and DHIS2 KBRs, we identified five design considerations; 1) Designing KBRs for comprehensiveness and specificity 2) Broadcasting tutorials, guides, references and explanations, 3) Performing boundary spanning activities, 4) Provisioning interactive broadcasted KBRs and 5) Providing non-platform specific knowledge. While we argue that our design considerations go beyond the context of DHIS2 and can be used by other software platform owners to design their platform's KBRs to onboard complementors, the design considerations are not meant to be followed blindly. Instead, they provide prescriptive knowledge about factors that platform owners should consider when designing these KBRs and should be considered in relation to the respective platform and its complementors. For instance, providing non-platform specific knowledge is more impactful for platforms with less experienced developers.

Second, we contribute to the DHIS2 platform by improving the DHIS2 platform's KBRs, benefitting the platform ecosystem as a whole. The DHIS2 core team has expressed their interest in using the DHIS2 App course further to onboard new complementors to their platform.

## **8.2 Contributions to research**

The thesis contributes to research on platform ecosystems (Tiwana, 2013; Schreieck et al., 2016) and particularly how platform owners interact with complementors (Ghazawneh & Henfridsson, 2013; Bianco et al., 2014; Engert et al., 2022). We extend research on KBRs (Foerderer et al., 2019) by introducing the concepts of specificity, comprehensiveness and interactive broadcasting KBRs. We also present boundary spanning activities as a mechanism for improving broadcasting KBRs.

### **8.2.1 Comprehensiveness and specificity**

We extend existing research on knowledge boundary resources by contributing with the concepts of comprehensiveness and specificity. By applying the theoretical concept of scope (Foerderer et al., 2019) in an onboarding context, we identified a weakness of it not being prescriptively useful. By distilling parts of the prescriptive knowledge provided by our kernel theory Diátaxis in the related field of technical documentation authoring, we contribute with the concepts of comprehensiveness and specificity which have greater descriptive and prescriptive potential than the concept of scope alone. Comprehensive KBRs are oriented towards the platform's boundary resources, while specific KBRs are oriented towards the tasks complementors perform on the platform. While Foerderer et al. (2019) allude to the importance of these concepts, we extend the literature on knowledge boundary resources by defining these two attributes of KBRs for onboarding concretely. This is congruent with other research arguing that development-related platform resources should be designed towards the developers and not just exclusively mirror the underlying platform architecture (Bianco et al., 2014). Furthermore, we identify a tension between specificity and comprehensiveness that platform owners must address when creating KBRs for onboarding. This tension lies between providing a KBR which covers more of the platform's functional extent, thus requiring more effort from the complementor to learn from or utilise, compared to providing less extraneous knowledge to the detriment of the generalisability of the knowledge to other tasks.

While the concepts of comprehensiveness and specificity evolved from the context of application development, we believe that they could be applicable to other KBRs as well. Engert et al. (2022) propose a framework for complementor engagement with platform boundary resources (PBIs). Because the “troubleshooting” and “technical integration” complementor engagement types are similar to our research, we argue that the concepts of

specificity and comprehensiveness are applicable. However, we believe these concepts could be applicable to other types of complementor engagement as well. For instance, the “legal compliance” complementor engagement type revolves around ensuring that the platform is compliant with the relevant laws and regulations a complementor must comply with. The platform could provide a comprehensive KBR providing all regulatory details, however for some complementors providing a specific KBR informing them that they are, for instance, GDPR or CCPA compliant is sufficient. By applying specificity and comprehensiveness to the “Differentiation” complementor engagement type other insights can be deduced. The differentiation engagement type refers to how the complementor engages with the platform owner to competitively position themselves in the ecosystem. A platform owner could, for instance, provide specific knowledge to an individual complementor to encourage the complementor to position their application in a way that is mutually beneficial. Alternatively, they could provide the complementor with a more comprehensive KBR containing competitive data, usage statistics and other meaningful data through, for example, a dashboard. This more comprehensive KBR could inspire a more varied set of value-creation actions, however, it relies on the complementors ability to interpret and analyse the data itself. To conclude, we contribute to theory by introducing the concepts of comprehensiveness and specificity and the tension that lies between them in development-related onboarding KBRs.

### **8.2.2 Boundary spanning as mechanism for improving broadcasting KBRs**

Our findings showed that broadcasting KBRs are highly effective in onboarding complementors to a platform due to the high scale that these KBRs provide. Because platform ecosystems often include large networks of geographically dispersed complementors (Foerderer et al., 2019), the standardisation of platform resources is important for the scalability of the ecosystem (Hein et al., 2019). We found that KBRs with high scale are important during an onboarding process of a complementors because they equip the complementors with information related to the platform's functional extent. However, we have also seen that new complementors often struggle using highly standardised KBRs such as references. Broadcasting KBRs have not always worked for all complementors during an onboarding process due to differences in knowledge of terminology and technical skills between the complementors. In addition, we found that the heterogeneity of the complementors with respect to their varying skills and competencies made the provisioning of broadcasting KBRs alone insufficient for onboarding complementors. We suggested therefore that platform owners also should provision boundary spanning activities for onboarding complementors. While we found broadcasting KBRs important for the effective onboarding of many complementors simultaneously, boundary spanning activities were crucial for assisting complementors where the broadcasting KBRs failed to anticipate the complementors' needs. Similarly, Engert et al. (2022) and Hein et al. (2019) differentiated between highly standardised one-to-many platform resources and platform resources dedicated to supporting the individual complementors. For instance, Engert et al. (2022) distinguished between uniform boundary resources and individual boundary resources that are both frequently utilised by complementors and affect their engagement in application development. Platform owners can thus effectively scale their platform resources while at the same time answering the needs of individuals and fostering innovation (Engert et al., 2022). Throughout our study, we found, however, that provisioning platform resources to address the individual complementors also can help platform owners improve their standardised platform resources. Besides assisting complementors, the boundary spanning activities were crucial for discovering insufficiencies with existing broadcasting KBRs and allowed us to continuously improve the platform's KBRs. Essentially, boundary spanning activities serve as a mechanism for improving a platform's broadcasting KBRs in order to better address complementors at scale.

### **8.2.3 Interactive broadcasting KBRs**

Finally, we extend existing research on knowledge boundary resources with the concept of interactive broadcasting KBRs. Foerderer et al. (2019) conceptualised a platform as consisting of a set of boundary resources and a set of knowledge boundary resources. The conceptualisation further divides a platform's KBRs into three different types; broadcasting, bridgering and brokering with the goal to overcome knowledge boundaries between complementors and platform owners. Due to their high scalability, broadcasting KBRs can address many complementors in a platform ecosystem (Foerderer et al., 2019) Throughout our study, we identified a subset of broadcasting KBRs which challenge the distinction between a KBR and a boundary resource by Foerderer et al. (2019). Interactive KBRs represent or interact directly with the platform's boundary resources. By providing an interface built on top of the boundary resource they provide an environment where a complementor can use or learn a boundary resource more effectively compared to a traditional KBR such as a reference or a training video. Because an interactive KBR can function as a boundary resource and knowledge boundary resource simultaneously, it blurs the line between a BR and a KBR. In Bianco et al. (2014) conceptual framework interactive KBRs would be classified as a "development boundary resource". They emphasise how the boundary resources themselves can transfer knowledge to complementors through, for example, function names, error messages and the source code. This is congruent with our view on their effectiveness as they provide a complementor with an arena where they can learn the boundary resource through use. We extend on this research by identifying traits that make interactive KBRs highly effective mediums for learning. These are a more intuitive, low friction, low-risk, and short feedback loop environment. To conclude, we contribute with the conceptualisation of interactive broadcasting KBRs and their effectiveness in onboarding complementors to software platform ecosystems.



### 8.3 Limitations

The first limitation we want to discuss is the applicability of our design considerations to other software platforms. In our thesis, we studied the enterprise software platform DHIS2. Further, many of the concepts used to discuss and reflect upon our findings, are based on the research provided by Foerderer et al. (2019) which also focuses on enterprise software platforms. However, in our thesis we have intentionally decided to address software platform owners more broadly. We argue that our findings are applicable to both consumer and enterprise software platforms. However, we do believe that our considerations are dependent on the functional extent of a platform. Less complicated platforms with limited functional extent would likely require less knowledge transfer to complementors. Providing extensive non-platform specific knowledge or boundary spanning activities may then be excessive. However, they may still be worthwhile considerations. Additionally, many of our design considerations revolve around broadcasting KBRs, fostering loose coupling between complementors and platform owners. Our design considerations may therefore be less useful for platform owners who maintain close relationships with a select few complementors through high-touch partner relationships.

Because we were employed as seminar teachers in the university course, we knew many participants personally and gained a closer relationship with them throughout the semester. While this allowed us to capture the students' behaviour, our personal relationship with the students may also have influenced the data collected. Early on, we told students about the goal of our thesis and data collection. The students were also aware that we were the ones responsible for the KBRs in the course. Therefore, we had to consider the possibility for social desirability bias in our study. We tried to eliminate any bias by highlighting at all times that feedback of any kind is appreciated and would only help to improve the artefact in the future. For example, we started all interviews by stating that there is no need to withhold any negative feedback towards the artefact or the university course as a whole

While we do argue that our findings are generalisable to the rest of the DHIS2 ecosystem and software platforms in general, there are some key differences between the students in the course and complementors in other software platforms. First, the students who went through the university course were more homogenous than complementors in other platform contexts. They had similar backgrounds and skills due to most of them having a bachelor's degree in

informatics. This simplified KBR design as it was easier to tailor it to their skill level. Second, due to the university context, most students who entered the course had little or no relevant non-platform specific knowledge. This led us to put more effort into providing more prerequisite and explanatory knowledge which is less common on other platforms. Finally, our students had different incentives than traditional complementors. Whereas most complementors in other ecosystems are incentivised out of, for example, economic or intrinsic reasons, students in our course were incentivised to pass the course. Because the students did not have any inherent purpose in using the platform, we had to direct them by creating artificial tasks through assignments and projects. The resulting homogeneity in tasks simplified the creation of specific resources as we knew ahead of time what tasks the students were likely to encounter. However, despite these differences, we believe the findings are generalisable to other software platform ecosystems. Importantly, we argue none of these limitations invalidate our findings and we have taken measures to avoid generalising from aspects of the course context which do not reflect other software platform ecosystems.

## 8.4 Future research

To conclude the research, we will give some recommendations for future research. First, we would suggest more research into KBRs in platform ecosystems in general. Knowledge transfer is a useful lens for conceptualising a platform ecosystem and it plays an important role in the governance of them. We purposefully chose onboarding as the focus of our research because it fit our research context well. However, onboarding is just a single step in the application development process. Before a complementor enters the onboarding process they have to evaluate the platform and make the decision that they want to join the ecosystem. Furthermore, after they have developed an application they need updated knowledge about changes to the platform that is relevant to them. We suggest further research on how KBRs should be designed in platform ecosystems, for instance by researching how they play a role in the acquisition and retention of complementors.

We introduce the concepts of comprehensiveness and specificity as two attributes which influence the effectiveness of a KBR at transferring knowledge. We suggest that further avenues of research can explore their applicability to KBRs outside of an onboarding and application development context. For example, a relevant avenue for further research could be how specific and comprehensive KBRs are used for governance-related KBRs.

Finally, we suggest more research be done in interactive broadcasting KBRs. Because we did not enter into this research project with the intention of researching them and they emerged through the evaluation, we have limited knowledge about their mechanisms and functions. Due to their efficiency in transferring knowledge while at the same time reducing knowledge required to complete a task, they appear like a promising avenue for research with regards to improving the ability of complementors to contribute on a platform.

## 9. Conclusion

Through a 1,5 year-long engaged design science research study conducted in collaboration with the platform owner of the DHIS2 platform, we explored how KBRs can be designed for onboarding complementors to a software platform. Informed by the practitioner's framework Diátaxis for structuring technical documentation, we designed and developed the comprehensive online course “DHIS2 App Course” which aimed to bring a complementor with no experience in web development to being able to build an application on DHIS2. We introduced the artefact to the university course “Development in Platform ecosystem” and evaluated how it and other DHIS2 KBRs onboarded 137 students to application development on DHIS2. Through the design, development and evaluation of the course and other DHIS2 KBRs, we identify five design considerations; 1) Designing KBRs for comprehensiveness and specificity 2) Broadcasting tutorials, guides, references and explanations, 3) Performing boundary spanning activities, 4) Provisioning interactive broadcasted KBRs and 5) Providing non-platform specific knowledge. These design considerations contribute to practise by guiding software platform owners in the design of KBRs to onboard complementors. Additionally, we contribute to the DHIS2 platform by improving the platform’s KBRs, benefitting the platform ecosystem as a whole. We contribute to academic research by extending current knowledge on KBRs. Concretely, we identify boundary spanning as a mechanism for improving KBRs and present the concepts of comprehensiveness, specificity and interactive broadcasting KBRs. We also outline a set of avenues for further research.

## 10. References

- Baldwin, C.Y. & Woodard, C.J. (2008). *The Architecture of Platforms: A Unified View*. SSRN ElectronicJournal. [https://www.researchgate.net/publication/228207063\\_The\\_Architecture\\_of\\_Platforms\\_A\\_Unified\\_View](https://www.researchgate.net/publication/228207063_The_Architecture_of_Platforms_A_Unified_View)
- Basques, K. (2021, February 2). *A Framework for Writing Better Documentation*. YCombinator. <https://news.ycombinator.com/item?id=26004300>
- Bianco, V., Myllärniemi, V., Komssi, M., & Raatikainen, M. (2014, April). *The role of platform boundary resources in software ecosystems: a case study*. In 2014 IEEE/IFIP Conference on Software Architecture (pp. 11-20). IEEE. <https://ieeexplore.ieee.org/abstract/document/6827094>
- Braun, V., & Clarke, V. (2006). *Using thematic analysis in psychology*. *Qualitative research in psychology*, 3(2), 77-101. [https://www.researchgate.net/publication/235356393\\_Using\\_thematic\\_analysis\\_in\\_psychology](https://www.researchgate.net/publication/235356393_Using_thematic_analysis_in_psychology)
- Brocke, J., & Hevner, A. & Maedche, A. (2020). *Introduction to Design Science Research*. 10.1007/978-3-030-46781-4\_1. [https://www.researchgate.net/publication/345430098\\_Introduction\\_to\\_Design\\_Science\\_Research](https://www.researchgate.net/publication/345430098_Introduction_to_Design_Science_Research)
- Constantinides, P., Henfridsson, O., & Parker, G. (2018). *Platforms and Infrastructures in the Digital Age*. *Information Systems Research*. <https://doi.org/10.1287/isre.2018.0794>
- Cozzolino, A., Corbo, L. & Aversa, P. (2021). *Digital platform-based ecosystems: The evolution of collaboration and competition between incumbent producers and entrant platforms*. *Journal of Business Research*. Pages 385-400. <https://www.sciencedirect.com/science/article/pii/S0148296320308894>
- DHIS2 (n.d.). *Worldwide Map: DHIS2 in Action* [Online Image]. Retrieved 21 May 2022, from <https://dhis2.org/in-action/#map>
- DHIS2 (n.d.). *Managing dashboards* [Online Image]. Retrieved 21 May 2022, from <https://docs.dhis2.org/en/use/user-guides/dhis-core-version-237/analysing-data/dashboards.html>

- DHIS2 (n.d). *The Data Model* [Online Image]. Retrieved 14 April 2022, from [https://docs.dhis2.org/archive/en/2.30/developer/html/techarch\\_data\\_model.html](https://docs.dhis2.org/archive/en/2.30/developer/html/techarch_data_model.html)
- DHIS2 (n.d.). *DHIS2 App Hub*. Retrieved 14 May 2022, from <https://apps.dhis2.org>
- DHIS2 (n.d). *DHIS2 Documentation*. Retrieved 10 May 2022, from <https://docs.dhis2.org/en/home.html>
- DHIS2 (n.d). *DHIS2 Storybook*. Retrieved 21 May 2022, from <https://ui.dhis2.nu/demo/>
- Eaton, B., Elaluf-Calderwood, S., Sørensen, C., & Yoo, Y. (2015). *Distributed Tuning of Boundary Resources: The Case of Apple's iOS Service System*. *MIS Quarterly*, 39(1), 217–244. <https://www.jstor.org/stable/26628348>
- Edwards, R., & Holland, J. (2013). *What is qualitative interviewing?*. A&C Black. [https://www.researchgate.net/publication/313397132\\_What\\_is\\_Qualitative\\_Interviewing](https://www.researchgate.net/publication/313397132_What_is_Qualitative_Interviewing)
- Engert, M., Evers, J., Hein, A., & Krcmar, H. (2022). *The Engagement of Complementors and the Role of Platform Boundary Resources in e-Commerce Platform Ecosystems*. *Information Systems Frontiers*, 1-19. <https://link.springer.com/article/10.1007/s10796-021-10236-3>
- Erikson, M. (2021, February 2). *A Framework for Writing Better Documentation*. YCombinator. <https://news.ycombinator.com/item?id=26004534>
- Evans, P.C. & Gawer, A. (2016). *The Rise of the Platform Enterprise*. [https://www.thecege.net/app/uploads/2016/01/PDF-WEB-Platform-Survey\\_01\\_12.pdf](https://www.thecege.net/app/uploads/2016/01/PDF-WEB-Platform-Survey_01_12.pdf) [pdf].
- Flick, U. (2004). *Triangulation in qualitative research*. A companion to qualitative research, 3, 178-183.
- Foerderer, J., Kude, T., & Schuetz, S. (2014). *Add-on solution success: A configurational view on knowledge sharing in digital platforms*. ICIS. <https://aisel.aisnet.org/icis2014/proceedings/GeneralIS/20/>

- Foerderer, J., Kude, T., Schuetz, S.W., Heinzl, A. (2019) *Knowledge boundaries in enterprise software platform development: Antecedents and consequences for platform governance*. Wiley. <https://doi.org/10.1111/isj.12186>
- Gawer, A. (2021). *Digital platforms' boundaries: The interplay of firm scope, platform sides, and digital interfaces*. Long Range Planning, 54(5), 102045.  
<https://www.sciencedirect.com/science/article/pii/S0024630120302442>
- Ghazawneh, A. & Henfridsson, O. (2013). *Balancing platform control and external contribution in third-party development: The boundary resources model*. Information Systems Journal. 23. Doi: 10.1111/j.1365-2575.2012.00406.x  
[http://www.olahenfridsson.com/Ola/Publications\\_files/Ghazawneh%20and%20Henfridsson\\_late%20version.pdf](http://www.olahenfridsson.com/Ola/Publications_files/Ghazawneh%20and%20Henfridsson_late%20version.pdf) [pdf]
- Gill, P., Stewart, K., Treasure, E., & Chadwick, B. (2008). *Methods of data collection in qualitative research: interviews and focus groups*. British dental journal, 204(6), 291-295. <https://www.nature.com/articles/bdj.2008.192>
- Gregor, S. & Hevner, A. (2013). *Positioning and Presenting Design Science Research for Maximum Impact*. MIS Quarterly  
[https://www.researchgate.net/publication/262350911\\_Positioning\\_and\\_Presenting\\_Design\\_Science\\_Research\\_for\\_Maximum\\_Impact](https://www.researchgate.net/publication/262350911_Positioning_and_Presenting_Design_Science_Research_for_Maximum_Impact)
- Goldkuhl, G. (2012). *Pragmatism vs interpretivism in qualitative information systems research*. European Journal of Information Systems.  
<https://link.springer.com/article/10.1057/ejis.2011.54>
- Hein, A., Schreieck, M., Riasanow, T., Soto Setzke, D., Wiesche, M., Böhm, M., & Krcmar, H. (2019). *Digital platform ecosystems*. Electronic Markets. In press. 1-12.  
10.1007/s12525-019-00377-4.  
[https://www.researchgate.net/publication/337186627\\_Digital\\_platform\\_ecosystems](https://www.researchgate.net/publication/337186627_Digital_platform_ecosystems)
- Hevner, A. & Park, J. (2004). *Design Science in Information Systems Research*. MIS Quarterly [https://www.researchgate.net/publication/201168946\\_Design\\_Science\\_in\\_Information\\_Systems\\_Research](https://www.researchgate.net/publication/201168946_Design_Science_in_Information_Systems_Research)

- Holscher, E. (2021, February 2). *A Framework for Writing Better Documentation*. *YCombinator*. <https://news.ycombinator.com/item?id=26005918>
- Huber, T. L., Kude, T., & Dibbern, J. (2017). *Governance practices in platform ecosystems: Navigating tensions between cocreated value and governance costs*. *Information Systems Research*, 28(3).  
[https://www.researchgate.net/publication/312976468\\_Governance\\_Practices\\_in\\_Platform\\_Ecosystems\\_Navigating\\_Tensions\\_Between\\_Cocreated\\_Value\\_and\\_Governance\\_Costs](https://www.researchgate.net/publication/312976468_Governance_Practices_in_Platform_Ecosystems_Navigating_Tensions_Between_Cocreated_Value_and_Governance_Costs)
- Ivari, J. (2014). *Distinguishing and contrasting two strategies for design science research*. *European Journal of Information Systems*. 107–115.  
<https://doi.org/10.1057/ejis.2013.35>
- Kauschinger, M., Schrieck, M., Boehm, M., & Krcmar, H. (2021, March). *Knowledge Sharing in Digital Platform Ecosystems—A Textual Analysis of SAP’s Developer Community*. In *International Conference on Wirtschaftsinformatik* (pp. 21-39). Springer, Cham. [https://link.springer.com/chapter/10.1007/978-3-030-86797-3\\_2](https://link.springer.com/chapter/10.1007/978-3-030-86797-3_2)
- Kaushik, V. & Walsh, C.A. (2019). *Pragmatism as a Research Paradigm and Its Implications for Social Work Research*. <https://doi.org/10.3390/socsci8090255>
- Li, M. (2019). *An Approach to Addressing the Usability and Local Relevance of Generic Enterprise Software*. IRIS.  
[https://www.researchgate.net/publication/337946615\\_An\\_Approach\\_to\\_Addressin\\_g\\_the\\_Usability\\_and\\_Local\\_Relevance\\_of\\_Generic\\_Enterprise\\_Software](https://www.researchgate.net/publication/337946615_An_Approach_to_Addressin_g_the_Usability_and_Local_Relevance_of_Generic_Enterprise_Software)
- Li, M. (2021). *Generic Enterprise Software Implementation as Context for User- Oriented Design: Three Conditions and their Implications for Vendors*. Association for Information Systems. <https://aisel.aisnet.org/scis2021/4/>
- Moen, K., & Middelthon, A. L. (2015). *Qualitative research methods*. In *Research in medical and biological sciences* (pp. 321-378). Academic Press.
- Myers, M.D., Section Editor (living version): *Qualitative Research in Information Systems*. The University of Auckland, New Zealand. Originally published in MISQ Discovery, June 1997. <https://www.qual.auckland.ac.nz>



- Peffer, K., Tuunanen, T., Rothenberger, M. & Chatterjee, S. (2007). *A design science research methodology for information systems research*. Journal of Management Information Systems. 24. 45-77. <https://doi.org/10.2753/MIS0742-1222240302>
- Pershina, R., Soppe, B. & Thune, T. M. (2019). *Bridging analog and digital expertise: Cross-domain collaboration and T boundary-spanning tools in the creation of digital innovation*. [https://www.researchgate.net/publication/334762095\\_Bridging\\_analog\\_and\\_digital\\_expertise\\_Cross-domain\\_collaboration\\_and\\_boundary-spanning\\_tools\\_in\\_the\\_creation\\_of\\_digital\\_innovation](https://www.researchgate.net/publication/334762095_Bridging_analog_and_digital_expertise_Cross-domain_collaboration_and_boundary-spanning_tools_in_the_creation_of_digital_innovation)
- Procida, D. (2017). *Diátaxis documentation framework*. <https://diataxis.fr/>
- Sarker, S., Sahaym, A., & Bjørn-Andersen, N. (2012). *Exploring Value Cocreation in Relationships Between an ERP Vendor and its Partners: A Revelatory Case Study*. MIS Quarterly, 36(1), 317–338. <https://doi.org/10.2307/41410419>
- Schrieck, M., Wiesche, M., & Krcmar, H. (2016). *Design and governance of platform ecosystems-key concepts and issues for future research*. [https://aisel.aisnet.org/ecis2016\\_rp/76](https://aisel.aisnet.org/ecis2016_rp/76)
- Star, S. L., & Griesemer, J. R. (1989). *Institutional ecology, translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39*. Social studies of science, 19(3), 387-420. <https://www.jstor.org/stable/285080?seq=1>
- Tiwana, A., Konsynski, B. & Bush, A.A. (2010). *Research Commentary - Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics*. <https://pubsonline.informs.org/doi/abs/10.1287/isre.1100.0323>
- Tiwana, A. (2013). *Platform Ecosystems: Aligning Architecture, Governance, and Strategy*. Newnes.
- Tushman, M. L. (1977). *Special boundary roles in the innovation process*. Administrative science quarterly, 587-605. <https://www.jstor.org/stable/pdf/2392402.pdf> [pdf]
- UiO (n.d). *HISP UiO Strategy Update 2019-2022*. Retrieved 21 May 2022, from <https://www.mn.uio.no/hisp/english/about/strategy/hisp-uio-strategy-2019-2021.pdf> [pdf]

Von Hippel, E., & Katz, R. (2002). *Shifting innovation to users via toolkits*. *Management science*, 48(7), 821-833.

[https://www.researchgate.net/publication/5176470\\_Shifting\\_Innovation\\_to\\_Users\\_Via\\_Toolkits](https://www.researchgate.net/publication/5176470_Shifting_Innovation_to_Users_Via_Toolkits)

Yoo, Y., Henfriddson, O. & Lyytinen, K. (2010). *The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research*. *Information Systems Research*. 21. 724-735. 10.1287/isre.1100.0322. *Information Systems Research*. 21. 724-735. 10.1287/isre.1100.0322. <https://www.jstor.org/stable/23015640?seq=1>

Yoo, Y., Boland, R. J., Lyytinen, K., & Majchrzak, A. (2012). *Organizing for Innovation in the Digitized World*. *Organization Science*, 23(5), 1398–1408.

<http://www.jstor.org/stable/23252314>

# Appendix 1: HTML, CSS and Javascript survey

## Your background

Where you taking this course from abroad during the first assignment? \*

No

Yes

How much programming experience do you have? \*

*(not just front-end development)*

None

A bit (0-30 credits of university-level courses or an online course or two)

Some (30-60 credits, several online courses or some smaller personal projects)

Considerable (60+ credits, maybe some work experience or personal projects)

Extensive (120+ credits, work experience or larger personal projects)

Did you have experience with any of the following technologies before starting this course?

	No	Yes
HTML	<input checked="" type="radio"/>	<input type="radio"/>
CSS	<input checked="" type="radio"/>	<input type="radio"/>
Javascript	<input checked="" type="radio"/>	<input type="radio"/>
React	<input checked="" type="radio"/>	<input type="radio"/>
Vue	<input checked="" type="radio"/>	<input type="radio"/>
Angular	<input checked="" type="radio"/>	<input type="radio"/>

## HTML

Did you use the app-course website to learn or refresh your HTML knowledge? \*

*App-course website: [www.dhis2-app-course.ifi.uio.no](http://www.dhis2-app-course.ifi.uio.no)*

No

Yes, a bit

Yes, a lot

How helpful were the HTML resources on the app-course website? \*

Not at all helpful

Not so helpful

Somewhat helpful

Very helpful

Extremely helpful

## CSS

Did you use the app-course website to learn or refresh your CSS knowledge? \*

*App-course website: [dhis2-app-course.ifi.uio.no](http://dhis2-app-course.ifi.uio.no)*

No

Yes, a bit

Yes, a lot

How helpful were the CSS resources on the app-course website? \*

Not at all helpful

Not so helpful

Somewhat helpful

Very helpful

Extremely helpful

## Javascript

Did you use the app-course website to learn or refresh your Javascript knowledge? \*

*App-course website: dhis2-app-course.ifi.uio.no*

No

Yes, a bit

Yes, a lot

How helpful were the Javascript resources on the app-course website? \*

Not at all helpful

Not so helpful

Somewhat helpful

Very helpful

Extremely helpful

Could you expand on why you found the app-course website resources helpful or not helpful? \*

- Please specify what section you are giving feedback about (HTML, CSS, JS)

Did you use any other resources than the app-course website to learn HTML, CSS and JS? \*

If so, what?

Google

Stack overflow

W3schools

Developer documentation

Other online courses / MOOCs

YouTube

Mattermost channel

Fellow students

Group seminars

Other

Group seminars

How many group seminars did you attend? \*

▪ *Before the assignment due date (10.09.2021)*

None

1

2

## Assignment

From a scale 1-10 did this assignment help you learn HTML, CSS and JS? \*



Value



Did you have any challenges with the assignment? \*

*If so, how did you overcome these?*

Do you have any feedback about the assignment?

- What worked well?
- What did not work so well?
- What could be improved?

# Appendix 2: React Survey

## Your background

How much programming experience do you have? \*

*(not just front-end development)*

- None
- A bit (0-30 credits of university-level courses or an online course or two)
- Some (30-60 credits, several online courses or some smaller personal projects)
- Considerable (60+ credits, maybe some work experience or personal projects)
- Extensive (120+ credits, work experience or larger personal projects)

Did you have any previous experience with React before this course? \*

- Yes
- No

Did you have any previous experience with querying APIs before this course? \*

- Yes
- No

## App-course

Did you use the app-course website to learn or refresh your React knowledge? \*

- No
- Yes, a bit
- Yes, a lot



## App-course

Did you use the app-course website to learn or refresh your React knowledge? \*

No

Yes, a bit

Yes, a lot

How helpful were the React resources on the app-course website? \*

Not at all helpful

Not so helpful

Somewhat helpful

Very helpful

Extremely helpful

Could you expand on why you found the app-course website resources helpful or not helpful? \*

- *Were the concepts explained well enough?*
- *Did the tutorials help to understand React?*
- *Did the linked resources help to understand the concepts?*

Did you do/ watch any of the React tutorials linked to from the app-course webpage?

Programming with Mosh

Tic Tac Toe

Ticking clock

## Group seminar

How many group seminars about React did you attend? \*

*After the deadline of the first assignment but before the second assignment due date (25.09)*

0

1

2

3

More

Do you have any feedback regarding these group seminars? \*

- What worked well?
- What did not work so well?
- What could be improved?

## Assignment

From a scale 1-10 how well did this assignment help you learn React? \*



Value



Did you have any challenges with the assignment? \*

*If so, how did you overcome these?*

Do you have any feedback about the assignment?

- What worked well?
- What did not work so well?
- What could be improved

# Appendix 3: DHIS2 Survey

How much programming experience do you have? \*

*(not just front-end development)*

- None
- A bit (0-30 credits of university-level courses or an online course or two)
- Some (30-60 credits, several online courses or some smaller personal projects)
- Considerable (60+ credits, maybe some work experience or personal projects)
- Extensive (120+ credits, work experience or larger personal projects)

How helpful were the DHIS2 resources on the app-course website? \*

- Not at all helpful
- Not so helpful
- Somewhat helpful
- Very helpful
- Extremely helpful

Could you expand on why you found the app-course website resources helpful or not helpful? \*

- *Were the concepts explained well enough?*
- *Did the tutorials help you understand DHIS2?*
- *Did the linked resources help?*

Did you use any other resources to learn DHIS2 app development?

developers.dhis2.org

docs.dhis2.org

DHIS2 Academy YouTube videos

runtime.dhis2.nu

ui.dhis2.nu

cli.dhis2.nu

Other

Did you try the Data Query Playground? \*

No

Yes

Did you find the Data Query Playground useful? \*

Did you use Storybook to implement UI components? \*

No

Yes

Did you find Storybook useful? \*

## Group seminar

How many group seminars about DHIS2 did you attend? \*

*After the deadline of the second assignment but before the third assignment due date (08.10)*

0

1

2

3 or more

Do you have any feedback regarding these group seminars? \*

- What worked well?
- What did not work so well?
- What could be improved?

## Assignment

From a scale 1-10 how well do you feel this assignment prepared you for the group project? \*



Value



Was the API from the second assignment helpful in preparing you for working with the DHIS2 API? \*

Not at all helpful

Not so helpful

Somewhat helpful

Very helpful

Extremely helpful

Did you have any challenges with the assignment? \*

*If so, how did you overcome these?*

Do you have any feedback about the assignment?

- What worked well?
- What did not work so well?
- What could be improved

# Appendix 4: Final Survey

## Warming Up

How much programming experience do you have? \*

*(not just front-end development)*

- None
- A bit (0-30 credits of university-level courses or an online course or two)
- Some (30-60 credits, several online courses or some smaller personal projects)
- Considerable (60+ credits, maybe some work experience or personal projects)
- Extensive (120+ credits, work experience or larger personal projects)

What project case did your group choose? \*

- Case 1: Commodity Dispensing
- Case 2: Data entry forms

What did you work with in the project? \*

*Select all applicable*

- Design and/or wireframing
- User interface
- API / data preparation
- Project management
- Testing
- Other

## Course

When the project started, how well prepared were you for developing an application on DHIS2? \*

*With regards to technical skills and knowledge e.g. HTML, CSS, JS, React, DHIS2, APIs and git.*



Value





Was there anything that you were not prepared for when the project started? \*

*With regards to technical skills and knowledge e.g. HTML, CSS, JS, React, DHIS2, APIs and git.*

## Project

How helpful was the DHIS2 section on the app-course website when working on the project? \*

Not at all helpful

Not so helpful

Somewhat helpful

Very helpful

Extremely helpful

Could you expand on why you found the resources helpful or not helpful for the project? \*

Did you ask for the group teachers for guidance during the project? \*

No

Yes

What did you ask for guidance about? \*

Did the group teachers help you overcome these problems? \*

No

Yes, somewhat

Yes

## API

Did you personally work with the DHIS2 API during the project? \*

No

Yes

Was it challenging to work with the DHIS2 API? \*

Extremely challenging

Very challenging

Somewhat challenging

Not so challenging

Not at all challenging

What specific challenges did you face when working with the API? \*

How helpful were these resources when using the API?

	Didn't use	Not at all helpful	Not so helpful	Somewhat helpful	Very helpful	Extremely helpful
<a href="https://dhis2-app-course.ifi.uio.no">dhis2-app-course.ifi.uio.no</a>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<a href="https://developers.dhis2.org/">developers.dhis2.org/</a>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<a href="https://docs.dhis2.org">docs.dhis2.org</a>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Data Query Playground	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
DHIS2 academy webinars	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Group seminars	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Mattermost	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other students	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Could you expand on why these were helpful or not helpful? \*

*Please specify which resource you are referring to.*



Could you expand on why these were helpful or not helpful? \*

*Please specify which resource you are referring to.*

**Other**

Is there anything else you'd like to add?

# Appendix 5: Preliminary study interview guide

1. What program are you currently enrolled in?
2. Before you took the IN5320 course, did you have any previous programming experience?
3. Do you program in your freetime, through work or mostly school-related?
4. How many programming courses did you take throughout your bachelors?
5. Are you planning on working with software development (specifically programming) after graduating your masters?
6. What was your motivation for taking this course? *Interested in global development? Learning front-end development? Working with practical projects? Working with a real-life project?*
7. Did the course fulfil your motivations?
8. This is a bit of a recap, what do you remember from the programming part of course?
  - *Tasks, HTML, CSS, React, DHIS2 API, Assignment 1 (website), Assignment 2 (DHIS2), Project, Home exam.*
9. What did you think about the mandatory exercises?
  - *Assignment 1, 2. Should we have more assignments?*
10. What didn't you like about the course?
11. What did you think about the group project?
12. What role did you take in the group project?
13. Did you find the DHIS2 App Course website helpful as a learning resource?
  - *Which elements of the website were especially helpful? Why? Were there any elements that you found confusing about the website*
14. How did you experience the interactive coding exercises?
15. How did you learn HTML?
16. How did you learn CSS?
17. How did you learn Javascript?
18. How did you learn React?
19. How did you learn the DHIS2 API?
20. Did you find any topics covered by DHIS2 App Course insufficient?
21. Did you feel the course website prepared you enough for the mandatory exercises and the group project?
22. Name three things that you liked about the course website.
23. Name three things that could be improved on the course website.
24. What formats do you prefer for learning, for example learning by doing, lectures, youtube videos etc?

# Appendix 6: Focus groups

1) Walkthrough **React** content individually and write down (10 min):

- What did you like about how the content was presented to you?
- What didn't you like about the content and how it was presented?
- What could be improved to give a better presentation of React?

2) Discuss your opinions with the others in the group. Everyone should have the opportunity to share their opinion and have a say in the discussion. The goal is to exchange your opinions and find a consensus. Finally, present your findings to us (5 min)

3) **Assignment 2.** Discuss (5 min):

- What did you like or dislike about the assignment?
- Were there any challenges related to solving the assignment?
- What did you learn?

4) Walkthrough **DHIS2** getting started guide individually and write down (10 min):

- What did you like about how the content was presented to you?
- What didn't you like about the content and how it was presented?
- What could be improved to give a better introduction of DHIS2?

5) Discuss what you wrote down with the others (10 min):

- Present your findings to us.

6) **Assignment 3.** Discuss (5 min):

- What did you like or dislike about the assignment?
- Were there any challenges related to solving the assignment?
- What did you learn?

7) Discuss (10 min). *Give the students a list of all of the boundary resources:*

- Which ones have you used?
- Were the resources helpful when learning about DHIS2?
- What did you like/dislike about the resources?

8) **Group lectures.** Discuss (5 min):

- What did you like about the group lecture?
- What could be improved?
- Were the group lectures necessary in addition to the provided online resources?