FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Virtual guide to oppidum Závist
Student:	Vladislav Komkov
Supervisor:	Ing. Radek Richtr, Ph.D.
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Computer Graphics
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2023/2024

Instructions

The aim of the work is to design and implement a mobile application for the iOS operating system, which will serve as a virtual guide to the Celtic oppidum Závist. The application will use augmented reality for the realistic presenting 3D models of historical buildings. The application will also include an interactive map and gamification of the guide.

1. Conduct research on similar applications using augmented reality to display 3D models.
2. Analyse the requirements of users and project stakeholders.
3. Using the SI methods, design a prototype of the application.
4. Implement the prototype application for the iOS operating system.
5. Subject the prototype to appropriate tests.

Electronically approved by Ing. Radek Richtr, Ph.D. on 23 February 2023 in Prague.

Bachelor’s thesis

VIRTUAL GUIDE OF THE OPPIDUM ZÁVIST

Vladislav Komkov

Faculty of Information Technology
Department of Software Engineering
Supervisor: Ing. Radek Richtr, PhD
May 11, 2023

Czech Technical University in Prague
Faculty of Information Technology
© 2023 Vladislav Komkov. All rights reserved.
This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis: Komkov Vladislav. *Virtual guide of the oppidum Závist.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

List of Figures

List of Tables

List of code listings

I would like to thank my supervisor Ing. Radek Richtr, PhD for assisting me throughout the process of conducting the research and the development, as well as David Pesek who was eager to help with all questions connected to the project management. Not to forget Jiri Chludil who assisted with the network communications and Manda Michal for helping with the organization of the User testing. And huge appreciation to all the members of the CCHAOS research group for the opportunity to work on this project and for all the information and help provided during the process.

Declaration

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. I further declare that I have entered into a licence agreement with the Czech Technical University in Prague for the utilization of this thesis as a school work pursuant to Section 60(1) of the Copyright Act. This fact does not affect the provisions of Section 47b of Act No. 111/1998 Coll., the Higher Education Act, as amended.

In Prague on May 11, 2023 . . . .

Abstract

This thesis presents the development of the iOS application of a virtual guide for Celtic Oppidum Závist. The application will provide visitors of the Oppidum immersive experience extending the historical walk with information relevant to the place where the user is located on the map. The project involves an analysis of existing digital guides, including the current solution of the Oppidum Závist. The domain analysis touches on historical and archaeological domains, documenting the requirements of the project. The study also examines options for iOS development as well as different software architecture patterns. The implementation section focuses on user-centred design and the prototype creation process, including the cognitive test. It summarises tools and dependencies used in the development as well as specific details of native iOS development in Swift programming language. The thesis concludes with user testing of the solution with the analysis of the results and comparison with the current solution. The thesis aims to provide an engaging and informative experience for visitors of the Oppidum, while also exploring the latest iOS development and software engineering techniques.

Keywords virtual guide, mobile application, Oppidum Závist, user-centred design, user-testing, Figma, iOS, Swift

Abstrakt

Tato práce prezentuje vývoj iOS aplikace virtuálního průvodce pro keltské oppidum Závist. Aplikace poskytne návštěvníkům oppida zážitek rozšiřující historickou procházku informacemi relevantními k místu, kde se uživatel na mapě nachází. Projekt zahrnuje analýzu stávajících digitálních průvodců, včetně současného řešení pro oppidum Závist. Analýza domény se dotýka historické a archeologické oblasti jenž popisuje požadavky projektu. Práce také zkoumá možnosti vývoje pro iOS a různé architektonické vzory softwaru. Implementační část se zaměřuje na user-centered design a proces vytváření prototypů, včetně jeho kognitivního testování. Shrnuje nástroje a závislosti použité při vývoji, stejně jako specifické detaily nativního vývoje pro iOS v programovací jazyce Swift. Práce končí uživatalským testováním řešení, analýzou výsledků a porovnáním s aktuálním řešením. Cílem práce je poskytnout návštěvníkům oppida zajímavý a informativní zážitek, přičemž prozkoumává nejnovější techniky vývoje pro iOS a softwarové inženýrství samotné.

Klíčová slova virtuální průvodce, mobilní aplikace, oppidum Závist, user-centered design, uživatelské testovaní, Figma, iOS, Swift

List of abbreviations

API	Application Programming Interface
App	Application
AR	Augmented reality
HIG	Human interface guidelines
IA	Information Architecture
IS	Information system
MVC	Model View Controller
MVVM	Model View ViewModel
UI	User Interface
UX	User Experience

Introduction

Introduction

In the current era, technology has advanced significantly and has the potential to revolutionize the way people navigate historical sites and learn about the history surrounding them. With the growing interest in outdoor activities and the historical heritage of various places, there is a need for innovative solutions that can attract and educate more people about history. Learning about history through new technologies has the potential to make it more engaging and accessible to a wider audience.

However, the lack of investments and interest from companies has resulted in multiple challenges for virtual guides. Existing applications have several weak points in information architecture as well as in user interface design. Analysis of such applications gives a chance for an improved solution for Celtic oppidum, chosen historical heritage with huge potential of telling a story about more than 5000 years old settlement.

The main focus of this thesis is the user-centered design, how different approaches of prototyping and user-experience testing can influence development cost and speed as well as the final product. The thesis is aimed at iOS development, but for the sake of digital sustainability it is important to research different technologies for iOS development including multiplatform and game development technologies.

To achieve this goal, a survey will be conducted on applications serving a similar purpose of acting as a virtual guide to historical heritage using modern approaches and technologies. It is important to note that there is a virtual guide to Celtic oppidum Závist that is referenced throughout this text. For further improvement of the solution research will be carried out on the domain and the information provided by archaeologists which would further improve user experience, thus benefiting the user. It is also crucial to analyse the requirements of the ordering party, as well as the needs and expectations of potential users.

The practical part of the thesis will focus on the design workflow as well as the implementation of a high-fidelity prototype in Swift programming language. The implemented prototype will be subjected to user experience and unit tests. The main challenge is the integration of advanced interactive features such as a 3D browser for artefacts and augmented reality that can help users understand the scale of buildings and other artefacts, keeping the user interface simple and understandable. This will provide visitors with an immersive experience as well as the opportunity to learn more when they are on the historical site without spending time figuring out how to use the application.

The proposed solution can serve as a blueprint for other researchers and developers looking to create digital solutions for historical sites. The significance of this thesis lies in its potential to promote interest and educate visitors on historical sites through innovative digital solutions.

Part I

Research

Chapter 1
Solutions

In this chapter, similar solutions will be analyzed in information architecture, interactive and visual design to better understand their strengths and weaknesses. By examining these solutions, we will be able to identify common patterns and trends that can be applied to our work. Additionally, we will explore alternative approaches that have been used in the past, and consider whether these might be applicable to guide. Through this comprehensive analysis, we hope to gain a deeper understanding of the digitalization of the guide.

1.1 Current solution of digital guide Oppidum Závist

To ensure the effectiveness of the virtual guide application, it is important to analyze the current solution. This analysis will enable a comprehensive understanding of the problem and facilitate an evaluation of the strong and weak aspects of the current solution. Identifying areas for improvement will be essential to enhancing the application's features and functionality, and to provide an even more engaging and informative experience for users.

Despite the potential for improvement, the application is currently available for both iOS and Android operating systems. The cross-platform compatibility of the application also ensures that users with different device preferences can experience it.

1.1.1 Information architecture

On the main screen 1.1, the user sees the main parts of the application with helpers such as language change buttons and Instructions to use information. A user chooses and starts one part of the application. From there he can go back to the main menu or continue with the application flow. Two main parts of the application are:

Digital guide The application displays an image of the map to the user 1.2, which provides several options for engaging with the content. From the map image, the user can start navigation, access detailed information about the location, and move to the relevant segment of the game. The detailed information 1.3 presented to the user includes unstructured text and feature buttons, enabling a more comprehensive exploration of the location's history and cultural significance. This feature enhances the user's experience by providing a more interactive and informative means of engaging with the virtual guide.

Game The game follows a linear story in which the user assists a scientist in solving problems related to various historical eras. As the user progresses through the story, they gain a deeper understanding of the life and culture of Oppidum in different time periods.

The application's information architecture (IA) presents several challenges that impact the user experience. One key issue is the inability of users to freely switch between the game and map modes, which can limit the user's ability to navigate the application effectively. Additionally, the point details section suffers from a lack of text organization, which can make it difficult for users to access the information they need. Addressing these issues will be critical to improving the user experience and enhancing the effectiveness of the virtual guide application.

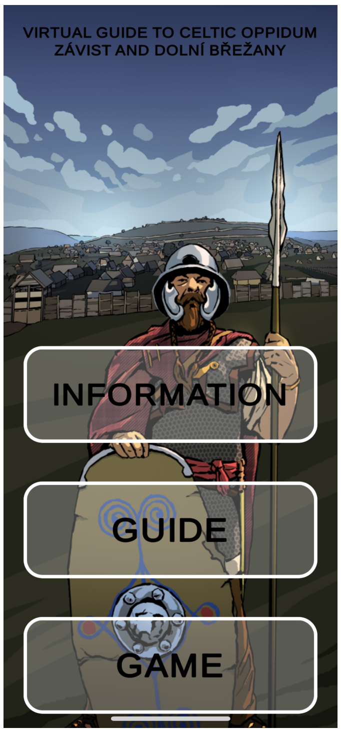


Figure 1.1 Current solution menu interface [1]

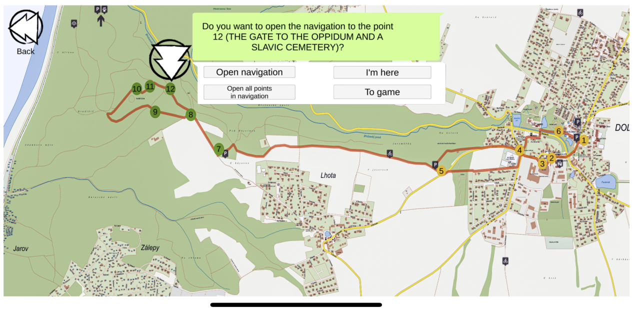
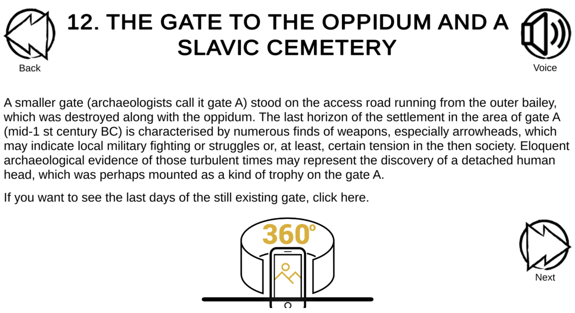

Back
Do you want to open the navigation to the point 12 (THE GATE TO THE OPPIDUM AND A SLAVIC CEMETERY)?
Open navigation
I'm here
Open all points in navigation
To game
Lhota
Jarov
Zálepy
DOL

Figure 1.2 Current solution map screen. [1]



12. THE GATE TO THE OPPIDUM AND A SLAVIC CEMETERY

A smaller gate (archaeologists call it gate A) stood on the access road running from the outer bailey, which was destroyed along with the oppidum. The last horizon of the settlement in the area of gate A (mid-1 st century BC) is characterised by numerous finds of weapons, especially arrowheads, which may indicate local military fighting or struggles or, at least, certain tension in the then society. Eloquent archaeological evidence of those turbulent times may represent the discovery of a detached human head, which was perhaps mounted as a kind of trophy on the gate A.

If you want to see the last days of the still existing gate, click here.

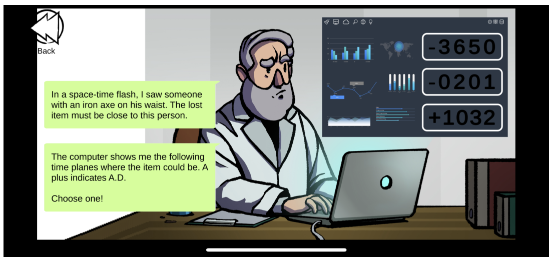
Figure 1.3 Current solution point detail screen. [1]

1.1.2 Interactive design

The app gives users 2 different interactive experiences, one of them being 360 panoramas and the second being Augmented reality. Buttons for each of them are located between the text and the context for interaction should be read in the text.

The game section 1.4 of the application guides them on what to do with text instead of relying on UI. This results in forcing users to read instructions instead of explore, limiting users' creativity. Taking into consideration targeted users (children), gameplay could benefit from more interactive elements such as one in 1.5. The inability of switching between map and game segments creates a lack of flexibility and limits users to a single activity at a time, which can reduce the overall effectiveness of the virtual guide application. [2]

The application presents an orientation issue, as the main menu is designed to be viewed in portrait mode while the remainder of the app is locked to landscape mode. This creates a disorienting experience for the user, which can reduce the overall effectiveness of the virtual guide. According to the Apple Human Interface Guidelines (HIG), it is recommended to avoid locking the user to only one orientation. [3]


In a space-time flash, I saw someone with an iron axe on his waist. The lost item must be close to this person.

The computer shows me the following time planes where the item could be. A plus indicates A.D.

Choose one!

Figure 1.4 Current solution game story. [1]

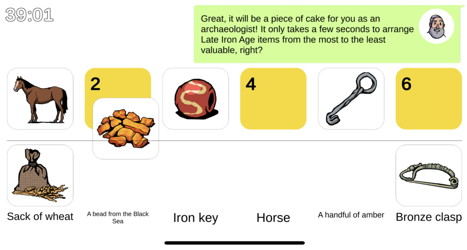

Great, it will be a piece of cake for you as an archaeologist! It only takes a few seconds to arrange Late Iron Age items from the most to the least valuable, right?

Figure 1.5 Current solution gameplay. [1]

1.1.3 Visual design

The Unity game engine, while suitable for game development, presents challenges when developing an interactive system such as the virtual guide application. The application lacks elements of the iOS system navigation flow and other design elements specific to the iOS platform, which can result in a disorienting experience for users who are accustomed to these elements. Additionally, the application's scaling issues, caused by the use of static scale constraints and images as backplates, pose a challenge to the application's adaptability to different device sizes. This scaling issue is also noted in Apple's Human Interface Guidelines [3]

1.2 Visit.More

Visit.More is a virtual guide developed by a Czech company for iOS and Android that provides a catalogue of historical monuments and infrastructure in the area. The application offers detailed information, including historical significance, architectural features, and cultural context, making it an engaging and informative tool for those interested in exploring the rich cultural heritage of different regions.

1.2.1 Information architecture

The application has two distinct components. The first of these components is the location library 1.6 and overview section, which provides users with a list of available locations for discovering. The second component of the application is the guide to a particular location, which serves as detailed information about individual landmarks located on the site.

- On the home screen, the user can choose the language of the application and select a monument to start the guide.
- Upon selecting a monument, the user is immediately provided with information about the location, illustrated with an aerial snapshot.
- The detail screen provides the user with more detailed information about the location, including basic information associated with the location such as website, navigation, and operators.

The Guide Map features a 3D viewer that highlights points of interest 1.7. For each point of interest, the user can access extended information in the form of text, images, 3D models, and augmented reality. The application distinguishes between these different types of information by using separate icons for each of them.

1.2.2 Interactive design

The application has numerous forms of information, enabling user to select one that suits him the best. Upon initial launch, a guide presents a help screen to familiarize the user with the navigation and the application's features. The content is available in a diverse range of information formats, including conventional textual and visual modes, in addition to advanced features such as audio guides, 3D model viewing, and augmented reality. This features differs from point to point using the most suitable type of information 1.8

To ensure that the application operates efficiently, the developers have implemented a data size notification feature. Before downloading content, the user is notified of the data size to be downloaded and prompted to connect to Wi-Fi. This feature enhances the user experience by reducing the likelihood of unexpected data charges or slower download speeds.

1.2.3 Visual design

The absence of system elements in the application design may increase the learning curve for users who are unfamiliar with the app's interface, as they may not be able to rely on preexisting knowledge of system-level design conventions. The application interface presents a complex and information-dense layout but maintains consistency in its design elements, which may aid users in locating and utilizing application features more efficiently. Although the application does not display the user's location, it offers a unique 3D map view feature that may provide users with an alternative method of orienting themselves within the application's geographic context. However, without conducting tests it remains unclear how effective this feature compensates for the absence of the user location indicator and whether it offers comparable functionality for user orientation.

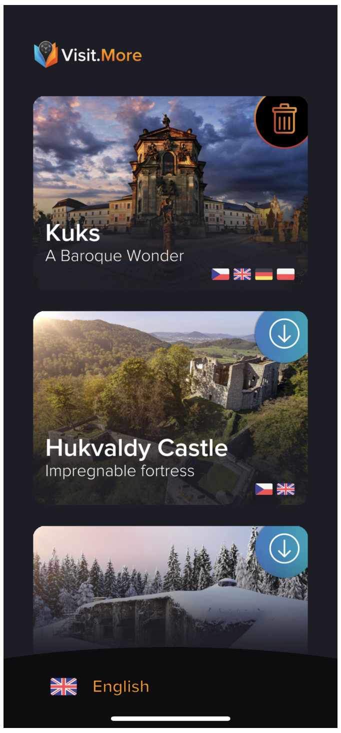


Figure 1.6 Visit.More Menu [4]

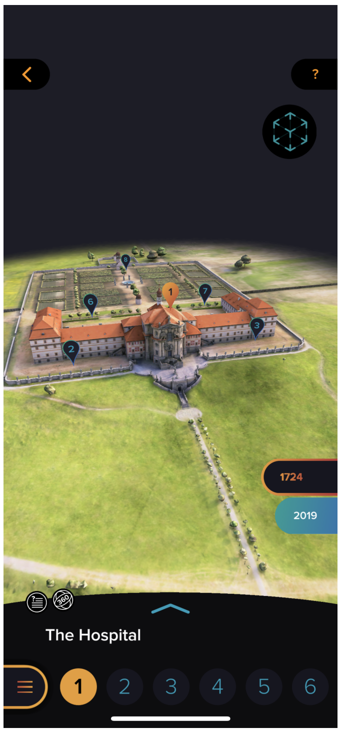
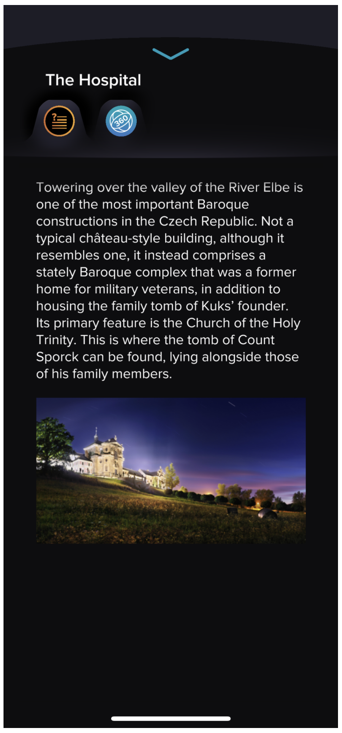


Figure 1.7 Visit.More 3D map [4]


The Hospital

Towering over the valley of the River Elbe is one of the most important Baroque constructions in the Czech Republic. Not a typical château-style building, although it resembles one, it instead comprises a stately Baroque complex that was a former home for military veterans, in addition to housing the family tomb of Kuks' founder. Its primary feature is the Church of the Holy Trinity. This is where the tomb of Count Sporck can be found, lying alongside those of his family members.

Figure 1.8 Visit.More Information content [4]

1.3 VMCK virtual guide

VMCK(Věnná města českých královen) is an app for the Android operating system, it is an augmented reality (AR) application designed to provide an immersive experience of exploring the historical heritage of the Dowry towns. The app leverages the capabilities of GPS technology to enable the accurate positioning of 3D models of historical buildings at their actual location. The application is currently in development mode and focuses on addressing several challenges related to compass tracking, thus UX and UI are not in the scope of this analysis.

The primary feature of the app is the ability to overlay 3D models of historical buildings onto the user's view of the real-world environment. The app requires access to the GPS location of the user's device to determine the location and to present the building that could be seen from the user's location. Once the location is determined, the app retrieves the 3D model of the historical building from a remote server and displays it on the user's device.

One of the significant challenges of the app is the accuracy of the GPS data. The accuracy of GPS location data can be affected by several factors, including environmental conditions such as interference from buildings, trees, and other structures. In addition, the accuracy of compass tracking can be compromised in urban environments due to the presence of electromagnetic interference. These issues can result in incorrect positioning of the 3D models of the buildings, leading to a less immersive user experience.

The app implements a calibration process to improve the accuracy of the compass tracking. The calibration process involves the user pointing the device in the known direction. This data is then used to calibrate the compass, improving its accuracy.

All the data required to render the 3D models of the historical buildings are fetched from the remote server and stored on the user's device until the model is updated. This approach ensures that the app can be used offline and reduces the latency associated with fetching data from the server.

In summary, the proposed AR application is designed to provide an immersive experience of exploring historical buildings by overlaying 3D models onto the user's view of the real-world environment. The app employs several techniques to address the challenges associated with GPS accuracy and compass tracking. The app's development is ongoing, and further work is required to optimize the performance and user experience of the application.

1.4 Analysis results

Each of the three solutions employs distinct strategies to address the challenge of virtual guidance, incorporating modern technologies in diverse ways. While some emphasize the authenticity of the building's location, others seek to engage the user through the use of gaming narratives. Each approach exhibits unique strengths, and leveraging these strengths can facilitate further enhancements to our solution.

Table 1.1 Overview of the analysed solutions

Name	**Description**	**Structure**	**AR**	**Orientation**	**Accessibility**
Oppidum Závist	Virtual guide to Keltic oppidum Závist including a game for better orintation on side	Separate game and map sections	Activated with marker	Mixed, locked	Limited
VMCK	A city guide accurately showing historical buildings in different period in the history of the monument	Map with zones in which user can activate Extended reality	Models' location are determined by GPS for accurate historical representation	Vertical	Limited
Visit.More	Guide to many Czech attractions with 3D maps	User chooses a location, after flying over 3D map and learns about marked points	Limited, not anchored to the location	Vertical	Limited

Chapter 2

Product Analysis

The application domain is quite specific, combining the historical and archaeological domains not forgetting the guide part. Thus analysis of the domains as well as the analysis of the requirements of the stakeholders is essential to ensure that the developed application meets the requirements of the domain and the intended users.

2.1 Product requirements document

A product requirements document is a comprehensive and detailed outline of all the necessary information about the product: problematics, goals and personas for the understanding of the problem and user in one place. The process of creating a product requirements document involves the effort of all team members and stakeholder engagement, providing everything needed for the creation of the product requirements. Through this process, the team gains a thorough understanding of the problem and the user, which ultimately helps to guide the design and development of the product. The main outputs from the analysis are the summary of the goals and personas further specifying the application development process. The full document can be found in the appendix C.

2.2 Domain analisys

Domain analysis is a phase in software development that involves researching the methods and specifics of the domain to better understand the problem space and design a better information architecture that suits specific scenarios of virtual guides. This process is based on information science, which is concerned with the collection, organization, analysis, and dissemination of information [5]. Although information science is not the primary topic of your thesis, it is still essential to understand its fundamental principles to conduct an effective domain analysis.

2.2.1 Historical and Archaeological domains

To gain a better understanding of the domain of Oppidum Závist, consultation with archaeologists and other domain experts was done to gather information about the historical and cultural significance of the site.

During the discussion with archaeologist Mgr. Daniel Bursák [6], it was revealed that the archaeological research conducted at Oppidum Závist was active for almost three decades, from 1963 to 1990. The findings of this research were meticulously documented by all archaeological standards, resulting in a wealth of valuable information about the historical site.

Mr Bursák emphasized the importance of making this information accessible to the public and popularizing the fascinating findings of the research. However, the current application for the virtual guide is unnecessarily complicated, which makes it difficult to use while exploring the site. As the guide himself, he points out that using the phone is not a priority for visitors to a historical site. These insights will be valuable in informing the design and development of an improved virtual guide application that is more user-friendly and tailored to the needs of visitors to Oppidum Závist.

2.3 Requirements

Writing down the functional and non-functional requirements of a software application is a critical component of the software development process. Functional requirements are describing functions, services, and features that a software system should have to satisfy the user's needs and solve their problems. While non-functional describes qualities of a software system that are important but do not relate to its specific functionality [7]. This will help with the understanding of the project goals, identify constraints and limitations, and establish a clear roadmap for project completion. Requirements that are separated by the application parts are described application flow diagram in appendix B

2.3.1 Functional requirements

F1 Navigation

- The user should be able to navigate the Oppidum Závist site using the app. It implies that the app should provide an intuitive and user-friendly interface for navigating the site.

F2 Points of Interest

- Users must be able to view points of interest on the map and get more information about them.

F3 Interactive Elements

- The app must include interactive elements such as 3D models, panorama, and AR view.

F4 Geocaching Game

- The app should include a geocaching-style game where users can interact with points of interest and answer quiz questions.

F5 Progress Tracking

- The app should keep track of the user's progress in the game and provide rewards upon completion.

F6 User is able to turn off game elements

- Setting panel must be present for the user to enable/disable game functionality

2.3.2 Nonfunctional requirements

N1 Web Application Integration

- The app should be integrated with a web application that allows administrators to modify points of interest.

N2 Usability

- The app should have an intuitive interface that is easy for users to navigate and interact with.

N3 Performance

- The app should be fast and responsive, with minimal lag or delay.

N4 Compatibility

- The app should be compatible with iOS devices and meet Apple's App Store requirements.

Conducting a thorough domain and requirement analysis helps identify the key concepts, entities, relationships, and processes that took place on the Oppidum Závist. This knowledge can then be used to develop an application that meets the needs and expectations of the target audience and provides an engaging and informative virtual guide to this fascinating historical site.

Chapter 3

Technologies

3.1 Options for iOS development

Abstract

The focus of this thesis centres on the development of iOS applications. As iOS development has evolved over the years, the field has transitioned from a predominantly native approach to one that incorporates a range of development frameworks. Researching each of these frameworks can facilitate informed decision-making regarding the selection of an optimal framework for our specific solution.

3.1.1 Multiplatform Development

As the demand for mobile applications continues to rise, the need for efficient and cost-effective cross-platform development solutions has become increasingly critical, leading to the emergence of promising technologies [8] such as Kotlin Multiplatform (KMM), React Native, and Flutter for developing applications that can run seamlessly on both iOS and Android platforms.

The main advantage of all such platforms is the ability to share code between multiple platforms. This feature streamlines development by reducing the need for duplicated code, enabling developers to focus on platform-specific functionalities. Furthermore, it provides a single codebase, which can result in significant time and cost savings.

Its biggest advantage of generics is leading to the complex development process due to the platform-specific differences, which may require developers to write platform-specific code. Each framework solves this problem differently.

According to a study on the performance of cross-platform mobile app development using web-based multiplatform tools [9] can struggle with performance whenever there are callbacks from one codebase to another, some of the frameworks address this problem by converting code to use native components.

3.1.1.1 Kotlin Multiplatform

KMM – Kotlin Multiplatform Mobile goes around the problem of platform-specific code. It generates a common core for the business and data layers of the app, leaving implementation of the user interface to the developers of each platform [10]. It does not eliminate the need for platform-specific code. Thus, developers must possess a deep understanding of both platforms in order to effectively utilize KMM.

One of the key benefits of KMM is the ability to use the full potential of native development. With KMM, developers can write native code for both Android and iOS platforms, which can lead to increased performance and a better user experience, providing developers with access to the native APIs and toolkits of each platform.

However, it is important to note that KMM is still in beta version and there are no stable production versions yet [11]. This means that there may be bugs, compatibility issues, or other problems that could impact the stability and performance of KMM. Developers must be aware of these limitations and be prepared to work around them in order to effectively utilize KMM in their projects.

Despite these challenges, KMM represents a promising approach to mobile app development that can help reduce development time and improve code quality. As such, developers need to stay informed about KMM to stay ahead in the ever-evolving mobile app development.

3.1.1.2 React native and Flutter

Flutter and React Native are two popular cross-platform mobile app development frameworks, each with its strengths and weaknesses. Table 3.1 highlights some of the key differences between the two frameworks across five aspects: programming language, user interface, performance, development tools, and community support. Based on the documentation of the React Native from Facebook team [12] and Flutter provided by its developers [13]

From a UI standpoint, React uses native design components, and therefore user gets as native experience as possible with multiplatform development. Whereas Flutter uses Its widgets, creating equivalent UX on both platforms.

Table 3.1 Differences between Flutter and React Native

Aspect	**Flutter**	**React Native**
Programming Language	Dart	JavaScript
User Interface	Flutter renders its widgets resulting in a consistent look and feels across platforms	React Native uses native components resulting in differences in the UI on different platforms
Performance	Dart and reactive programming model provide faster app performance	Performance may be affected by differences in the way native components are implemented on each platform
Development Tools	Comes with a comprehensive suite of tools including its integrated development environment (IDE) and widget library	Relies on third-party tools and libraries
Community Support	Growing rapidly and known for its active contributions to open-source development	Larger community with a long history resulting in a more extensive ecosystem of third-party libraries, plugins, and tools

3.1.2 Native Development

Native mobile development provides several benefits over cross-platform development approaches. By building a native app, developers can create an app that delivers the best possible performance and user experience, taking full advantage of the design elements and features specific to each platform.

Native apps offer superior performance because they are optimized for the platform's hardware and software [9]. This means that the app can run faster and more smoothly than cross-platform alternatives, providing users with a better experience. Additionally, native apps can take advantage of platform-specific features, such as the LidAR scanner in iPhones or the S Pen in Samsung Galaxy devices. These features allow developers to create innovative and immersive user experiences that are not possible with cross-platform development frameworks.

Another advantage of native development is the ability to deliver a high-quality user interface that is consistent with the platform's design language. This means that the app will look and feel like a natural part of the device, making it easier for users to understand and use. Native development also provides access to the full range of design elements and widgets available on each platform, allowing developers to create interfaces that are tailored to the needs of their users. Not to mention the presence of the accessibility features, that is missing on most multiplatform platforms.

In summary, native mobile development offers several advantages over cross-platform development, including better performance, access to platform-specific features, and the ability to create a consistent and high-quality user interface. While the cost of development may be higher, the benefits make it a worthwhile investment for delivering the best possible user experience.

3.1.3 Development using Game Engine

Developing an information system on a game engine may not be a suitable choice due to several factors. Firstly, the game engine is designed primarily for game development and may not possess the necessary features to support the creation of a responsive design with interface-building capabilities. This lack of built-in tools for interface building could result in additional development time and effort being required to create a suitable user interface. Secondly, networking handling in game engines is typically focused on game development, and may not support communication with REST APIs, which are commonly used in information systems. This would require additional work and potentially result in further delays in the development process. Additionally, game engines are not typically designed with responsive design in mind, and thus may not be able to provide an optimal user experience across different devices and screen sizes. Finally, accessibility features, such as support for screen readers or other assistive technologies, may be lacking in game engines, making it difficult for users with disabilities to interact with the information system. In conclusion, while game engines can be powerful tools for game development, they may not be the most appropriate choice for developing information systems due to their limitations in terms of interface building, networking, responsive design, and accessibility.

3.2 Architecture

The architecture pattern chosen for software development plays a pivotal role in determining the development course of the project. It sets the foundation for the development process and directly impacts the overall quality and maintainability of the software. Therefore, it is essential to carefully consider the architectural pattern that best suits the specific requirements of the project. The choice of architecture pattern can either simplify or complicate the development process, affecting the efficiency and effectiveness of the development process. In this context, this thesis focuses on the analysis of three popular architecture patterns, namely MVVM/MVC, TCA

(The Composable Architecture), and The Clean Architecture, in the context of iOS application development for the virtual guide of Oppidum Závist. The aim is to evaluate the strengths and limitations of each pattern and provide insights into the suitability of each for the specific requirements of the project.

3.2.1 MVVM/MVC

MVC (Model-View-Controller) is a widely used design pattern in software development that separates the application logic into three interconnected components: Model, View, and Controller. It might be as well visible from the diagram 3.1. A controller has the responsibility for passing through users' interactions from view as well as model updates. Adding a loose description of the responsibility network requests could go as in model as well as in controller. This leads to a bigger controller that is harder to manage test and maintain. [14]

The MVVM (Model-View-ViewModel) design pattern is often compared to the widely used MVC pattern due to their similarities. However, MVVM introduces some fundamental differences that distinguish it from MVC. In MVVM, the View is responsible for owning the ViewModel, which acts as an intermediary between the View and the Model. The ViewModel retrieves data from the Model and publishes it to the View, which updates its state accordingly. Conversely, the ViewModel receives input from the View and communicates with the Model to update its state. By separating the View and ViewModel in this way, MVVM enables better separation of concerns and enhances the testability and maintainability of the code.

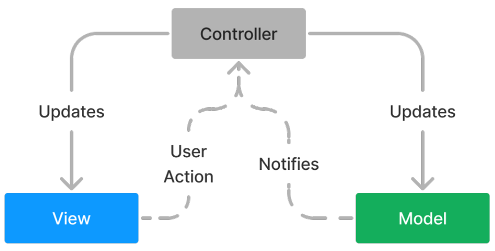


Figure 3.1 MVC design pattern [15]

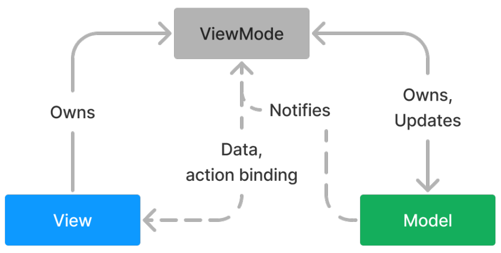


Figure 3.2 MVVM design pattern [15]

3.2.2 The Composable Architecture

TCA or Composable Architecture brings a compositional and functional approach to iOS software development. Independent developers of Point Free created a library that provides a set of tools for developing software differently. At first glance, it might look complicated, but once understood, it is easy to use.

The main component of this pattern is the store, it consists of the connection between State, Reducer and Environment, and each of them has its role. 3.3

State resembles the state of the store, it combines all the data and view states. Important to note that sometimes `store` owns `ViewStore` for the further abstraction of a particular screen state and the application state

Reducer is a pure function, defining the effects caused by the action called from the `View`

Effect is the wrapped publisher user for the data flow.

Action is the enum of all possible *actions* that store handles

Environment is the collection of the dependencies for the store to function properly, there might be the networking or the schedulers.

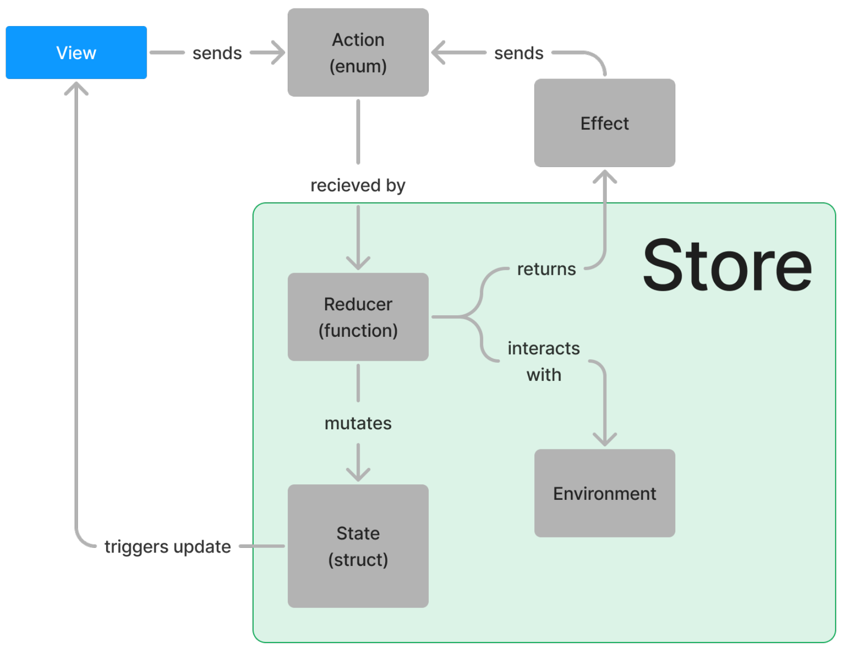


Figure 3.3 TCA design pattern [16]

By separating concerns and modelling the application as a pure function of its state and actions, The Composable Architecture enables developers to build applications that are easier to scale, test, and maintain. One disadvantage is that this framework is still in development, at the moment of writing, the library is still in the beta/unstable development stage. But it is promising technologies that have all the chances to compete with other patterns described in this section.

3.2.3 Clean Architecture

Clean architecture is the most robust of all presented. It provides layers of abstraction, for complete modularity. Coming with the price of generating most of the template code in comparison with other patterns 3.4. On the other side, it is in favour when talking about the cleanness of the code described by Robert C. Martin [17]. Nevertheless, from clean approach mostly benefits applications of great size, with several developer teams developing solutions over a considerable amount of time.

Evaluating three popular architecture patterns, namely MVVM/MVC, TCA (The Composable Architecture), and The Clean Architecture, in the context of iOS application development. MVC separates the application logic into three interconnected components: Model, View, and Controller. However, the MVVM pattern enhances the separation of concerns and enhances the testability and maintainability of the code. TCA provides great separation, easy testing, and

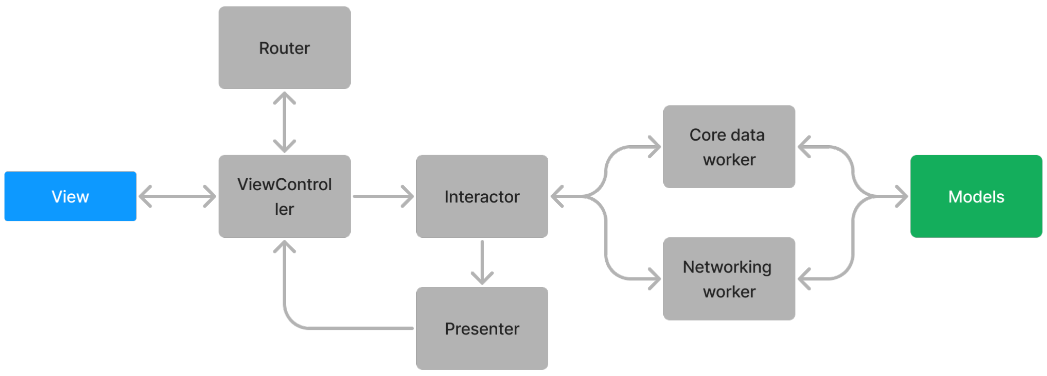


Figure 3.4 Clean Architecture design pattern [18]

maintenance but is still in the beta/unstable development stage. Clean architecture is the most flexible of all and provides layers of abstraction but comes with the cost of an overhead that unnecessarily complicates the development process. MVVM is recommended as it is flexible and provides distinction between the layers, with a relatively small overhead.

Part II

Implementation

Chapter 4

Design and prototype creation

This section walks through the design decisions and prototypes made during the process of getting the best results out of user testing. It covers the various techniques and tools used for designing and creating prototypes, as well as the importance of iteration and collaboration in the design process.

4.1 Prototyping process

4.1.1 Design process

The complete design process was executed in Figma, it is a cloud-based design and prototyping tool that has become the industry standard for many designers due to its ease of use and collaborative features. As such, it was chosen as the primary application for the creation of the mobile app design in this study.

One of the key benefits of using Figma is its ability to create designs and prototypes within a single platform. This eliminates the need for multiple applications and simplifies the design process. Additionally, Figma's collaborative features, such as real-time editing and commenting, allow for efficient teamwork and communication among designers and stakeholders.

When designing mobile applications, creating a flow for the user experience (UX) is critical. Figma provides an intuitive interface for designing UX flows such as one on the image 4.1, enabling designers to create user journeys and wireframes that align with the app's goals. Moreover, Figma's design elements library contains a variety of pre-made icons, buttons, and UI elements that designers can use to speed up their workflow.

In addition to designing UX flows, Figma also allows for the creation of low and high-fidelity prototypes for user testing. Each of these types plays a significant role in different stages of the design process.

Low-fidelity prototypes enable designers to quickly test the functionality and usability of their app design without getting bogged down in details, although the standard method for low-fi prototyping is pen and paper it is proven [19] that computer-based disadvantages prototyping disadvantages are neglectable for the most of the scenarios. As it was determined that drawing was not my strong side, the decision was made to create lo-fi prototypes 4.8 directly in Figma, with all design elements contained within one system. The goal of this prototype was to experiment with different elements leaving the UX as simple as possible, which means minimising the navigation and allowing the user to reach their goal (learn more about a specific part of oppidum) effortlessly.

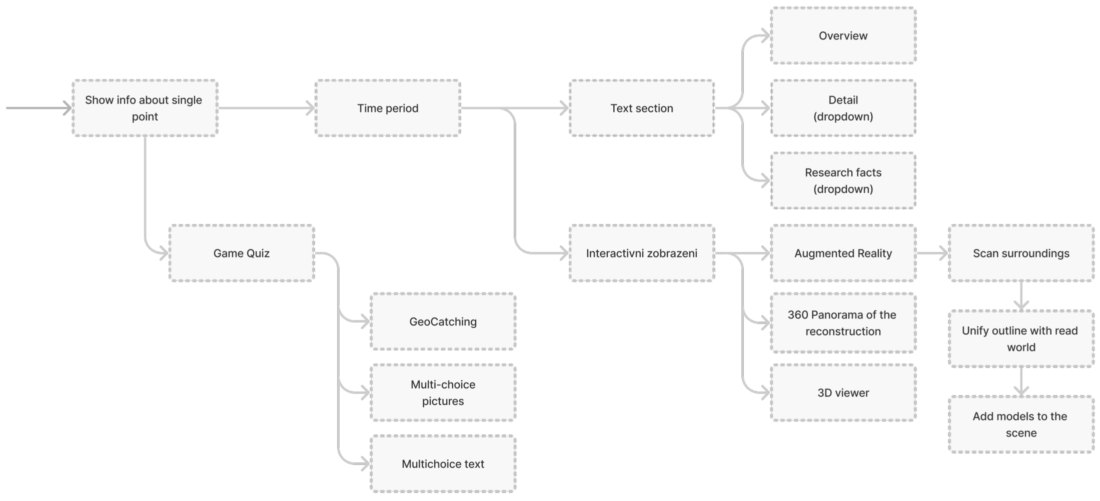


Figure 4.1 Point detail flow

High-fidelity prototypes provide a more polished and refined user experience that closely resembles the final product. While it is proven that low-fidelity prototypes can identify the same amount of usability problems, high-fidelity 4.8 is useful for getting a better vision of how final product, which helps with the communication with the client and other departments working on the project [20]. User testing the prototype should not be instructed and need no additional information for using it.

Prototyping is an essential part of the design creation process that enables designers to identify usability problems and refine the user experience. The creation of low- and high-fidelity prototypes plays a crucial role in different stages of the design process, allowing designers to quickly test the functionality and usability of their design and provide a more polished and refined user experience that closely resembles the final product. While low-fidelity prototypes are useful for experimenting with different design elements and testing functionality, high-fidelity prototypes are useful for getting a better vision of the final product and communicating with the client and other departments working on the project. Although there are various tools for prototyping, Figma has become an industry standard for many designers due to its versatility, ease of use, and collaborative features. Nonetheless, different prototype and creation methods serve their purposes, and designers should choose the most suitable one for their specific design needs.

4.2 Prototype testing

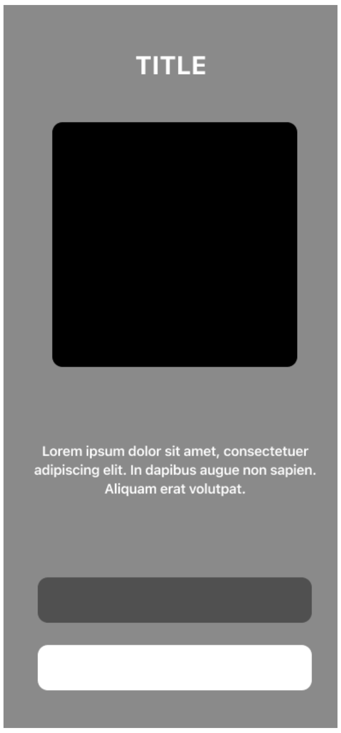
As it was stated in the previous chapter main purpose of the prototype is to build the feeling of the application and identify critical usability issues for faster development in the future. There are several common principles for testing prototypes. As the Figma prototype will be as well implemented and tested by user testing, the design prototype is tested by heuristics methodology.

Heuristics is a set of well-recognized usability design principles used for discovering usability problems [21]. Nevertheless, this set is not stated and may vary depending on the purpose of the evaluation.

For the heuristics evaluation of the Oppidum prototype Nielson Heuristics principles was chosen. It provides a general overview of all aspects of the design prototype. For more information, we will compare the prototype with the existing application to see if the prototype is solving some of the issues of the original solution. Each row of the table 4.1 is evaluating one of the principles stated by Nielsen, Jakob [22]

Table 4.1 Heuristics Evaluation Results

Heuristic	**Prototype Evaluation Result**	**Existing Solution Evaluation Result**
Visibility of system status	Users always know in which state of the app they are and how to navigate forward/backwards	Users may be confused when in the state of the game but the navigation screen is presented as part of the game
Match between the system and the real world	The language and terminology used in the system match the user's mental model, as there is a possibility to change text style to the child version with more suitable texts for the users of different age groups	The language and terminology used in the information section are more suitable for adults whereas the game is focused on the children audience
User control and freedom	The user can undo and redo actions and navigate back to previous pages.	The user can return to the previous page but does not have the freedom to change between sections of the application.
Consistency and standards	The system follows consistent design patterns as well as HIG for the iOS operating system	The design system is fairly consistent but does not follow HIG
Error prevention	The system uses system Alerts according to apples HIG	The system does not have error states, actions that lead to a failing state are immediately rewind.
Recognition rather than recall	The system minimises navigation entries to make elements more accessible	All the instructions are shown to the user in text form and are consistent
Flexibility and efficiency of use	The system is adapting to different user groups through interface and content customisation.	A historical guide is merely efficient despite the format of the text.
Aesthetic and minimalist design	The system is visually appealing and does not contain extraneous information.	System design elements are not minimalistic.
Help users recognize, diagnose, and recover from errors	The system uses native elements to handle the error	The system does not handle users' wrong answers to the game questions.
Help and documentation	The system provides an onboarding flow for users to set up the application for specific use case.	The system provides a help screen for users to learn more about the application.



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In dapibus augue non sapien. Aliquam erat volutpat.

Figure 4.2 Low-fidelity onboarding screen

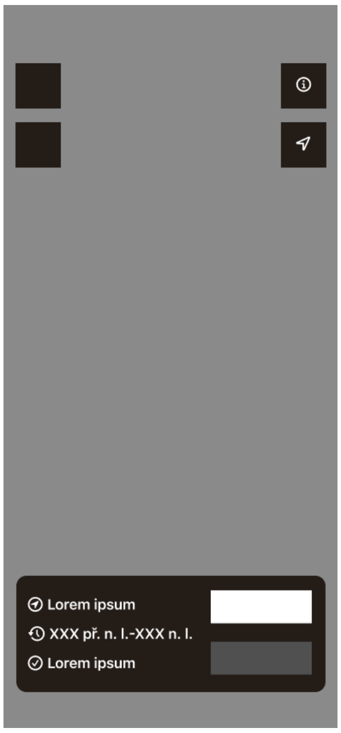
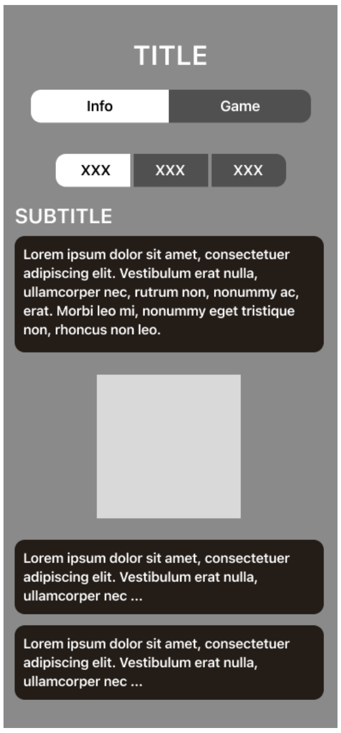


Figure 4.3 Low-fidelity map screen

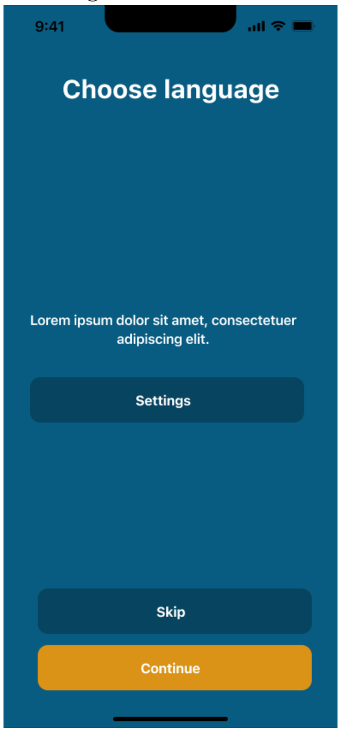


Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum erat nulla, ullamcorper nec, rutrum non, nonummy ac, erat. Morbi leo mi, nonummy eget tristique non, rhoncus non leo.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum erat nulla, ullamcorper nec ...

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum erat nulla, ullamcorper nec ...

Figure 4.4 Low-fidelity detail screen



Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Figure 4.5 High-fidelity onboarding screen

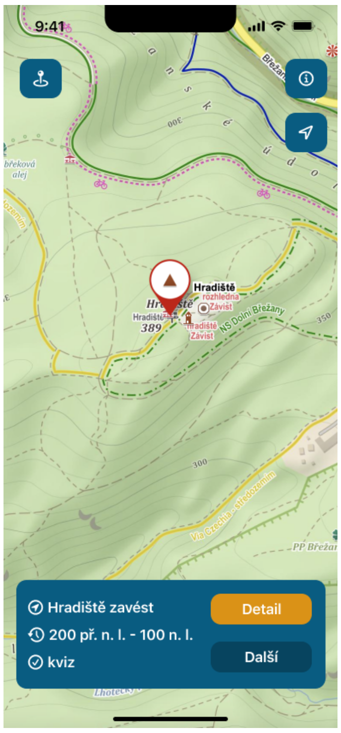
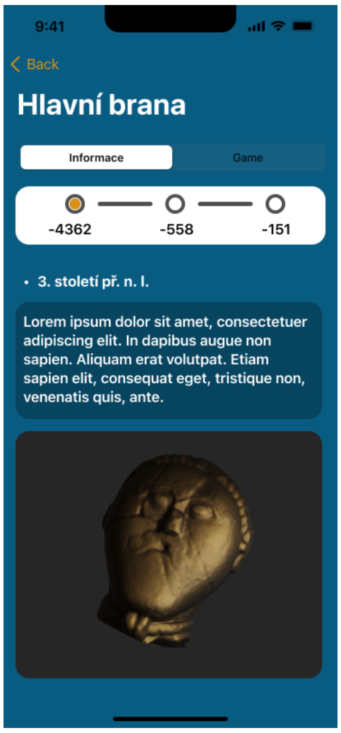


Figure 4.6 High-fidelity map screen



• 3. století př. n. l.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In dapibus augue non sapien. Aliquam erat volutpat. Etiam sapien elit, consequat eget, tristique non, venenatis quis, ante.

Figure 4.7 High-fidelity detail screen

Figure 4.8 Examples of low- and high-fidelity prototypes

Chapter 5

Implementation

This chapter describes the implementation of the prototype for the operating system iOS using Swift programming language. Describing the reasoning behind the usage of each tool and package for the implementation. As well as decisions related to Software Engineering.

5.1 Tooling

5.1.1 Dependency management

App uses **Carthage** as a primary dependency manager, Providing a decentralised solution that builds dependencies while not changing project files leaving the installation process to the user. However, not all dependencies are available to install through Carthage. So the second package manager used is **SPM**. Although this solution is provided by Apple it does not provide as much flexibility as Carthage do. Both of the dependencies managers are integrated using **Tuist Dependencies**. All of the dependencies are defined in *Dependencies.swift* file. For tooling dependencies, **Mint** dependency manager is used, it is providing easy access to the tools that are used for localization.

5.1.2 Dependencies used

The described dependencies are external modules that facilitate the implementation of the application's functionality. It is important to note that tests and strict versioning are crucial when utilizing external dependencies.

The dependencies used in this project:

Alamofire Powerful networking library for iOS and macOS, providing a simple and intuitive API for making network requests. It simplifies the process of handling network code, such as serialization, authentication, and error handling, while also providing advanced features such as response validation.

CTPanoramaView One of the most exciting interactive features of our application is the browser that enables users to reconstruct scenes from the early days of the oppidum. By utilizing this feature, our application creates an incredibly immersive experience for users, providing them with a unique window into the past through the screen of their phone. CTPanoramaView is a sceneKit wrapper that creates 3D scenes with photos wrapped around and uses the phone's gyroscope for recreating camera movement so the panorama follows the user's phone movements.

CodableGeoJSON GeoJSON implementation in Swift, for extending maps with the touristic paths. In our particular case system maps have not provided the level of detail that would simplify user navigation on the side. On the other side, GeoJSON is the accepted standard for extending maps. GeoJSON data is used to add colour-coded paths for users to navigate through the oppidum.

5.1.3 Tools used

Tools simplify development processes. When talking about production applications there are a lot of processes that could be automatized. Each project is unique, and getting the right tooling can save time as well as prolong it setting up the tools that won't be used on the project at their full potential.

ACKLocalization Localisations are a crucial aspect of UX that significantly impact accessibility. By implementing localization tools, users from different regions and language backgrounds can effectively engage with the product or service, providing a more inclusive and personalized experience. This tool enables management of the localisation texts with ease using Google Spreadsheet as the source for the generation of localisation files.

Tuist command line tool which is used to generate, maintain and interact with Xcode projects and dependencies. The Xcode project file has quite a complicated structure and it brings unnecessary complexity. Tuist is used for easier management of the project including modularisation as well as resource and dependency management.

5.1.4 Frameworks used

Thanks to the decision to use native development discussed in 3.1.2 it is possible to use the most optimised solutions for the platform utilising the full potential of the CPU as well as the GPU of the mobile device using frameworks from Apple to build interfaces as well as using Camera and LiDAR sensor for Extended reality.

SwiftUI user interface (UI) framework for building iOS, macOS, watchOS, and tvOS applications. It uses a declarative syntax to create UI elements, which makes it more flexible and easier to use than the traditional UIKit framework. SwiftUI also supports a preview feature, which allows developers to see how their UI will look and behave in real-time, making the development process faster and more efficient [23]. Overall, SwiftUI is a powerful tool for creating modern and intuitive user interfaces for iOS applications.

Combine Any communication with the network requires a way for asynchronous communication in the app. The combined framework from Apple introduces declarative API for processing values over time. Combine follows a reactive programming paradigm, declaring publishers to expose values that can change over time, and subscribers to receive those values from the publishers. [24]

SceneKit Apple Framework for working with 3d scenes, control lighting and geometry objects its position in the scene and rendering options. Thanks to the optimisation done on the Apple side SceneKit utilises the full power of the GPU with a low power consumption of the device it is possible to integrate a 3D model browser to the page seamlessly so the feature is enabled by default and the user does not need to do anything to enable it. [25]

RealityKit Apple Framework for working with Augmented reality. One of the latest Apple Frameworks for forking with Extended Reality. It is a powerful tool for anything connected with placing objects in the real world. Using a native solution like RealityKit enables not only integration of the objects to the real world but also enables interactions with the objects and advanced features such as People Occlusion and instant place detection using LiDAR technologies integrated into some models of iPhone and iPad. [26]

5.2 Implementation of the prototype

In this section Implementation details are discussed. The benefits and principles of using MVVM architecture are described in 3.2.1. Therefore this chapter discusses specific implementation details relevant to iOS native development.

5.2.1 Modularisation

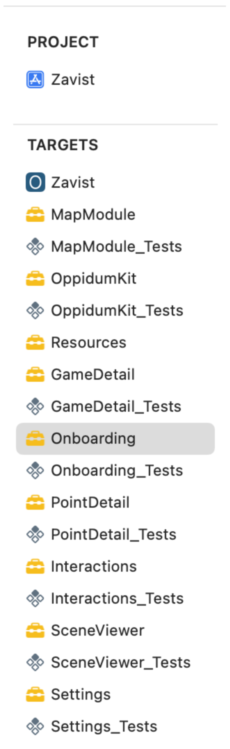
Modularization is an important aspect of iOS development that involves breaking down an app's functionality into smaller, more manageable modules or components. By doing so, developers can achieve several benefits, such as increased code reusability, better structure and organization, and faster compile times. Modularisation of the current app is shown in the image 5.1

One of the main advantages of modularization is that it enables greater code reusability. By breaking an app down into smaller, self-contained modules, developers can create code that is more easily shared and reused across different parts of the app or even across different projects. This can save significant amounts of time and effort, as developers can avoid duplicating code and instead focus on building new features or improving existing ones.

Testing the individual modules is

Another benefit of modularization is that it promotes better overall structure and organization of an app's codebase. By separating different parts of an app's functionality into distinct modules, developers can more easily reason about how the app is put together and how different components interact with each other. This can lead to more maintainable code, as well as easier troubleshooting and bug fixing.

Finally, modularization can also help to speed up compile times. By compiling each module in parallel and continuously in separate threads, developers can reduce the overall time it takes to build and test an app. This can be especially beneficial for larger, more complex apps with many different modules that need to be built and tested together.


PROJECT
Zavist

TARGETS
Zavist
MapModule
MapModule_Tests
OppidumKit
OppidumKit_Tests
Resources
GameDetail
GameDetail_Tests
Onboarding
Onboarding_Tests
PointDetail
PointDetail_Tests
Interactions
Interactions_Tests
SceneViewer
SceneViewer_Tests
Settings
Settings_Tests

Figure 5.1 Project modules inside Xcode

5.2.2 Data layer

On the data layer, separate DTOs (Data Transfer Objects) implement Data models that are received from API (Application Programming Interface). Having DTO written in the Project and mapped to the app structures 5.1 makes the app independent from *Back End* models. For decoding of the models, each DTO implements `Decodable` [27] protocol, from the functional requirements 2.3.1 there is no need for Encodable implementation, for encoding data and sending it back to the server, but the implementation of such functionality is the matter of one changing conformation from `Decodable` to `Codable` [28], that enables encoding and decoding objects. On any changes on the Back End, it is possible to update relevant DTO and its mapping to the models without further editing of the application code.

```
public struct PointDTO: Decodable {
    public let id: String
    public let label: Int
    public let name: String
    public let location: LocationCoordinatesDTO
    public let timeStart: Int
    public let timeEnd: Int
    public let hasGeoGame: Bool
}
public struct LocationCoordinatesDTO: Codable {
   public let longitude: Double
   public let latitude: Double
   public let altitude: Double

   enum CodingKeys: String, CodingKey {
       case longitude = "lon"
       case latitude = "lat"
       case altitude = "alt"
   }
}
extension LocationCoordinatesDTO {
    public var domain: LocationCoordinates {
        .init(longitude: longitude, latitude: latitude)
    }
}
```

Code listing 5.1 Example of the point DTO.

5.2.3 Services

Services are responsible for getting data from API. Since communication with the API is an operation that involves transferring data over the network, the methods used for communicating with the API must be asynchronous to prevent the user interface from becoming stuck during the API method call 5.2.

```
public protocol Networking {
    func fetch<T: Decodable>(_ endpoint: Endpoint) -> AnyPublisher<T, AFError>
    func getURLSession(_ endpoint: Endpoint) -> Session
}

public final class NetworkManager: Networking {
    // GET request using Alamofire returns publisher
    public func fetch<T: Decodable>(_ endpoint: Endpoint) -> AnyPublisher<T, AFError> {
        return sessionManager.request(endpoint.url)
            .publishDecodable(type: T.self)
            .value()
            .eraseToAnyPublisher()
    }
```

Code listing 5.2 Network Manager class managing communication with the API

For asynchronous networking prototype uses `AFResults` from *Alamofire* that is wrapped *Combine* publisher. Provider uses another level of abstraction called `Networking` 5.3 that manages all the request to the network, this way it is possible to change data source fetching the results from another API or even different type of data source such as firebase for example. Providers also stores fetched information in memory, to prevent fetching same data several time decreasing the load on the server.

```
public final class PointProvider: PointProviding {
    public var network: Networking
    public var points: [PointDTO] = []
    public func getPoints() -> AnyPublisher<PointsDTO, AFError> {
        network.fetch(.allPoints)
    }

    public init(network: Networking) {
        self.network = network
    }
}
```

Code listing 5.3 Example of the Point Provider

5.2.4 Views

Views is the key element of the *SwiftUI* and MVVM architecture, it is represented by the any object that conforms to the `View` protocol. Only requirement of the view is to have computed property `body` of type `Some View` meaning that exact type of the view is computed on the compile time, as the `View` type might be hard to distinguish giving the declarative programming style. View is constructed appling the different modifiers to the elements. As the example of the view Map annotation is shown in 5.4, that builds Annotation view for the given location on the map.

```
func MapAnnotationView(location: LocationOverview) -> some View {
    Text(String(location.number))
        // Add padding around the text label
        .padding(12)
        // Set the colour of the background
        .background {
            viewModel.annotationColor(location: location)
        }
        // Set shape to the circle
        .clipShape(Circle())
        // Set on tap action
        .onTapGesture {
            withAnimation {
                viewModel.setChosenIndex(location: location)
            }
        }
        // Increase scale on selected
        .scaleEffect(location.number == (viewModel.chosenLocation?.number ?? -1) ? 1: 0.7)
        // Add animation for the changing of the color and size
        .animation(.default, value: viewModel.chosenPointIndex)
}
```

Code listing 5.4 Example of the Map Annotation view

Chapter 6

Testing

This chapter discusses the testing process of the Oppidum Závist guide application, which was distributed for testing using Apple's TestFlight platform. The chapter emphasizes the importance of usability testing to gain valuable insights into user needs and preferences, and to identify usability problems for further improvement of the application's guide. The chapter presents a detailed description of how to prepare for usability testing. The chapter also discusses the results of usability testing defining areas where the application can be improved to better meet user needs. Lastly unit-testing of the application was discussed, pointing out the importance of the modularisation for the sake of testing units separately. By following the methods outlined in this chapter, developers can ensure that their application is thoroughly tested and ready for release to the public.

6.1 Distribution for testing

In order to distribute an iOS app for testing, Apple's platform for test distribution called TestFlight [29] was used. This platform is designed to simplify the process of testing pre-release versions of iOS apps and is available to developers who have a developer account with Apple.

TestFlight offers two types of testing: internal and external. Internal testing is intended for the testing of pre-release apps within a development team or other authorized group. This type of testing requires the use of an invitation system, where testers are invited via email to download and install the app on their devices.

External testing, on the other hand, allows developers to share their app with a wider audience outside of their development team. This can be useful for gathering feedback from potential users or for beta testing with a larger group. External testing requires the app to go through a review process before it can be distributed, which can take a few days to complete.

Using TestFlight for app distribution provides developers with an efficient and streamlined process, enabling the distribution of versions through CI (Continuous Integration) for testing pre-release versions of their iOS apps. With the ability to conduct both internal and external testing, developers can ensure that their app is tested thoroughly and is ready for release to the public.

6.2 User usability testing

▶ **Definition 6.1.** *Usability testing—the process of learning about users from users by observing them using a product to accomplish specific goals of interest to them. [30]*

Anyone working on the project has their own subjective opinion and evaluating the product's usability solely based on one's opinion may not provide accurate results of the usability tests. To mitigate this issue, usability testing provides an effective means to focus on the user's perspective and their interaction with the product. Obtaining feedback from external sources can gain valuable insights to improve the product's usability. However, to maximize the benefits of usability testing, proper preparation is crucial.

Usability testing on the working prototype of Oppidum Závist provided valuable insights into user needs and preferences, and helped identify usability problems, for further improvement of the guide.

6.2.1 Preparation for the tests

The definition of usability according to the International Organization for Standardization (ISO) emphasizes that a system, product, or service should be evaluated based on its ability to meet the needs of specific users to achieve specific goals with effectiveness, efficiency, and satisfaction within a specific context of use. [31]

The specification of users, goals, and context is important as it determines the scope of usability testing. Testing should be conducted with potential users who are representative of the intended audience, and the goals that the product or system is designed to achieve should be clearly defined. Furthermore, the context in which the product or system is used should also be taken into consideration during testing.

For the conducted study, a group of users was selected to represent personas of adults with varying levels of interest in history and technology. The participants were recruited based on their demographic profiles and were asked to perform a set of predefined scenarios to achieve specific testing goals.

To ensure that the testing goals were clearly defined, scenarios were provided to each user along with step-by-step instructions to help them achieve their objectives. This approach helped to ensure that the testing was conducted consistently and that the results obtained were accurate and reliable.

The simulation of the testing environment was based on the story of a real scenario in which the user would visit the historical site and install the virtual guide application. By providing the users with a realistic context, the testing was conducted under conditions that closely mirrored the actual use of the product. This approach helped to ensure that the feedback obtained from the users was more relevant and reliable.

The study was conducted with a representative group of users, clear testing goals, and a realistic testing environment. This approach helped to ensure that the usability testing was conducted in a systematic and controlled manner, which in turn yielded reliable and valuable insights into the product's usability.

6.2.2 Conducting tests

The tests were conducted in a user-usability lab in CTU FIT equipped with camera equipment, testing devices, and a quiet environment that was non-destructive for the user. The cameras were strategically located to monitor the user's nonverbal communication, as well as their actions during the test.

In the room with the user was a moderator who explained the user's goals and the necessary information for accomplishing those goals. During the testing process, the moderator did not take part, allowing the user to discover the interface and solve problems by themselves. This approach helped to obtain more accurate results of the testing as the user's actions were not influenced by the moderator.

Video and audio from the test were streamed to the usability review room where observers noticed user behaviours and took notes for further evaluation to identify pain points and usability problems 6.2. This approach allowed for real-time observation, helping to identify usability issues and areas of improvement for the product.

After the tests were completed, the user was asked to provide feedback on the product. This feedback was obtained after the user had used the application, giving an overall impression of the product. Gathering feedback in this way gives valuable insights into the user's experience of the product and helps identify areas where the product could be improved to better meet user needs.

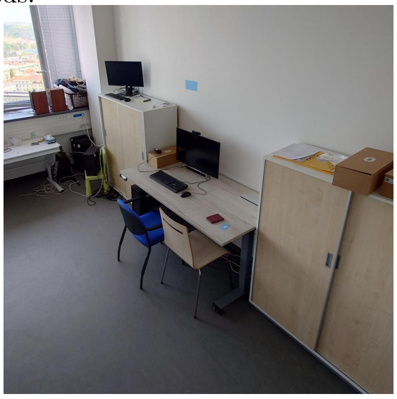


Figure 6.1 UX usability laboratory

Figure 6.2 Usability review room, with the running stream from the usability laboratory

6.2.3 Analysing the results

In the user usability testing, two scenarios were designed to evaluate the effectiveness and efficiency of the application in achieving the specified goals covering the core functionality of the application.

The first scenario focused on the onboarding process and familiarizing the user with the design elements of the application. All participants completed and one issue worth mentioning is changing the language. By design language change is conducted throw system settings, deep linking the user to the language setting in the system settings, after the user is supposed to change back the app manually, requiring the user deeper knowledge of the iOS operating system. This and several other insignificant problems found during this scenario are documented in the table 6.1 along with suggestions for solutions.

The second scenario aimed to test the main goal of the application, which was to provide information about the points of interest around Oppidum Závist. In the first step, users were asked about their preferred navigation app and source of information. They were then asked to find the point of interest by name and explore the interactive elements, such as panoramas and augmented reality. While there were only minor issues encountered during this section of the test, they are documented in the table 6.2.

The second part of the scenario focused on testing the integration of game elements. Several issues were revealed, including a lack of feedback upon completion of the game segment and navigation flow problems. These issues are also described in the table 6.2.

The results of the usability testing provided valuable insights into the user experience of the application and helped identify areas for improvement. The findings from the scenarios will inform the design changes and updates to improve the usability and overall user satisfaction of the virtual guide for Oppidum Závist.

Table 6.1 Result of the usability tests (scenario 1)

Goal to achieve	Timecode	Findings	Possible solution
The first interaction with the application	12:09:10 (user 3)	User trying to swipe tab View, but the panorama is swiped overwise, user uses a button to proceed	Disable touch gestures on panorama, change the type of interaction on the first screen not recognizing gestures
Change the language of the application	11:49:40 (user 2) 12:09:30 (user 3)	Changing the language, opening system settings, switching the application, that puzzles the user on coming back	Change language directly in the application
	11:32:35	Button back from settings gets the user to come back to system settings, not an application	
Find and open application information	12:10:30 (user 3)	Using only system gesture to dismiss(swipe down) may cause the user a hiccup on closing tap	Add button to dismiss or use another style of navigation
	11:32:44 (user 1)	Interactive element of the 3D model is indistinguishable from the photo, users may not see the possibility of interaction	Add icon to all 3d elements or make default animation for a user to see that it is a 3D model

Table 6.2 Result of the usability tests (scenario 2)

Goal to achieve	Timecode	Findings	Possible solution
Map orientation	11:35:35 (user 1)	Settings menu not disappearing while navigating on the map	Add gesture recognition to hide settings on the first tap outside the settings menu
Find the point of interest by name	11:54:10 (user 2)	Labels are numbers, the only way to find them is to go through all the points	Add dynamic labels that add names on Zoom in, a list with an overview of all the points
Open point detail	11:54:30 (user 2)	Definition of buttons "More" and "Next" are not clear.	Change Texts with more describing
Look through all interactions on the point detail	12:14:00 (user 3)	Scroll is interacted by the rotating gestures on the 3D Model, resulting in the user not freely scrolling information	Disable interactions with 3d model by default, the user starts it with the button, adds padding to the 3d model viewer
Answer game question	11:56:00 (user 2)	App does not provide feedback on the correct answer, the user does not know if the question is answered	Provide additional feedback through graphics, haptics or sounds
Find the game progress screen	11:37:41 (user 1)	User expects progress of the game on the screen with the game itself	Provide more links from the game to progress and another way around

In conclusion, the user usability testing conducted in this study provided valuable insights into the design and usability of the virtual guide for Oppidum Závist. By focusing on specific users and goals, simulating the environment, and providing scenarios and instructions, the study revealed usability problems and provided insights into how users interact with the product. The testing was conducted in a controlled environment with cameras to monitor user behaviours, and the observers in the usability review room took notes for further evaluation. Based on the results of the testing, recommendations for improvements were made. Overall, this study highlights the importance of user usability testing in identifying and addressing design and usability issues before releasing the application to the public.

6.3 Unit Testing

Another important part of the software testing process is unit testing. As the name suggests it puts separate units of the program to the test. It involves testing individual modules or components of software code to ensure that they function as intended and that they can be integrated seamlessly into the overall application. This approach to testing offers several benefits over testing the application as a whole. Firstly, it allows for more granular and targeted testing, as each unit can be tested in isolation. This, in turn, makes it easier to identify and fix errors or bugs in the code. Secondly, by testing each unit separately, developers can ensure that the code is modular and easily maintainable. This means that changes or updates can be made to individual units without affecting the rest of the application. Additionally, modularisation allows for faster compilation and testing, as units can be tested independently of one another. Overall, unit testing is a valuable practice that can greatly improve the quality and reliability of software applications.

The application uses a module concept where each feature is separated as described in 5.2. This allows us to create another *test module* for testing *feature modules*. In this module test source file is created and unit tests go here. For a better understanding of the tests, all unit test files follow similar order, having in common the following:

- `XCTest` is the framework providing a set of tools for running unit tests, and performance tests. Providing methods such `XCTAssert` for assertion and `self.measure ...` for testament of the performance
- `testable import` allows access to the internal module implementation, that would not be accessible from another module otherwise.
- method `setUpWithError()` is called before the invocation of each test method in the test class.

One of the Unit tests is presented as the example 6.1. Here the tests are done on the MapViewModel, where separate methods of the viewModel are put to test. Test of the data is possible thanks to the wrapping providers with the protocols described in 5.2.3, which makes the creation and use of the mock data easy.

```swift
import XCTest
...
@testable import MapModule

final class MapViewModel_Tests: XCTestCase {
    var viewModel: MapViewModel!

    override func setUpWithError() throws {
        viewModel = MapViewModel(pointProvider: MockPointProvider())
    }
    func testPointFetchCorrectly() throws {
        XCTAssert(viewModel.points == PointDTO.mockList.domain)
    }

    func testChooseLocation() throws {
        viewModel.setChosenIndex(location: PointDTO.mockList[0].domain)
        XCTAssert(viewModel.chosenLocation == PointDTO.mockList[0].domain)
    }

    func testChooseNextLocation() throws {
        viewModel.setChosenIndex(location: PointDTO.mockList[0].domain)

        for i in (1..<viewModel.points.count) {
            viewModel.chooseNextLocation()
            XCTAssert(viewModel.chosenPointIndex == i)
        }

        viewModel.chooseNextLocation()
        XCTAssert(viewModel.chosenPointIndex == 0)
    }

    func testPerformanceInitialiser() throws {
        self.measure {
            _ = MapViewModel(pointProvider: MockPointProvider())
        }
    }
}
```

Code listing 6.1 Example of MapViewModel unit test.

Conclusion

It takes a lot of effort to make something look effortless.
—Ben Mitchell [32]

This thesis presents a comprehensive description of the development process of the Virtual Guide for the Oppidum Závist, specifically for the iOS operating system. The development goal was to make a performant and informative application using advanced features, keeping the user interface simple and user-friendly.

To develop an optimal solution within the given time and resource constraints, analysis was conducted throughout the research part of the thesis. Analysis of similar solutions given the understanding of the product alternatives its advantages and disadvantages. Domain analysis and writing down the requirements allow for better planning and understanding of the problem application tries to solve. In the overview of the most used frameworks, the native way of development using SwiftUI was chosen as the most appropriate for the problem. Furthermore, through the choices of architectures, MVVM was chosen as the most reliable and comprehensive. With a solid theoretical background in place, the development phase commenced.

The implementation starts with the development of the design. The creation of the Low- and High-Fidelity prototypes were discussed implying further creation and testing using heuristic evaluation methods. The application implementation implies many different tools and frameworks, the usage of which is discussed along with the main implementation principles. The implemented prototype was put to unit tests, testing individual modules separately as well as user usability tests that were conducted in the usability laboratory having the potential users go through the core functionality of the applications. Results of the test are summarised giving the list of the improvements for further development.

Future Development

Future development of the virtual guide prototype for Oppidum Závist offers a promising opportunity to bring the application to a broader audience. With further refinement and expansion, the prototype can be transformed into a production application and released in the application store for iOS, thus enabling visitors to the archaeological site to have a more enriched and immersive experience.

Additionally, the design principles and implementation strategies employed in the development of the virtual guide for Oppidum Závist can serve as a template for the development of virtual guides for other historical sites. The insights gained from the user testing and analysis of similar applications can be applied to future projects, enabling the development of more efficient, user-friendly, and visually appealing virtual guides.

Furthermore, the application can be expanded to include additional historical sites and tourist destinations, creating a comprehensive travel guide application that combines multiple locations. Providing the familiar interface on each location.

Appendix A
Scenarios for user-testing

Two scenarios were designed to test the usability of the guide's core functionality. As the product's target market is the Czech Republic and the tester's majority were Czech it was decided to conduct tests in the Czech language. The first scenario is focused on the user's first interaction with the application. The first page includes a set of step-by-step instructions to complete a specific task within the app, whereas the second page describes user-expected behaviour. Normally the behaviour patterns would be placed inside the table, but taking the small size of the testing (3 testers), separated notes on the tests were enough in our case.

První spouštění

Abstrakt

V tomto scénáře chceme otestovat první interakce uživatele a aplikaci. Hlavním cílem je nastavit aplikaci podle pokynů, následně tyto nastavění změnit.

Kroky

Splníte cíl pomoci následujících kroků

1. Nastavte následující nastaveni:
   (a) Jazyk aplikaci na "Češtinu"
   (b) Zapnete režim aplikací pro dětí
   (c) Vypnete funkcionalitu "Kviz"
2. Najdete a otevřete záložku informaci o aplikaci
3. Najdete a otevřete záložku nastavení
4. Zapnete zpatky funkcionalitu "Kvizu"
5. Vypente režim aplikací pro dětí

Příprava

- Zkoušeme Software ne Vás
- Aplikace je prototyp
- Budu Vás provázet na každém kroku
- Zkuste se uvolnit
- Uživatel přijel na oppidum stahnul aplikaci

Pre-testovaci otazky

1. Jak často jezdíte na výlety do přírody?
2. Používali jste někde operační systém iOS?
3. Používáte-li mobilní aplikaci, v museu?

Očekávané kroky uživatele

1. Nastavte následující nastaveni:
   (a) Jazyk aplikaci na "Češtinu"
   Uživatel přejde do nastavení a změní jazyk aplikaci. Předpoklad uživatel umí ovládat iOS a přejde zpátky, jinak pomoct.
   (b) Zapnete režim aplikací pro dětí
   Uživatel přepne spínač
   (c) Vypnete funkcionalitu "Kviz"
   Uživatel přepne spínač
   (d) Vypente režim aplikací pro dětí
2. Najdete a otevřete záložku informaci o aplikaci
   Uživatel najde označeni pro informaci otevře záložku
3. Zapnete zpatky funkcionalitu "Kviz"
   Uživatel otevře nastaveni a zapne funkcionalitu kvizu

Post-testovaci otazky

1. Je něco co Vám nebylo jasné, na co byste potřebovali víc informaci?
2. Bylo vám jasně ovladani aplikaci

The second scenario is focused on the user's main interaction with the application, using the guide including the interactive features as well as game elements. The first page includes a set of step-by-step instructions to complete a specific task within the app, whereas the second page describes user-expected behaviour.

Přehlednost informaci o zastavení

Abstrakt

V tomto scénáře chceme otestovat aplikace jako Interaktivního průvodce. Hlavním cílem je prozkoumat informaci k zastavení. Následně odpovědět na kvizovou otázku a prozkoumat progres kvizu.

Kroky

Splníte cíl pomoci nasledujicich kroků

1. Najdete na mapě bod s nazvem "Předhradí Závistí"
2. Najdete a otevřete detail bodu
3. Kouknete informaci o zastavení, vyzkoušejte
   - obrazky
   - panoramu
   - 3d model
   - Doplněnou realitu
4. Najdete sekce kvizu
5. Přečtete kvizovou otazku a odpovezte na nej
6. Najdete kde se dá sledovat progres kvizu
7. Najdete na kolik otázek jste zodpověděli

Pre-testovaci otazky

1. Jak moc Vás zajímá historie nebo archeologie?
2. Pokud jste na historickém místě, jaké prvky používáte pro navigaci?
3. Jaké prvky používáte pro nalezení informaci o místě?

Očekované kroky uživatele

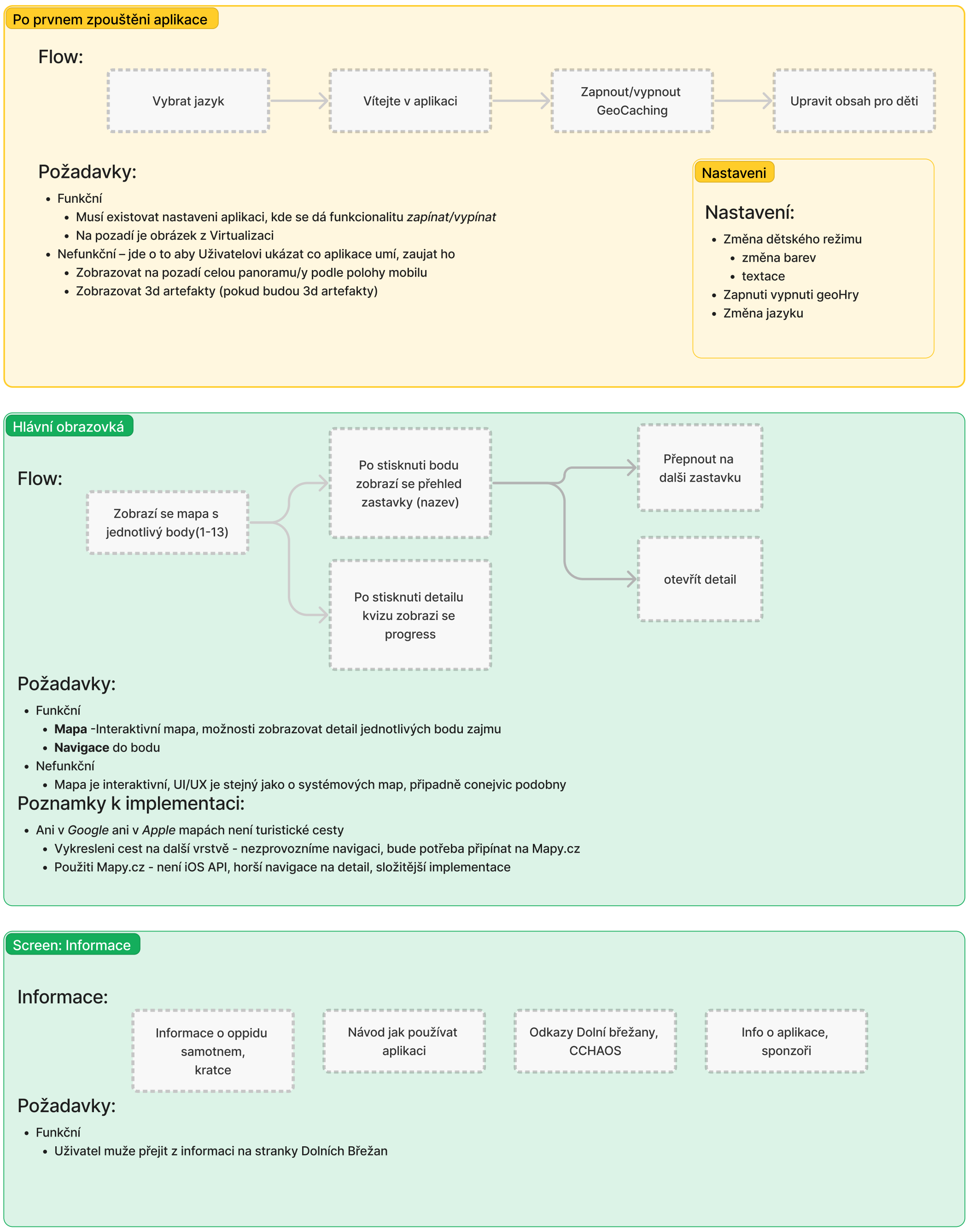
1. Najdete na mapě bod s nazvem "Předhradí Závistí"
   - Uzivatel klikne na nahodny bod
   - Uživatel klikne na označeni sve polohy
   - Uživatel použije tlacitko "další zastavení"
2. Kouknete informaci o zastavení, vyzkoušejte ...
   - Uzivatel pochopi jak funguji panoramy
   - Uzivatel pochopi jak funguji 3d objekty
   - Uzivatel pochopi jak funguje prostředí rozšířené reality
3. Najdete sekce kvizu
   - Uzivatel přepne sekce
4. Přečtete kvizovou otazku a odpovezte na nej
   - Uzivatel pochopí že odpověděl správně/špatně
5. Najdete kde se dá sledovat progress kvizu.
   - Uživatel vrací na mapu
   - Uživatel pozná značeni ikony informaci o kvizu

Post-testovaci otazky

1. Pokud byste používali aplikaci, zapnuli byste funkcionalitu kvizu
2. Pokud byste používali aplikaci, chtěli byste odpovědět na všechny otázky z kvizu?
3. Je něco co Vám nebylo jasné, na co byste potřebovali víc informaci?

Appendix B
Storyboard application flow in Figma

Storyboarding is a useful tool for designing and illustrating the flow of a mobile application's user interface. In this thesis, two storyboard flows have been created in Figma for an iOS application. Taking into consideration that all team members are Czech, StoryBoards was done in the Czech language. The first flow B.1 includes onboarding, a map view, and a settings view, while the second flow includes point detail B.2 and a quiz game. These flows were designed to enhance the user experience of the application by providing a seamless and intuitive navigation path. The following sections provide a detailed description of the two storyboard flows.


Po prvnem zpouštěni aplikace
Flow:
Vybrat jazyk
Vítejte v aplikaci
Zapnout/vypnout GeoCaching
Upravit obsah pro děti
Nastaveni
Hlávní obrazovká
Flow:
Zobrazí se mapa s jednotlivý body(1-13)
Po stisknuti bodu zobrazí se přehled zastavky (nazev)
Po stisknuti detailu kvizu zobrazi se progress
Přepnout na další zastavku
otevřít detail
Screen: Informace
Informace:
Informace o oppidu samotnem, kratce
Návod jak používat aplikaci
Odkazy Dolní břežany, CCHAOS
Info o aplikace, sponzoři

Požadavky:

- Funkční
  - Musí existovat nastaveni aplikaci, kde se dá funkcionalitu *zapínat/vypínat*
  - Na pozadí je obrázek z Virtualizaci
- Nefunkční – jde o to aby Uživatelovi ukázat co aplikace umí, zaujat ho
  - Zobrazovat na pozadí celou panoramu/y podle polohy mobilu
  - Zobrazovat 3d artefakty (pokud budou 3d artefakty)

Nastavení:

- Změna dětského režimu
  - změna barev
  - textace
- Zapnuti vypnuti geoHry
- Změna jazyku

Požadavky:

- Funkční
  - **Mapa** -Interaktivní mapa, možnosti zobrazovat detail jednotlivých bodu zajmu
  - **Navigace** do bodu
- Nefunkční
  - Mapa je interaktivní, UI/UX je stejný jako o systémových map, připadně conejvic podobny

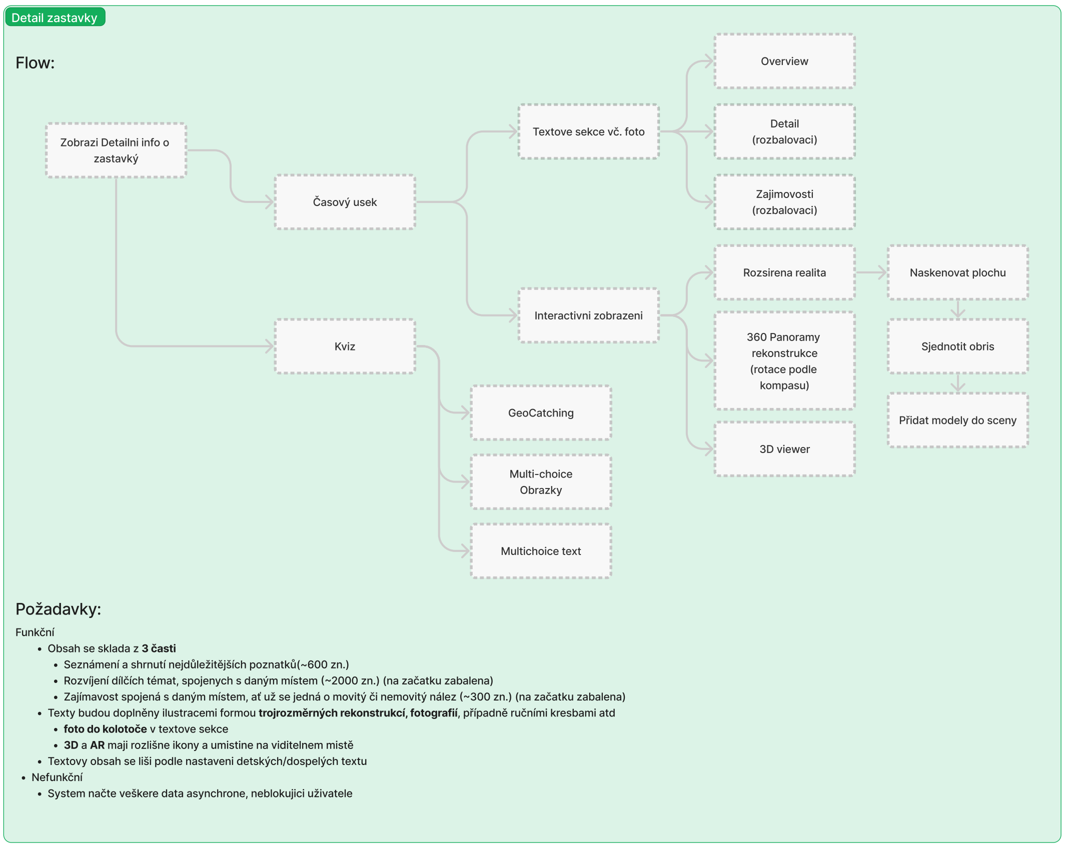
Poznamky k implementaci:

- Ani v *Google* ani v *Apple* mapách není turistické cesty
  - Vykresleni cest na další vrstvě - nezprovozníme navigaci, bude potřeba připínat na Mapy.cz
  - Použiti Mapy.cz - není iOS API, horší navigace na detail, složitější implementace

Požadavky:

- Funkční
  - Uživatel muže přejit z informaci na stranky Dolních Břežan

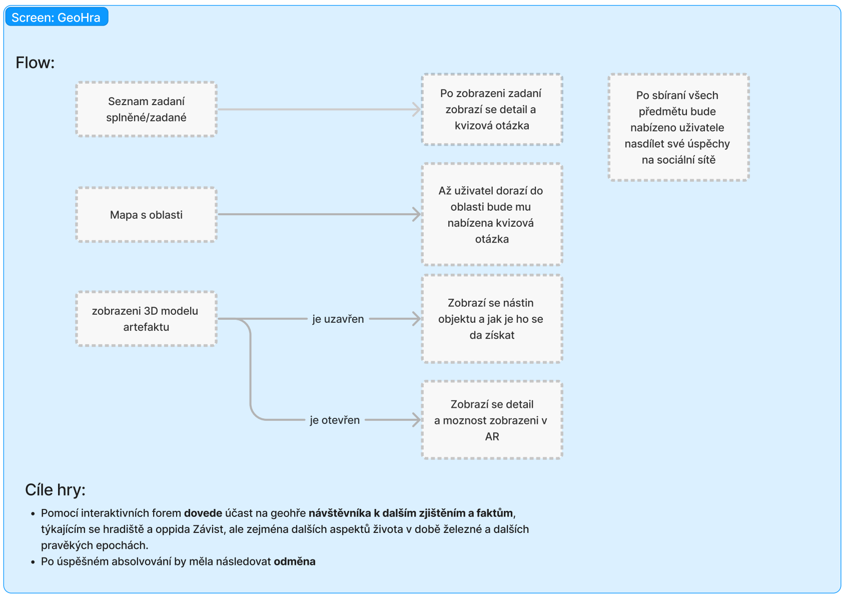
Figure B.1 Storyboard flow diagram with the requirements



Požadavky:

Funkční

- Obsah se sklada z **3 časti**
  - Seznámení a shrnutí nejdůležitějších poznatků(~600 zn.)
  - Rozvíjení dílčích témat, spojenych s daným místem (~2000 zn.) (na začatku zabalena)
  - Zajímavost spojená s daným místem, ať už se jedná o movitý či nemovitý nález (~300 zn.) (na začatku zabalena)
- Texty budou doplněny ilustracemi formou **trojrozměrných rekonstrukcí, fotografií**, případně ručními kresbami atd
  - **foto do kolotoče** v textove sekce
  - **3D** a **AR** maji rozlišne ikony a umistine na viditelnem mistě
- Textovy obsah se liši podle nastaveni detských/dospelých textu
- Nefunkční
  - System načte veškere data asynchrone, neblokujici uživatele


Screen: GeoHra
Flow:
Seznam zadani splněné/zadané
Po zobrazeni zadani zobrazí se detail a kvizová otázka
Po sbírani všech předmětu bude nabízeno uživatele nasdílet své úspěchy na sociální sítě
Mapa s oblasti
Až uživatel dorazí do oblasti bude mu nabízena kvizová otázka
zobrazeni 3D modelu artefaktu
je uzavřen
Zobrazí se nástin objektu a jak je ho se da získat
je otevřen
Zobrazí se detail a moznost zobrazeni v AR

Cíle hry:

- Pomocí interaktivních forem **dovede** účast na geohře **návštěvníka k dalším zjištěním a faktům**, týkajícím se hradiště a oppida Závist, ale zejména dalších aspektů života v době železné a dalších pravěkých epochách.
- Po úspěšném absolvování by měla následovat **odměna**

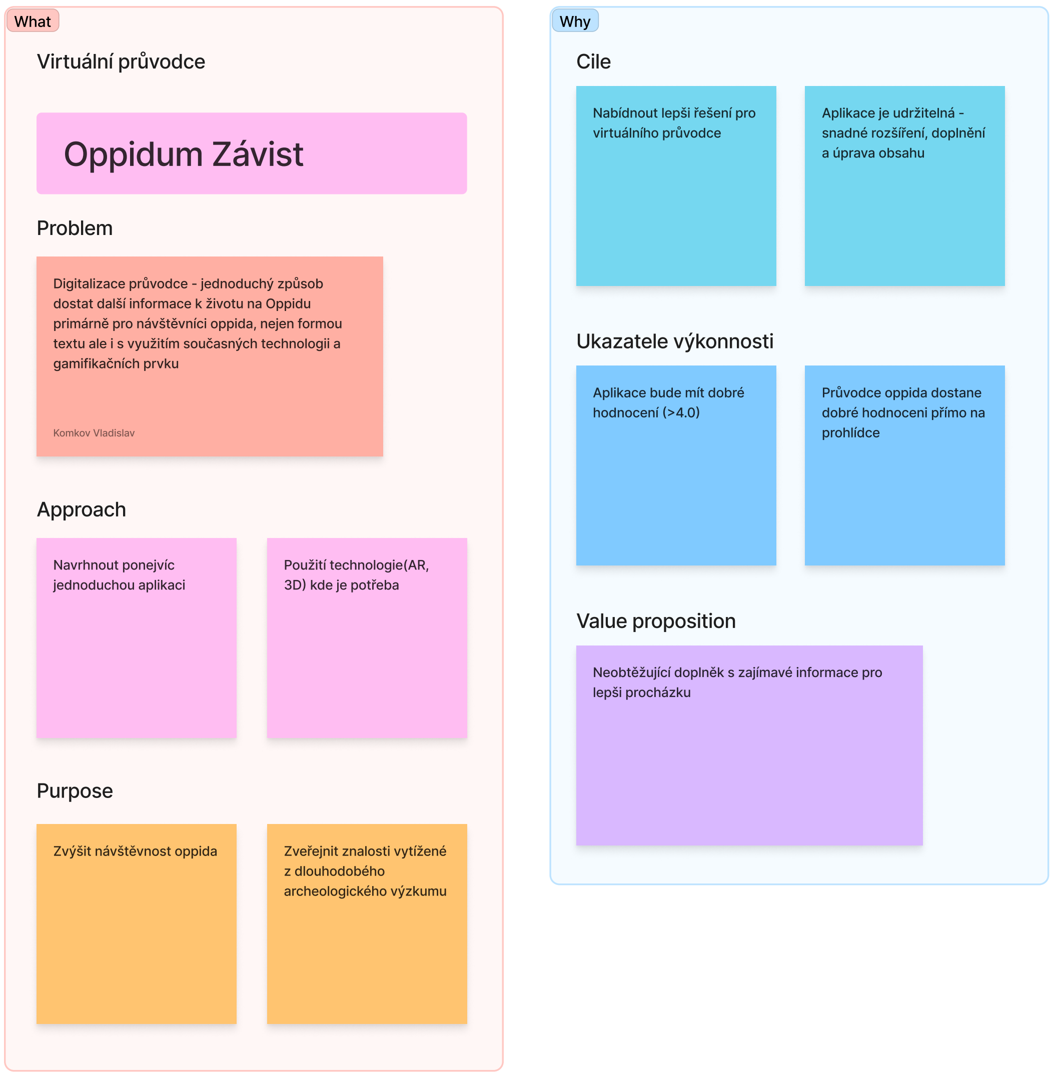
Figure B.2 Storyboard flow diagram with the requirements

Appendix C

Product requirements document in Figma

During the preparation for the development, products requirements were analysed, providing outputs on problematics and goals of the product C.1, and the analysis of the personas main and secondary C.2

The process of creating a product requirements document involves the effort of all team members and stakeholder engagement, considering the communication in the team and between stakeholders are in Czech language, thus the document is conducted in the same language. Figma makes the collaboration between all the members seamless, providing everything needed for the creation of the product requirements.



What

Virtuální průvodce

Oppidum Závist

Problem

Digitalizace průvodce - jednoduchý způsob dostat další informace k životu na Oppidu primárně pro návštěvníci oppida, nejen formou textu ale i s využitím současných technologii a gamifikačních prvku

Komkov Vladislav

Approach

Navrhnout ponejvíc jednoduchou aplikaci

Použití technologie(AR, 3D) kde je potřeba

Purpose

Zvýšit návštěvnost oppida

Zveřejnit znalosti vytížené z dlouhodobého archeologického výzkumu

Why

Cile

Nabídnout lepši řešení pro virtuálního průvodce

Aplikace je udržitelná - snadné rozšíření, doplnění a úprava obsahu

Ukazatele výkonnosti

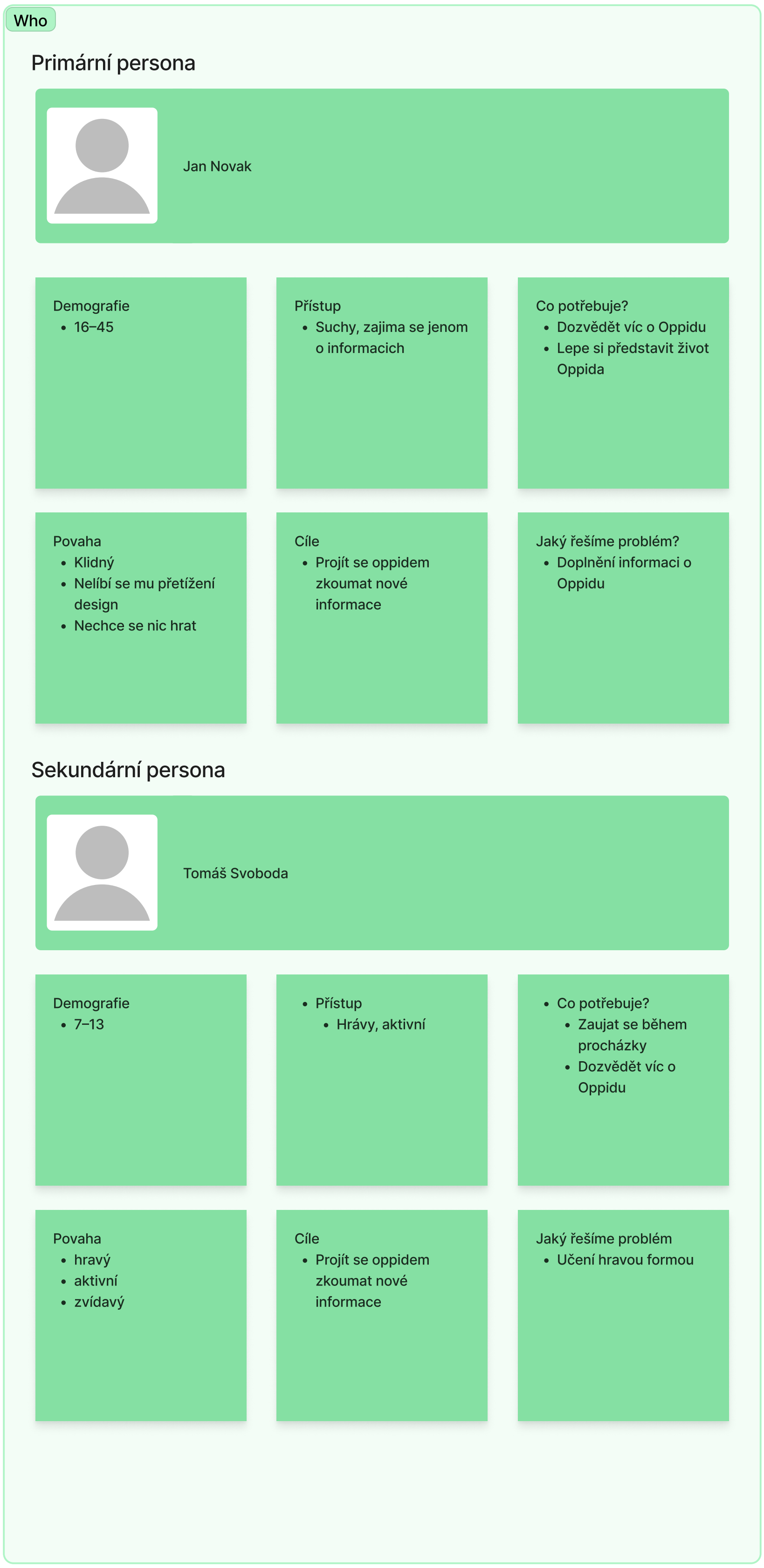
Aplikace bude mít dobré hodnocení (>4.0)

Průvodce oppida dostane dobré hodnoceni přímo na prohlídce

Value proposition

Neobtěžující doplněk s zajímavé informace pro lepši procházku

Figure C.1 Product requirements specification of the problem and the goals



Who

Primární persona

Jan Novak

Demografie
- 16–45

Přístup
- Suchy, zajima se jenom o informacich

Co potřebuje?
- Dozvědět víc o Oppidu
- Lepe si představit život Oppida

Povaha
- Klidný
- Nelíbí se mu přetížení design
- Nechce se nic hrat

Cíle
- Projít se oppidem zkoumat nové informace

Jaký řešíme problém?
- Doplnění informací o Oppidu

Sekundární persona

Tomáš Svoboda

Demografie
- 7–13

- Přístup
  - Hrávy, aktivní

- Co potřebuje?
  - Zaujat se během procházky
  - Dozvědět víc o Oppidu

Povaha
- hravý
- aktivní
- zvídavý

Cíle
- Projít se oppidem zkoumat nové informace

Jaký řešíme problém
- Učení hravou formou

Figure C.2 User research, creation of the personas

Bibliography

1. MATHESIO LTD. *Screenshot of Oppidum Zavist application for iOS* [online]. Mathesio Ltd, 2021. Available also from: `https://apps.apple.com/cz/app/oppidum-z%C3%A1vist/id1525870747?l=cs`.
2. KRUG, Steve. *Don't make me think, revisited: A common sense approach to web usability*. New Riders, 2019. ISBN 9780321965516.
3. APPLE INC. *Human Interface Guidelines* [online]. Apple Inc., 2022. Available also from: `https://developer.apple.com/design/human-interface-guidelines/foundations/layout/`.
4. MORE.IS.MORE LTD. *Screenshot of Visit.More application for iOS* [online]. More.is.More Ltd, 2020. Available also from: `https://apps.apple.com/cz/app/visit-more/id1459887821`.
5. PRIETO-DIAZ, Ruben. Domain Analysis: An Introduction. *SIGSOFT Softw. Eng. Notes*. 1990, vol. 15, no. 2, pp. 47–54. ISSN 0163-5948. Available from DOI: `10.1145/382296.382703`.
6. BURSÁK, Daniel. *Discussion on Oppidum Zavist* [Personal communication]. 2023.
7. WIEGERS, Karl; BEATTY, Joy. *Software Requirements*. 3rd. Microsoft Press, 2013. ISBN 978-0735679665.
8. CHARKAOUI, Salma; ADRAOUI, Zakaria; BENLAHMAR, El Habib. Cross-platform mobile development approaches. In: *2014 Third IEEE International Colloquium in Information Science and Technology (CIST)*. 2014, pp. 188–191. Available from DOI: `10.1109/CIST.2014.7016616`.
9. CORRAL, Luis; SILLITTI, Alberto; SUCCI, Giancarlo. Mobile Multiplatform Development: An Experiment for Performance Analysis. *Procedia Computer Science*. 2012, vol. 10, pp. 736–743. ISSN 1877-0509. Available from DOI: `https://doi.org/10.1016/j.procs.2012.06.094`. ANT 2012 and MobiWIS 2012.
10. JETBRAINS. *Kotlin for Mobile App Development* [online]. JetBrains, 2021. Available also from: `https://kotlinlang.org/lp/mobile/`.
11. JETBRAINS. *Kotlin - Components Stability* [online]. Accessed April 15, 2023. Available also from: `https://kotlinlang.org/docs/components-stability.html`.
12. FACEBOOK. *React Native - Architecture Overview* [online]. Accessed April 15, 2023. Available also from: `https://reactnative.dev/architecture/overview`.
13. FLUTTER TEAM. *Flutter - Frequently Asked Questions* [online]. Accessed April 15, 2023. Available also from: `https://docs.flutter.dev/resources/faq`.

14. ALJAMEA, Mariam; ALKANDARI, Mohammad. MMVMi: A validation model for MVC and MVVM design patterns in iOS applications. *IAENG Int. J. Comput. Sci.* 2018, vol. 45, no. 3, pp. 377–389.

15. ALJAMEA, Mariam; ALKANDARI, Mohammad. MMVMi: A validation model for MVC and MVVM design patterns in iOS applications. *IAENG Int. J. Comput. Sci.* 2018, vol. 45, no. 3, pp. 377–389.

16. SWIFT AND TIPS. *The basics of The Composable Architecture [Video]* [Online; accessed 2023/05/10]. 2023. Available also from: `https://www.youtube.com/watch?v=SfFDj6qT-xg`.

17. MARTIN, Robert C. *Clean Code: A Handbook of Agile Software Craftsmanship.* 1st. Prentice Hall, 2008. ISBN 978-0132350884.

18. LAW, Raymond. *Introducing Clean Swift Architecture (VIP)* [Online; accessed 2023/05/10]. 2017. Available also from: `https://medium.com/hackernoon/introducing-clean-swift-architecture-vip-770a639ad7bf`.

19. SEFELIN, Reinhard; TSCHELIGI, Manfred; GILLER, Verena. Paper Prototyping - What is It Good for? A Comparison of Paper- and Computer-Based Low-Fidelity Prototyping. In: *CHI '03 Extended Abstracts on Human Factors in Computing Systems.* Ft. Lauderdale, Florida, USA: Association for Computing Machinery, 2003, pp. 778–779. CHI EA '03. ISBN 1581136374. Available from DOI: `10.1145/765891.765986`.

20. VIRZI, Robert A; SOKOLOV, Jeffrey L; KARIS, Demetrios. Usability problem identification using both low-and high-fidelity prototypes. In: *Proceedings of the SIGCHI conference on human factors in computing systems.* 1996, pp. 236–243.

21. JIMENEZ, Cristhy; LOZADA, Pablo; ROSAS, Pablo. Usability heuristics: A systematic review. In: *2016 IEEE 11th Colombian Computing Conference (CCC).* 2016, pp. 1–8. Available from DOI: `10.1109/ColumbianCC.2016.7750805`.

22. NIELSEN, Jakob. Ten Usability Heuristics [online]. Accessed April 15, 2023.

23. APPLE INC. *SwiftUI* [Online]. 2021. Available also from: `https://developer.apple.com/xcode/swiftui/`.

24. APPLE INC. *Combine* [Online]. 2021. Available also from: `https://developer.apple.com/documentation/combine/`.

25. APPLE INC. *SceneKit* [Online]. 2021. Available also from: `https://developer.apple.com/documentation/scenekit/`.

26. APPLE INC. *RealityKit* [Online]. 2021. Available also from: `https://developer.apple.com/documentation/realitykit/`.

27. APPLE INC. *Decodable* [online]. 2021. Available also from: `https://developer.apple.com/documentation/swift/decodable`.

28. APPLE INC. *Codable* [online]. 2021. Available also from: `https://developer.apple.com/documentation/swift/Codable`.

29. INC., Apple. *TestFlight* [Online]. 2021. Available also from: `https://developer.apple.com/testflight/`.

30. BARNUM, Carol M. Introduction: Getting started guide. In: BARNUM, Carol M. (ed.). *Usability Testing Essentials.* Boston: Morgan Kaufmann, 2011, pp. 1–7. ISBN 978-0-12-375092-1. Available from DOI: `https://doi.org/10.1016/B978-0-12-375092-1.00012-X`.

31. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems* [Online]. ISO, 2010. Available also from: `https://www.iso.org/standard/52075.html`.

32. MITCHELL, Ben. *Quote: It takes a lot of effort to make something look effortless* [Online]. 2023. Available also from: `https://www.goodreads.com/quotes/1612992-it-takes-a-lot-of-effort-to-make-something-look`.

Content of the attached media

```
README.md ........................ overview of the thesis, resources and installation guide
├─ src ............................................... folder with the sources files
├─ src
│  ├─ oppidum ........................................... sources for the application
│  └─ thesis ................................. sources of the thesis in LaTeX format
└─ text ......................................................... text of the thesis
   └─ ctufit-thesis.pdf ........................... text of the thesis in PDF format
```