



**CZECH TECHNICAL
UNIVERSITY
IN PRAGUE**

F3

**Faculty of Electrical Engineering
Department of Computer Science**

Master's Thesis

Information system for presentation of hotel services and activities in surroundings as a part of the onboarding process

Jan Veselý

Open Informatics - Software Engineering

June 2022

Supervisor: Ing. Jiří Šebek

Acknowledgement / Declaration

I would like to thank my supervisor Ing. Jiří Šebek for helping me by giving me valuable advice in the course of writing this thesis and my parents who helped me conduct the user testing.

I hereby declare that I have completed this thesis on my own and that all the used sources are included in the list of references, in accordance with the *Methodological instructions on ethical principles in the preparation of university theses*.

In Prague, June 19, 2022

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s *Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací*.

V Praze dne 19. 05. 2022

.....

Abstrakt / Abstract

Tato diplomová práce se zabývá analýzou, jak efektivně uvést hosty na jejich dovolenou pomocí dotykového kiosku a online webové stránky obsahující cestovní informace o daném místě. Definuje nejvýznamnější skupiny uživatelů z penzion U Veselých, jejich hlavní zájmy a způsob, jakým jim může hoteliér poskytnout potřebné informace pomocí elektronického uváděcího systému.

Dvě uživatelské studie, ověřily potřebu tohoto systému. Ukázaly, že hosté pravděpodobně využijí elektronický kiosk a uvítací web k získání cestovních informací.

Dále práce navrhuje implementaci minimum viable product pro uvítací systém. Zahrnuje kiosk s dotykovou obrazovkou poháněný Raspberry Pi a webovou stránku.

Klíčová slova: cestovní informace, hotelový uvítací systém, dotykový informační kiosk, cestovní informační weby

Překlad titulu: Informační systém pro prezentaci služeb hotelu a aktivit v okolí v rámci pobytu a objednávky

This diploma thesis analyzes how to onboard guests efficiently on their holiday using an on-site touch screen kiosk and an online website containing travel information about the location. It defines the most significant user groups from the hotel Penzion U Veselých, their main interests, and how the hotelier could provide the needed information using the electronic onboarding system.

Two user studies we conducted to verify the need for the product. They showed that guests are likely to use the electronic kiosk and the onboarding website to gain travel information.

Further, the thesis describes the implementation of a minimum viable product for the onboarding system. It includes an on-site touch screen kiosk powered by Raspberry Pi and a website.

Keywords: travel information, hotel guest onboarding, electronic touch-screen kiosk, travel websites

Contents /

1 Introduction	1	6 Available Technology	17
1.1 Benefits	1	6.1 Web Frameworks	17
1.2 Thesis Goals	1	6.1.1 Spring Boot	17
2 Product Design	2	6.1.2 Django	17
2.1 Lean Startup Methodology	2	6.1.3 Ktor	18
2.1.1 Build-Measure-Learn Feedback Loop	2	6.2 Database Storage Engine	18
2.1.2 Minimum Viable Product	2	6.2.1 Scalability	18
2.2 Our Hypothesis	2	6.2.2 Document Databases	19
3 Collection of Requirements	3	6.2.3 Relational Databases	19
3.1 Data Source	3	6.3 Web Frontend Architecture	20
3.2 Target Audience	3	6.3.1 Server Rendered	20
3.3 Travel Platforms	4	6.3.2 Client Rendered	20
3.3.1 Booking.com	4	6.3.3 Hybrid Approach	21
3.3.2 Trip Advisor	5	6.3.4 React	21
3.3.3 Turistika.cz, Kuduznudy.cz	5	6.3.5 Vue	21
3.4 Trends in Hospitality	5	6.4 API Design	21
3.5 Future of Travel Platforms	6	6.4.1 REST	22
3.6 Guest Satisfaction	6	6.4.2 GraphQL	23
3.6.1 Accommodation	6	7 Backend Architecture	24
3.6.2 Holiday Experience	6	7.1 Requirements	24
3.7 Similar Products	7	7.2 Possible Approaches	24
3.7.1 Previo Alfred	7	7.2.1 Event-Driven	25
3.7.2 MyStay	8	7.2.2 Microservices	25
3.8 Related Systems	8	7.2.3 Monolith	25
3.8.1 Hotel Website	8	7.3 Monolithic Approach	26
3.8.2 Booking.com	8	7.4 Layered Architecture	26
3.9 Check-in Process	9	7.4.1 Data Layer	27
3.9.1 Before Arrival	9	7.4.2 Data Access Layer	28
3.9.2 After Arrival	9	7.4.3 GraphQL API	28
4 Formulation of Requirements	10	7.4.4 HTTP Layer	29
4.1 Onboarding Microsite	10	8 Implementation Details	30
4.1.1 Functional Requirements	10	8.1 Backend	30
4.1.2 Nonfunctional Re- quirements	11	8.1.1 GraphQL Entity Mapping	30
4.2 On-site Kiosk	11	8.1.2 GraphQL Authentication	31
4.2.1 Functional Requirements	11	8.1.3 Input Validation	31
4.2.2 Nonfunctional Re- quirements	11	8.1.4 Storing Images	32
4.3 User Types	11	8.2 Multiplatform client	32
4.4 ER Model	11	8.2.1 Kotlin Multiplatform	33
5 Product Idea Validation	13	8.2.2 Kotlin Symbol Processing	33
5.1 Designing User Experiments	13	8.2.3 Compiling DTOs	33
5.2 Kiosk User Testing	14	8.2.4 Client Library	34
5.3 Info Website User testing	15	8.3 Frontend	34
		8.3.1 State Management	34
		8.3.2 Navigation	35
		8.3.3 Onboarding Screen	37

8.3.4	Trip Screen	38
8.3.5	Touchscreen Kiosk	39
9	Kiosk - Hardware / Software	40
9.0.1	Screen	40
9.0.2	Computing Unit	40
9.1	Operating System	40
9.1.1	Touch Screen Correction	40
9.1.2	Autorun Configuration	41
9.2	Cost of Operation	42
10	Deployment	43
10.1	Pipeline	43
10.1.1	Stages	43
10.2	Docker Images	44
10.2.1	Vue frontend	44
10.2.2	Ktor Backend	44
10.3	Docker compose	45
10.4	Reverse Proxy	45
11	Testing	46
11.1	Test Goals	46
11.2	Backend Testing	46
11.3	Frontend Testing	48
11.4	Small Usability Testing	49
11.4.1	Survey	49
11.4.2	Results	49
11.5	Testing Summary	50
12	Project Future	51
13	Conclusion	52
	References	53
A	Thesis Assignment	57
B	Screenshots	59

/ Figures

3.1	Previo Key application.	7
3.2	MyStay application.	8
4.1	Onboarding process diagram ..	10
4.2	ER model.	12
7.1	Layered architecture diagram..	27
7.3	DAO objects.	28
8.1	Defining GraphQL entity from DB entity	30
8.2	Extending data fetcher with authentication	31
8.3	Validating multiplatform en- tity	32
8.4	Image persistence diagram.....	32
8.5	Kotlin data class to Type- Script compilation.	34
8.6	Multiplatform client action example	34
8.7	State management of Edit- TripDialog	35
8.8	HTA diagram of user interface .	36
8.9	Onboarding page screen	37
8.10	Trip popup Screen	38
8.11	Picture of the kiosk	39
10.1	Deployment diagram.	43
10.2	Dockerfile for Vue	44
10.3	Dockerfile for Ktor	44
10.4	Service Deployment diagram ..	45
11.1	Test coverage of layered ar- chitecture	46
11.2	Code coverage table.	47
11.3	Using Multiplatform client for testing	48
11.4	Cypress test	48
B.1	Screen edit trip popup.	59
B.2	Screen edit hotel settings	60

Chapter 1

Introduction

Tourism was worth 2.5% of the Czech Republic's GDP in the pre-pandemic year 2019. Around 240,000 people are permanently employed in this sector, which is 4.41% of the total employment[1].

According to the American Hotel and Lodging Association 2022 report section, future trends „The personalization of technology will take another leap forward, with hotels using digital technologies to ease workloads and further satisfy each individual guest with a new guest experience. As a result, hotel employees are spending less time on tasks, such as processing check-ins, and can pursue initiatives that can make a greater impact on customer service.“[2]

Going on holiday has significant positive social and health implications. However, this is only applicable if the guests feel satisfied on holiday. Holiday planning is a complex task which connects multiple scattered sources, often resulting in guests visiting just the most popular places, leading to overtourism and little satisfaction.

This thesis analyses the most common sources of travel information and proposes a minimum viable product which connects hospitality providers as a source of most valid travel information and guests via an online onboarding website and on-site touch screen kiosk.

1.1 Benefits

- Hotel guests will receive more comprehensive information about the location through an on-site kiosk and an online onboarding website.
- Decreased time needed to serve guests during rush hours.

1.2 Thesis Goals

- Research how to ideally inform guests about local travel tips.
- Propose and implement a minimum viable product which informs guests about local travel tips.

Chapter 2

Product Design

This chapter establishes the basic terminology and methodology for developing a new product. The terminology is built around the uncertainty of whether the product will be viable from the business point of view. It describes how to refine and test an idea.

2.1 Lean Startup Methodology

The lean startup is becoming¹ the new standard when refining and generating new ideas and insights for business. It challenges the usual waterfall way of designing a product.

We will focus on technique called Concierge MVP. It is one of three recommended ways to gain validated learning from the Lean Startup methodology. It states that one of the ways to gain validated learning is to build a product just for one customer a learn from his needs. In our case, it is Penzion U Veselých.

2.1.1 Build-Measure-Learn Feedback Loop

The Build-Measure-Learn Feedback Loop is a continuous quality improvement model consisting of 3 key stages: Build, Measure, Learn. It is the key to gaining validated learning. First, a hypothesis is proposed, a product is developed, and finally, validated learning is derived from the product's usage. This inside is then used as a foundation for another loop iteration, which might not even connect to the previous iteration, to learn what customers want and makes a profit.

2.1.2 Minimum Viable Product

The Minimum Viable Product is defined in Lean Startup by Eric Ries as „The MVP is that version of the product that enables a full turn of the Build-Measure-Learn loop with a minimum amount of effort and the least amount of development time. The minimum viable product lacks many features that may prove essential later on. However, in some ways, creating a MVP requires extra work: we must be able to measure its impact. For example, it is inadequate to build a prototype that is evaluated solely for internal quality by engineers and designers. We also need to get it in front of potential customers to gauge their reactions.“[3]

2.2 Our Hypothesis

Will guests use the hotel's electronic travel information system? If so, will that lead to decreased time to onboard and serve the guests.

¹ <https://hbr.org/2013/05/why-the-lean-start-up-changes-everything> Why the Lean Start-Up Changes Everything

Chapter 3

Collection of Requirements

The collection of requirements is essential for product design and its usefulness verification. In this chapter, we will look at products in a similar category as well as raw data from Penzion U Veselých.

Our goal is to formulate what information guests need for their holiday and find how to present the data in a way that will be useful to the guests.

3.1 Data Source

Most of the raw data and personal experience used in this thesis come from Penzion U Veselých, a family-run hotel in Pec pod Sněžkou.

Author's note: Our family has been running a small family hotel in Pec pod Sněžkou for over 25 years. Our long experience serving guests directly without any staff gives us an excellent empirical image of what guests might need. This information was not scientifically measured and is biased [4].

3.2 Target Audience

The research is limited only to domestic tourism. For simplicity, we will only consider summer holidays since, in the winter, the offer is generally directed towards skiing, and other activities are only additional. Guests can be divided into several distinct categories based on interviews with guests.

- **Relaxation stays** regardless of age. These visitors arrive at a place with the primary goal of releasing stress and leaving their city. The length of stay is typically a weekend. They prefer a maximally undemanding walk around the very close surroundings and wellness or massage services in the evening. These visitors make up a small part of the total number of guests. However, the weight of this category is slowly increasing.
- **Young couples** age typically under 30, prefer to stay active and in nature (hiking, biking, but also non-traditional sports and experiences - paragliding). At the same time, they are not afraid to go on longer hikes or drive further for some exciting destination, which they usually learn about at the place of stay, usually from the hotel staff. For example, hikes to the nearby highest peaks at sunrise are very popular in this category. Length of stay is typically 2-3 nights. In the evening, they prefer more private events such as wellness and dinner for two in a more luxurious environment.
- **Older couples** prefer to relax with shorter trips in the surroundings, especially in places with scenic views. They use cable cars to get around mountain peaks and other public transport. They gratefully receive information directly at the stay site, either from the hotel or they look for it in the information centres, because it is usually more difficult for them to organize their stay using the Internet. The length of stay is typically four nights. These visitors would probably most welcome a service

in the accommodation facility, which would offer them several proven and exciting variants of visit, whether in written or digital form. They seek peace and relaxation in the afternoon, sitting on the terrace with coffee or good wine.

- **Parents with small children** (typically up to 10 years old) are mainly looking for suitable attractions for children to whom they subject the whole holiday. They especially prefer outdoor attractions for children and short trips using cable cars. They like to guide children on educational, game-like trails. They usually visit various natural, cultural or technical attractions with the help of a car. They are very grateful to accept a variety of ideas for their holiday. For example, finding a sandpit or playground near the building or a path suitable for a stroller is often utterly impossible on the Internet. The length of stay is typically five nights. They seek peace and relaxation in the afternoon after a busy day with their children, sitting on the terrace with coffee or good wine.

There is one common phenomenon for all the above categories. Whether the decision about going on holiday is long-term or impulsive, almost everyone lacks whatever plan to do in a given place. When talking to them, there is an impression that they are going to a place mainly because it is known and not to visit something interesting. The visits are usually limited to some of the most famous destinations, such as Sněžka. They usually look for travel ideas on the spot, with the help of the Internet, less so by visiting the info centres. Penzion U Veselých provides them with handmade sets of the most exciting places to visit sorted and organized by categories: for children, in bad weather, with a car... We must spend approximately 30 minutes of talk to each guest to onboard them on their holiday. When searching the Internet, guests do not do extensive research to find less popular activities that are still exciting, which might be better for them due to over-tourism in some places.

Travel platforms focus primarily on accommodation offers. Tips for trips are not there. There is a complete lack of a more comprehensive view of the location. Info centres provide mainly information leaflets. The last source for visitors is the cities' websites, where there are usually only tips on what to do in the town.

As a result, visitors have many sources of highly fragmented information. Despite the lack of preparation, guests still expect all this information to be provided by the hotelier. There is a gap in the market for software that would provide guests with organized information about the local places from locals.

3.3 Travel Platforms

Travel platforms have gained massive popularity in recent years due to accessibility to search and order services without the need to contact the hospitality provider directly and agree on dates and services. From experience in Penzion U Veselých, it takes 1.5 to 3 days to communicate the order with all details via email or phone.

In most cases, the price guests pay for the comfort of booking platforms is unacceptable for them, given it was explained to them how much money they could have saved.

3.3.1 Booking.com

Booking.com is the number one online travel agency for lodging reservations, with its market share of over 67%^[5].

Booking.com launched in 2019 attractions feature, which is very similar to the trip advisor. Unfortunately, there are no free of charge activities, and it focuses primarily on leisure and fun in big cities. It is not suitable for countryside holiday information.

■ 3.3.2 Trip Advisor

Tripadvisor is an American online travel company that runs a mobile application and website with user-generated content. It offers online hotel reservations, transportation, travel experiences, restaurant reviews and recommendations.

As an international company, they benefit from a considerable user base. According to Google play, more than 100 000 000 users downloaded the TripAdvisor application. The content is more focused on tour guide content and tickets. It lacks non-tour, non-payable content such as nature wonders, hiking information, and local information about the place.

■ 3.3.3 Turistika.cz, Kuduznudy.cz

Turistika.cz, Kuduznudy.cz travel advice sites mainly focused on local travel tips, event aggregation and search. They inform guests what they can do and point them to the activities' official websites. Over the years, they have collected a massive amount of travel tips. The content is well written and very informative. The site has a massive audience. According to their website, [6] kudyznudy.cz displays more than 20 million activities per year.

However, the content is mainly organized by region. It lacks local information the guest needs, such as recommendations sets based on a location.

Kudyznudy.cz has a mobile application. There are no better search alternatives in the application than on the website, and overall the application is not suited to obtain any local information quickly, which is unfortunate.

■ 3.4 Trends in Hospitality

Tourism is an important economic activity at the national and regional level. In 2019, this branch accounted for 2.9 % of the Czech Republic's gross domestic product [1] . According to Accenture's 2021 US Holiday Shopping Survey, approximately 40 % of US consumers intend to focus on saving for a vacation [2]

- **Contactless hospitality services:** Especially in the aftermath of the 2022 pandemic of covid-19, guest expectations in health safety boosted demand for contactless services on all levels of hospitality services from check-in through using hotel services, searching upsell services to the check-out and payments. Many platforms and hospitality providers have embraced this model and started to automate information distribution through electronic channels such as email or the web, trying to fill in the decline of orders in search of other ways to connect with customers.
- **Information Quality:** The quality of information on hotel websites has increased. Guests searched for relevant and up to date information regarding travel rules which led the hospitality providers to update their web content regularly, as observed in Pec pod Sněžkou resort.
- **Travel Advice:** Guests are less likely to ask local staff for travel advice. The fear of covid-19 has significantly reduced social interaction between guests and staff. From the perspective of small businesses, it is a good thing since their time is spent mainly on the hotel operation, thus allowing them to finish sooner.

- **Online Booking:** The number of online bookings has expanded immensely. Due to uncertainty of travel rules, such as the number of vaccination doses, guests become much more flexible and reserve a holiday within a few days from arrival. Such a short frame of time further reduces the amount of information the guest is willing to search about the destination.

3.5 Future of Travel Platforms

The fundamental problem of aggregation servers from the perspective of holiday guests is their focus on what sells (accommodation), not the guest's experience. Providing guests with up to date and relevant information is highly labour intensive. It can be only achieved by working with someone on the spot, such as hotels or tour guides.

There is a possibility of a new business model which would involve hotels producing travel information in exchange for better online search positions. The travel information would bring more customers to the platform. The customers would buy accommodation, thus generating money for the platform.

The accommodation providers themselves seem to be the most suitable, not the cities, as they meet the guests directly, receive many questions, and have very well-founded answers. They know the location the best and can, based on their personal experience, recommend or not recommend a trip or place to visit.

3.6 Guest Satisfaction

Examining guest satisfaction can give us attributes we should focus on when designing our system. Guest satisfaction is based on two main factors accommodation and experience.

3.6.1 Accommodation

Accommodation is one of the core factors in guest satisfaction. It is also the entry point for a holiday. The critical problem discovered through the 25 years old experience working with guests is false but anticipated expectations of the hotel and its services. Guests are often unaware of what they have ordered or what services to expect to be free or paid.

Online travel agencies make this problem even harder. All the communication with the customer is indirect and often censored (Booking.com messaging system) to prevent the direct ordering of the stay on the hotel's website. They present themselves as the only source of hotel/travel information. The guests are not prompted to explore the hotel's official website, where more helpful information is available.

Should the hotels improve the satisfaction convertibility - The number of satisfied guests to the total number of guests - they need to improve the quality and the amount of information the guests are presented. Too much information is counterproductive as well.

3.6.2 Holiday Experience

The secondary source of guest satisfaction is what activities they do. This parameter is hard to grasp since it varies depending on the guest category, weather, and other conditions. However, we can say that not fulfilling this category has a tremendous impact on guest satisfaction.

As discussed in section 3.3, visitors have many sources of highly fragmented information about the place in the best-case scenario. In most cases, they go to the most popular places even though there are better options for trips.

3.7 Similar Products

3.7.1 Previo Alfred

Previo is a major provider of hotel management systems, from booking management to accounting. It integrates almost all main hotel aggregation sites. They present themselves as all in one system, and in 2021, they added a mobile check-in app to their portfolio.

Its main goal is to help check in the guests, but it also has useful features to inform guests what they can do on the spot. Unfortunately, hoteliers cannot integrate the app without opting into Previo ecosystem. The interconnection is the main drawback for our use in Penzion U Veselých since we do not use the Previo booking management system.

The application offers all the basic needs the guest could want: information about the location in article form, room services, and the ability to order hotel service throughout the check-in process.

The Previo Klíč application has over 1000 downloads in the Google Play store[7]. That is not many compared to that Previo serving 3 432 Hotels as of 2022, and the app was last updated in late 2020. The convergence is relatively small - the number of total guests to guests who used the app is small. The following picture 3.1 shows the application.

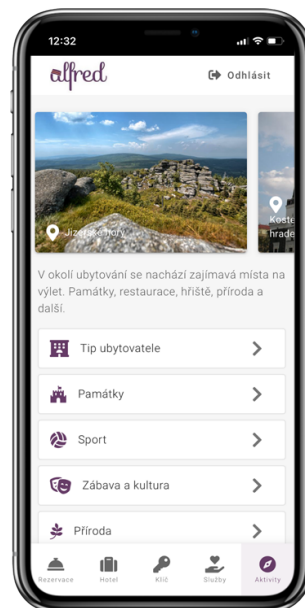


Figure 3.1. Previo Key application interface

3.7.2 MyStay

MyStay¹ is a Czech Startup offering a fully digitalized check-in service for hotels with more than a dozen integrations to other services such as Booking.com Mews. Its main targets are more prominent hotel resorts or B&B services.

Their check-in process includes pre-arrival SMS, weather forecast, digitalized key pickup, legal records of guests, upsell of hotel services, online payment and local travel information. As with Previo Klíč, MyStay has its application. The following picture 3.2 shows the application.

If a hotel is not using one of the supported hotel management systems, it is not possible to use just the onboarding or the information part of the system.

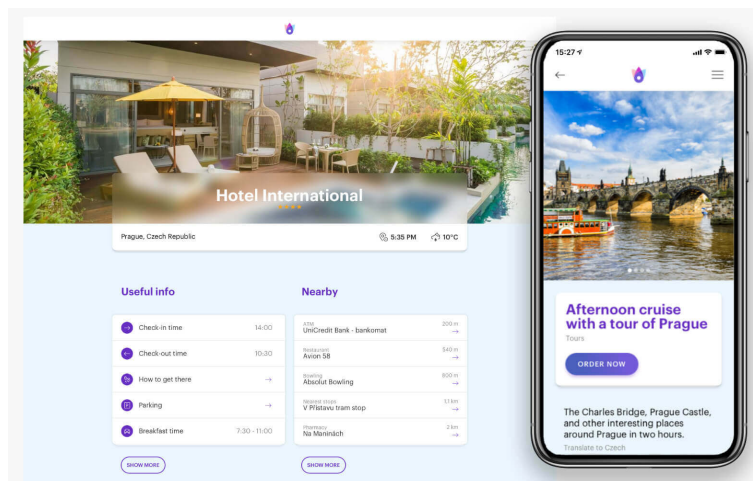


Figure 3.2. MyStay application interface.

3.8 Related Systems

The main one is the hotel website reservation and guest database POS system.

In Penzion U Veselých, we do not have a booking management system. Every order is managed manually. There are two main reasons: the first is a transfer cost, and the other is an operating cost. Every order has a record in one big MS Word table. It may seem ineffective, but it is beneficial for a small hotel like Penzion U Veselých, where flexibility is needed over scalability. Unfortunately, there is no single source of valid orders to use in our onboarding system.

3.8.1 Hotel Website

In 2021 Penzion u Veselých got new hotel website. New Order Process was introduced. Now guests can specify what rooms, how many people and what services they want.

This system is entirely under our control. We can use it to send more travel information through our onboarding process since we have contact information and the type of service the guests are requesting.

3.8.2 Booking.com

In 2022 around 45% of all orders were received through Booking.com in Penzion U Veselých. Booking has its ecosystem and APIs to help integrate hotels' custom needs.

¹ <https://www.gomystay.com/cs/discover/>

However, one must Booking.com partner. The partnership is meant for hotel management system providers and the overhead to integrate mandatory services and comply with all terms in the license is unmanageable and not practical on a small scale[8].

- PCI & PII compliance
- Cloud-based or Central Server based software
- Supports price and availability management
- Supports reservation management
- Supports real-time updates of rates and availability
- Supports instant confirmation of incoming reservations
- Supports managing property content

The idea was to hook into the booking.com events stream and dispatch information to the new guests. However, this is not possible.

■ 3.9 Check-in Process

Since we are a small business, we do not have a specialized booking system. Most of our orders go through online travel agencies, mainly booking.com and hotel.cz. We manually transfer these orders to one big DOCX table where all information is nicely grouped. This system has proven to be a nice balance between cost and efficiency. Booking systems are relatively expensive[9].

■ 3.9.1 Before Arrival

Booking.com guests receive initial information via the booking messaging system via an invitation letter.

After all order details (date and number of rooms) are worked out, direct order guests receive travel and accommodation information through an order summary email.

■ 3.9.2 After Arrival

Upon arrival, we usually spend 30 minutes explaining what guests can do in the surroundings depending on the time. During rush hours, all information is limited to just handing over keys to a room.

We usually explain what to do on the first day (short walks, restaurants) or how to go to main attractions. We show them not so well-known places.

Chapter 4

Formulation of Requirements

This chapter will formally summarize findings from the chapter Collection of Requirements. The following diagram 4.1] depicts how guests will be informed about the hotel and the activities they can do in the location.

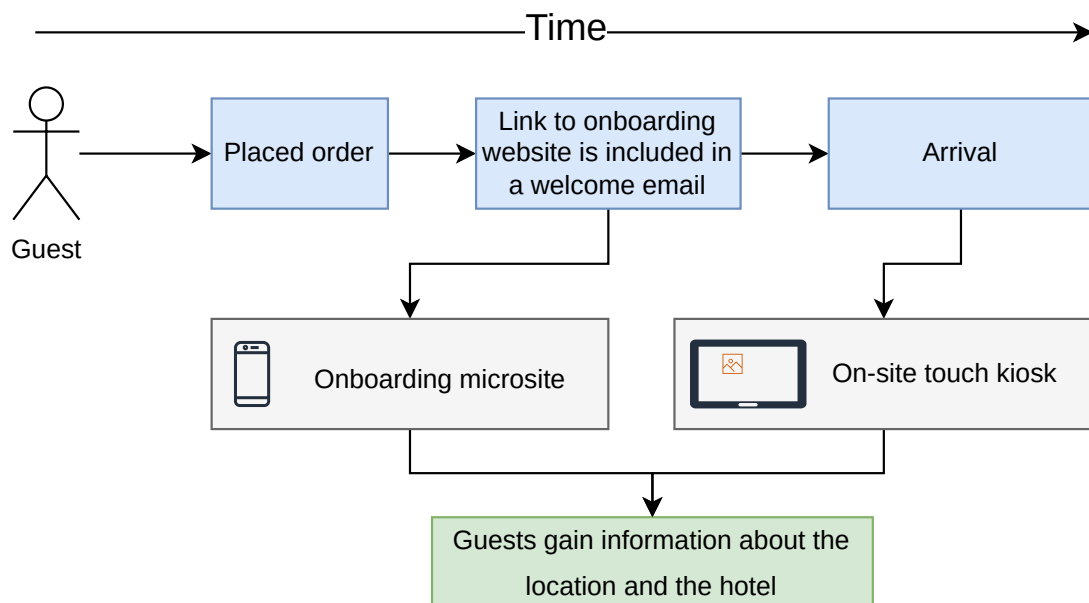


Figure 4.1. On boarding process diagram

4.1 Onboarding Microsite

This page will provide all non-order information that can be generalized and sent to all guests.

4.1.1 Functional Requirements

- **FR1 Microsite:** Guests will receive aggregate information about the location through a microsite accessible on the internet through a link sent in the order email.
- **FR2 Travel information:** Guests will be provided with activities and informative articles. Activity will contain a title text, navigation tags and images and be organized into categories. The article will contain text and a title.
- **FR3 Accommodation information:** Guests will receive the most essential hotel none order information, for example, what services are available, through a rich text editor.
- **FR4 Automated information delivery:** Upon a new order, the system automatically sends an email with a link to the information website with all travel and accommodation information.

4.1.2 Nonfunctional Requirements

- **NFR1 Easily accessible:** The microsite must be easily accessible for the user/guest — a link without registration.
- **NFR2 Single page:** All important information must be on a single page. The guests should not leave the page as they navigate through thorough details.
- **NFR3 Mobile support:** The online microsite must be mobile friendly for guests viewing content on site through portable devices.

4.2 On-site Kiosk

An interactable electronic kiosk with a touch screen will be placed next to the hotel reception.

4.2.1 Functional Requirements

- **FR5 Same content:** The same travel and hotel information as on the microsite will be available in the kiosk.
- **FR6 Hotel website:** Guests will be able to display the hotel website on the kiosk.

4.2.2 Nonfunctional Requirements

- **NFR4 Simple installation:** The electronic kiosk must be easy to install and manage for hotel staff.
- **NFR5 Software updateability:** The kiosk must be remotely updatable without any interaction with the hotel.

4.3 User Types

- **Admin:** One admin user per hotel. The user will log in through email and password. Will manage all data on the microsite.
- **Guest:** Will have read-only access to the microsite. No registration is required.

4.4 ER Model

The following diagram 4.2 summarizes which data is persisted. All fields named text contain rich text.

- **Hotel:** The core entity that holds all information about the hotel that is displayed on the onboarding microsite.
- **Travel info:** This entity represents helpful information that might be handy to guests but is not a single activity—usually information about local travel packages and transportation.
- **Trip category:** Groups trips for better navigation and search.
- **Trip:** This entity holds all information about one trip the guest can view.

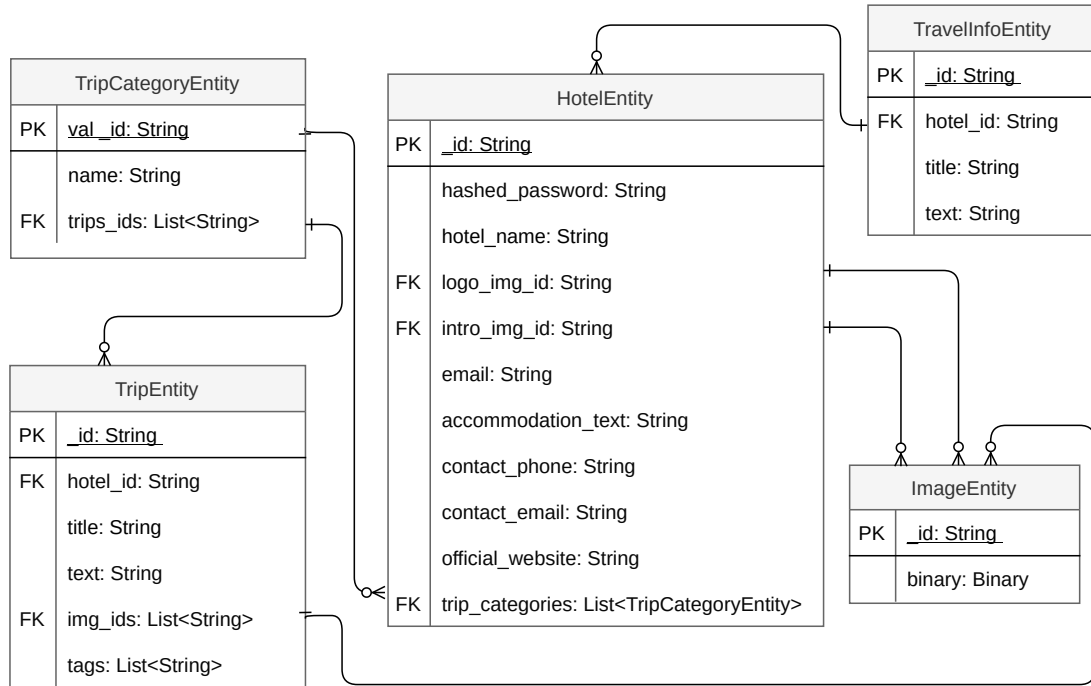


Figure 4.2. ER model.

Chapter 5

Product Idea Validation

An essential part of the product idea validation process is prototyping. Through early mockups of the application, often just visuals, we can validate the usefulness of the application concept and see its UI/UX pain points. We observe users' first impressions under predefined tasks or how they freely discover application features.

We define prototype tests based on their fidelity level. Fidelity can be defined as how well the prototype resembles the final application in terms of visuals and functionality [10]. We should choose an appropriate prototype for different types of testing.

Low fidelity prototype serves to validate the product idea and form the fundamentals of the application's design, such as structure and layout. Details such as colours, transitions, visual effects, and excessively detailed functionality should be avoided since they may focus the user on validating the UI/UX, not the application's concept itself [11].

- Prototyping does not require many resources.
- Allows iterating ideas quickly.
- Less pressure on the users. Focus on the concept, not the design.

High fidelity prototype comes to play once we have a solid grasp of what features the application includes. We can model the user interface more closely. High fidelity prototype usually serves as source material by which frontend programmers develop the application. Tools like Figma have greatly simplified the whole process of making a high fidelity prototype look real.

- Tools can produce almost accurate application mockups ready for presentation to stakeholders and to perform user testing.
- Allows testing design details menus and transitions, which often adds to user experience the most.
- Designing clickable prototypes.

5.1 Designing User Experiments

Two tests were designed for product idea validation—the first one focused on the onboarding website and the second one on the on-site kiosk.

For both tests, a high fidelity Figma clickable prototype was introduced to the user through an appropriate medium: web browser and on-site touch screen panel. The prototype can be seen on picture 5.1.

However, the Figma prototype cannot simply be connected to tools like google analytics. Especially for the kiosk where there is a static single page application where there are no routes to track. Measuring the number of clicks with tools like Hotjar and generating Heatmaps [12] can measure user satisfaction. For more complicated user testing, there is a tool Maze.co.

The goal was to observe if the guests were likely to use the kiosk or the onboarding website. Applying a more robust approach with users following scenarios would break the typical environment. The test goal was not to improve the product but to decide if the idea was interesting enough to the guests.

5.2 Kiosk User Testing

- Discover if guests will interact with a touch screen panel where they might find some helpful information about the local place and the hotel.
- Evaluate the usefulness of the kiosk for hotel guests.

Setup

A high fidelity prototype was designed with a focus on travel information and hotel information. It included four trips and two articles about Pec pod Sněžkou town and its surrounding.

A 22 ViewSonic TD2223 touch display was placed on a table in the dining room powered by Raspberry Pi 4. The Google Chrome browser was set to kiosk mode with the online Figma prototype.

On arrival, when introducing the hotel to the guests, my parents informed them that they could find more information in the kiosk and their opinion on it and observed their reaction.

The guests were divided into two subgroups. The first one was introduced to the kiosk at the introduction to the hotel. The second one was not and was used as a control group. At the end of their stay, we tried to ask if they interacted with the kiosk or if they liked the idea.

```
chromium -kiosk https://figma.com
```

Potential biases

- The usability testing was conducted in the winter when guests' primary focus was not travelling but skiing. Unfortunately, the schedule did not allow us to push the date forward.
- The screen did not resemble a standard kiosk panel

Results

Throughout testing, there were 22 guests in the hotel over two weekends. There were 12 guests in the group introduced to the kiosk, and 5 answered that they liked the idea. 7 guests did not directly answer and changed the topic of discussion, usually asking for local information.

In the control group, which was not introduced to the kiosk, there were 10 guests. When they returned the room key to the reception, we briefly asked if they had noticed the kiosk. One family of 4 tried to use it but found it incomplete with a small amount of information. That was expected since it was a prototype.

There is a potential bias that guests are usually grouped by 2, 3, or 4 people, and their opinions are not independent since they are a couple or family. The individual answers are hard to count. If one group answered yes, the whole group was counted as yes since it takes only one person to spread all the travel information between his companions.

Highlights from interviews: The guests seem to like the kiosk, especially the younger generation. The interaction rate was not as high as expected but sufficient to try the idea. The structure of displayed information (general travel information and trips) seemed to fit the guest's needs

5.3 Info Website User testing

- Measure guest satisfaction with information received from the hospitality provider.
- Decide if an information website should be developed.

Setup

Like the first experiment, a high fidelity prototype was designed with similar components, but further information about the accommodation was added.

A link to the prototype was added to the invitation letter sent to the new orders. Unfortunately, Figma does not have simple built-in tools for measuring prototype interaction. Option two was to wrap up the prototype in HTML iframe and hook up google analytics. However, I defaulted to asking guests on arrival.

Potential biases

- The website was very simple and did not contain all information that could be provided.
- The prototype was not that interactive for the guest to draw their attention

Results

The results are hard to estimate since they are measured in an online environment, and there is a time interval between the view of the online link and interviewing the guests. From our previous experience, we know that guests do not read much information from emails we send them.

We sent 10 links to the online kiosk to the weekend guests. On arrival, two couples found said they displayed the link; however, there was not much information. The rest did not notice.

Penzion U Veselých
Informace pro hosty

Karta pobytu

Webové stránky ubytovatele

▲ Navigace

✉ veselypenzion@email.cz

☎ +420 774 406 784

Všechny informace o Vašem pobytu na jednom místě.

Vám nabízí příjemný odpočinek a romantické ubytování ve známém lyžařském i turistickém centru v Krkonoších. Nádherná příroda, aktivní dovolená v zimě i v létě a wellness služby jsou základem nezapomenutelného pobytu.

Snídaně
Snídaně 8:00 - 9:30 formou bufetu. Prosíme hosty, aby neodnášeli jídlo ze snídaně na pokoje
Obědy, večeře
V penzionu nepodáváme. Doporučujeme navštívit hotel Atlas nebo penzion Relax v centru Velké Úpy
Bar
Veškerá nabídka baru je na objednávku u domácích. Nápoje Vám přineseme do kulečníku i na terasu
Wellness
Wellness je na objednávku, začínáme v celou hodinu v rozmezí 16-21:00. Minimální počet osob 2, maximální počet osob 5. Cena za hodinu pro 2 osoby 400Kč, pro 3-5 osob 600Kč. Děti mladší 10 let mohou být ve vířivce jen pod dohledem rodičů.
Fitness
Cvičit můžete zdarma do doby, než bude v provozu sauna nebo vířivka. Posilovací stroje mohou používat pouze osoby od 15 let. Je zakázáno používat stroje v průběhu provozu wellness.
Kulečník a stolní tenis
Otevřeno po celý den. Neodnášejte prosím páčky a míčky. Nechávejte je místě, stejně i tágo a koule. Kulečník smí používat jen osoby od 10 let

Nápady pro vaši dovolenou

Penzion U Veselých Jun 20,2020

TourPAS – využijte lanovky

V Pecí pod Sněžkou cestou na lanovku na Sněžku je umístěna bobová dráha o délce 900m a areál Lemurie pro malé i velké děti plný proltaček, dlouhých adrenalinových skluzavek, velkých

Penzion U Veselých Jun 20,2020

Návštěva Sněžky lanovkou

Návštěva vrcholu Sněžky patří k základním bodům každého pobytu. Na Sněžku vede lanová dráha složená ze dvou úseků. Horní úsek je často mimo provoz z důvodu větru. Od penzionu jedete

Penzion U Veselých Jun 20,2020

Relax Park v Pecí pod Sněžkou

V Pecí pod Sněžkou cestou na lanovku na Sněžku je umístěna bobová dráha o délce 900m a areál Lemurie pro malé i velké děti plný proltaček, dlouhých adrenalinových skluzavek, velkých

Penzion U Veselých Jun 20,2020

Jízda na koloběžkách

Můžete si vypůjčit koloběžku a jet z Portálek naproti penzionu 5 km nebo z Černé hory 10,5 km dolů. Půjčtiny koloběžek jsou na dohledních i horních stanicích lanovek. Pokud máte TourPASS, půjíte si

Vyberte si výlet

Nehledanější

Pro děti

Atrakce v okolí

Výlety autem

Turistika

V okolí

Penzion U veselých Jun 20,2020

Pevnost Stachelberg 20km

Pojedete asi 20 km autem, nejprve směrem dolů, před Trutnovem uhnete doleva na Žacléř, až se po stoupání dostanete nad obec Babí na parkoviště. Odtud je to asi 1 km pěšky k největší pevnosti z II sv.

Sport
Příroda
♥

Penzion U veselých Jun 20,2020

Zámek Kuks 50km

Pojedete autem do Trutnova, zde na 3 kr. objezdu doleva a dál směrem na Adršpach. V obci Chvaleč se dáte doleva. Aby jste zaparkovali u skal, je nutné si koupit online vstupenky s rezervací parkování, o

Sport
Příroda
♥

Penzion U veselých Jun 20,2020

Safari park Dvůr Králové 41 km

Pojedete směrem na Hradec Králové a v obci Kocbeňe uhněte doprava na Dvůr. Pojedete podle šipek až k ZOO a za vstupní bránou je největší parkoviště. Jinak půjíte pěšky ke vstupu dost

Sport
Příroda
♥

Penzion U veselých Jun 20,2020

Na kole v Pec pod Sněžkou

Pojedete směrem na Hradec Králové a v obci Kuks uhněte doprava na parkoviště. Zámek Kuks je proslulý zejména svým exteriérem s velkolepými sochami Matyáše Brauna a nádhernou

Sport
Příroda
♥

Travel Info
2022

Figure 5.1. High fidelity prototype for website and kiosk user testing

Chapter 6

Available Technology

6.1 Web Frameworks

6.1.1 Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can run without much configuration. Spring Boot extends the Spring framework (Build system and dependencies), eliminating the boilerplate configurations required to set up a Spring application. Spring or Spring Boot has been an industry level standard for more than a decade, initially released in 2004. It is a very mature framework. There are lots of examples and excellent documentation.

Spring Boot claims to be a very productive framework¹. The configuration, setup, and maintenance cost is relatively low due to the tremendous support from other vendors. Spring has many tools and extensions to cover all needs for a standard application. It is a Java Virtual Machine framework and even has support for Kotlin.

Compared to Ktor where all package management is done in Kotlin, the package and dependency management in spring boot could be troublesome². Sometimes requiring XML configuration files which can be hard to manage.

The biggest challenge many developers face when using Spring Boot is the lack of control. Everything is preconfigured; there is a much higher cost in implementing a feature if the programmer needs something extra.

6.1.2 Django

According to their website, „Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.“[13] The framework relies on the traditional model-view-controller (MVC) architecture.

The main advantage is its maturity; Django has been in development since 2005. It is effortless to use, and most general problems are solved and integrated into its standards. Python programming offers many libraries, from data science to server maintenance.

Python is not JVM compatible; hence, it is unsuitable for projects where a substantial part of the shared codebase can be reused in developing an Android or Kotlin Multiplatform Mobile application. Nevertheless, it is still a valid option despite its possibly higher maintenance cost. It is much easier to manage a less diverse codebase from a project cost perspective.

¹ <https://spring.io/why-spring> Why Spring?

² <https://docs.spring.io/spring-boot/docs/current/reference/html/howto.html#howto.application.troubleshoot-auto-configuration> Troubleshoot Auto-configuration

6.1.3 Ktor

Ktor is a lightweight yet extendable framework developed by JetBrains for its products made public in late 2018. The framework is built in Kotlin programming language and harnesses its newest features, such as kotlinox serialization and coroutines. Paralyzation and asynchronous computing is built into all of its components.

Despite its nice appearance and corporate backing, there are many drawbacks. The framework is not yet very mature and lacks any standards compared to Spring. The current version (2.0.0) has no official support for dependency injection and database engines. Although with some additional work and third-party plugins and libraries, it is possible to provide those features. These features are expected to be integrated into the next major release. The documentation is well written, though it lacks complex examples.

The framework excels at the simplicity of defining routes and extensibility. The flow of network requests is designed in a pipeline design pattern. It is easy to hook any middleware, validation, or custom logic into the flow.

Many of the most used plugins are provided out of the box by JetBrains, such as Authentication (JWT, OAuth, Basic). The relative abundance of plugins makes the development of a product quite fast.

Ktor is not just a server-side framework. Standard features such as handling and parsing requests and API calls are extracted to the separated module and can also be used on the client. It is extremely useful in mobile development and KMM (Kotlin Multiplatform mobile). If there is a plan to develop a native mobile app (Android, iOS), Ktor is a valid option.

6.2 Database Storage Engine

Database model and underlying database engine heavily influence the type of architecture in the application and vice versa. We need to consider the kind of data, how the data is used and queried, and what kind of application we are developing (real-time, long term storage, information critical).

The domain model conceptualizes the internal connection between different business entities in a given domain. The critical role in application development is choosing or developing a database compatibility and abstraction layer. The easier the abstraction layer is to use and the more flexible it is, the faster the development of the application will be. These are often contradictory properties.

In computer science, ACID is a set (atomicity, consistency, isolation, durability) of properties that guarantee data validity at any given time. The ACID property is mainly tied to database transactions. These are robust guarantees, which may slow our application. There are valid cases where we can omit them. Some NoSQL databases trade the performance gain over consistency[14].

6.2.1 Scalability

The fundamental question we need to ask when building the application is the expected database load and the company's growth rate. The bottleneck will most likely be the storage engine and data accessibility considering the typical commercial applications focused on users and data management.

Most database engines are well tuned in their default settings - single node, no partitioning. Scaling too early can lead to a much more significant development cost without

focusing on core user features that bring the product the most value. Nevertheless, the model should be prepared for future optimization and scaling.

- **Multiple nodes:** Most of the state of the art databases offer multi node database replication. The data is replicated in multiple nodes and synchronized depending on the configured policy. This gives us greater fault tolerance. Applications emphasizing reading over data writing can boost performance since the read queries are distributed to multiple nodes.
- **Horizontal partitioning:** It is beneficial when there are too many data records in a table. The table is split horizontally by rows in the SQL world, usually by some aggregate key, a region.
- **Vertical partitioning:** Very similar to the Horizontal Partitioning but the data is split vertically. One data record is divided into two records. For example, we strip low accessed big data like bio from the username and password, making the first table smaller, thus allowing the query to perform faster logins.

■ 6.2.2 Document Databases

A document-oriented database is a storage system designed to store, retrieve, and manage semi-structured data. The semi-structured data model is a model with no clear separation between schema and data.

Advantages:

- Related data but with different schema can be stored together.
- Querying relationships in relational databases is a costly operation. Often, a join table is needed for more complex relationships such as n:m. The Query document can internally store this relation with its data, thus allowing it to query faster join data.
- The schema can easily change - beneficial for fast development iteration where the product specification is not precise initially.
- The storage format often resembles the communication interface output format. The same parser can serialize database entities in different communication formats.

Disadvantages:

- Since the schema is volatile, consistency might suffer.
- Compared to relation databases, fewer constraints on data attributes. The application must validate the data.

■ 6.2.3 Relational Databases

Relational databases are based on a relational model. The model consists of tuples (data) and relations (connections) between the tuples. The definition of a tuple is fixed. In an implementation, we use tables that hold identical data records and are connected via primary keys. Since the schema is fixed and the types of individual fields in the table are known ahead, the database engine performs validation upon any modification, thus increasing the data consistency.

The relational databases are usually tuned to ACID properties compared to NoSQL databases where the BASE is preferred. Transaction handling is well implemented and offers strong guarantees.

Advantages:

- In applications where strong consistency is needed, relational databases are a great choice due to their focus on schema validation.
- There are many industry standards and libraries written for RDBMS which can speed up the development.
- Almost all frameworks have integrations to the most used relational databases: PostgreSQL, Mysql, MS SQL.

Disadvantages:

- Any changes to the scheme are a significant disruption to the application operation. Usually, migrations must be performed.
- If the nature of data is heterogeneous Relational databases are not appropriate.

6.3 Web Frontend Architecture

There are many ways to implement the frontend of an application. This field is rapidly developing. Moreover, we need to pay close attention to which technologies will allow us to be the most efficient and productive.

Applications have become more complex with an accent on user experience. Applications are more interactive than ten years ago, and users expect easy-to-use applications that are anything but simple to develop. Solutions to those problems might bring caveats in another field, such as SEO support in single-page applications, which might be interactive but not well optimized for search engines. Implementing an application without any framework is laborious and error-prone. There is a lot of duplicate code, and it is hard to manage the project.

When choosing a frontend web technology, we need to ask about the purpose of the application.

- For an internal application, we do not need a framework optimized for SEO.
- Is there a company preference for a framework.
- How well is the framework supported, and how big a community does it have.
- Are there any critical technology caveats, such as not supporting certain features?

6.3.1 Server Rendered

Server rendering means that the client will receive a complete minified version of a given website upon each web request. The server has a much higher workload than client rendered web applications since it must prepare the data for the website and then render it. It also requires higher internet traffic because the whole page is transferred every time.

6.3.2 Client Rendered

Improvements in JavaScript allowed us not just to make applications interactive and render them entirely on the client-side. The frontend is split into components which drastically simplifies the development. These frameworks offer many community plugins, and the closer to the web they are, the better they can adjust to the browser.

■ 6.3.3 Hybrid Approach

As discussed before, SEO optimization is a dealbreaker for some applications. Luckily, some meta frameworks, such as Nuxt for Vue and Next for React, offer the best of both worlds. They render the first page on the server or even prerender it and cache it. The page is served to the client and hydrated. Hydration means binding the component build system and event logic to the already rendered HTML tree.

■ 6.3.4 React

React is a library developed by Facebook for creating reactive web applications. Having a company of this size back the project is a tremendous benefit since the open-source project has ground in commercial business and thus has the resources to maintain and extend the features. Along with Vue, they create an industry standard. According to the state of the JavaScript survey 2021[15], React is the most used framework. React is used by Airbnb, Slack, and Instagram.

The most significant benefit of React compare to other frameworks such as Vue or Angular is its capability to develop multiplatform mobile apps. However, this is beneficial only for projects planning on developing mobile apps.

Highlights:

- **More packages:** React has more packages on NPM than Vue 190k to 60k.
- **Backed by Facebook:** Corporate backing brings trust into the project's stability and scalability.
- **Crossplatform:** React can be used to develop mobile applications.

■ 6.3.5 Vue

Vue is solely an open-source project backed by its community. It was created as a reaction to React disadvantages. According to the state of the JavaScript survey 2021[15], Vue is the third most used framework behind React and Angular. Companies that use Vue.js are Grammarly, Upwork and Nintendo.

Highlights:

- **Easy to learn:** Development with Vue.js is slightly faster than with React. Vue feels simple a has a narrower learning curve³.
- **Concise documentation:** Vue.j has Great documentation with extended comunity.
- **Prototyping:** Vue defaults to two way binding. It can easily manage many inter-actions. It is easy to set up and deploy and aims to be lightweight, thus allowing a developer to make prototypes quickly.

■ 6.4 API Design

Applications consist of multiple different technologies and components to compose the final product, and they all have to communicate with each other. Most modern web applications expose APIs that clients can use to interact with the application since the backend and frontend are entirely separate. According to MDN, there are two key concerns when designing the APIs.

³ <https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/>

- **Platform independence:** „Any client should be able to call the API, regardless of how the API is implemented internally. This requires using standard protocols, and having a mechanism whereby the client and the web service can agree on the format of the data to exchange.“[16]
- **Service evolution:** „The web API should be able to evolve and add functionality independently from client applications. As the API evolves, existing client applications should continue to function without modification. All functionality should be discoverable so that client applications can fully use it.“[16]

■ 6.4.1 REST

Representational State Transfer (REST) is an architectural technique for designing web APIs. REST APIs are designed for accessing resources, any object, data, or service on the web through a URI via standard protocols, most likely over HTTP methods.

- **GET:** means to retrieve some resources for a given URI. URI is usually structured in parts or has optional parameters to identify more specific resources.
- **POST:** creates or updates data on a given URI. POST has an additional input body that can carry the new or newly updated resources.
- **PUT:** is meant to create or replace a given resource by the PUT input body.
- **PATCH:** is for performing a partial update.
- **DELETE:** removes a resource on a specific URI.

OpenAPI / Swagger has become standard for designing REST APIs. OpenAPI is a resource description format that states the resource schema and URI parameters for resource manipulation.

Disadvantages:

- **No clear standard:** The lack of any clear standard allows API authors to bend or adjust their REST APIs to their visions. Therefore APIs differ, especially in using POST, PUT, and PATCH.
- **Data fragmentation:** REST by design separates resources into different URIs. However, this is not how applications usually display data. Applying filters or querying nested data can only be performed by making multiple calls and combining the data, which is ineffective.

Advantages:

- **Simplicity:** Setting up rest API is a straightforward thing since almost all technologies support this type of communication.
- **Tooling:** There are numerous tools to standardize API development, such as Swagger.

■ 6.4.2 GraphQL

„GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.“[17]. According to the state of JavaScript, the popularity of GraphQL is rising[18].

GraphQL was released in 2012 to solve the problem of querying nested data and its manipulation. Instead of maintaining multiple endpoints as in standard REST API, GraphQL exposes only one entry point for GraphQL queries and another for GraphQL Schema definition. GraphQL allows the programmer to query any data from the static schema in one request - applying parameters and filters. Managing one endpoint for data access is much more scaleable.

Advantages:

- **Fetching data with a single API call:** In GraphQL, data is linked together; a client can query those links by a GraphQL Query language.
- **Smaller message size:** GraphQL allows you to specify only the object fields that you need, thus making the response smaller.
- **Document Databases:** GraphQL blends nicely with document databases. Usually, the scheme is autogenerated from Backend entities, so adding one field to the document database will propagate to the GraphQL schema definition since the query language, and the document database is based on semi-structured schema.

Disadvantages:

- **Granularity of authentication:** In regular rest API, multiple endpoints could support different types of authentication. In some implementations, exposing parts of the schema to the public and keeping others under authentication is hard to implement.

Chapter 7

Backend Architecture

This chapter describes the decisions behind choosing the architecture for the travel-info.cz project. We will summarize critical points from the collection of requirements and combine them with future plans for this project.

7.1 Requirements

The minimum viable product described in the thesis is straightforward and does not require advanced approaches to architecture. In the assignment of this thesis is recommended to use event-driven architecture. The reasoning behind this was that the application was supposed to hook on Booking.com or the new hotel website order system. Unfortunately, the requirements to use Booking.com's API are not in Penzion U Veselých capability to comply. The main architecture requirements from the point of MVP are:

- Support CRUD operation on multiple entities
- Support image database
- API for remote electronic kiosk
- API for frontend
- Up to 60 daily active users

If proven sustainable, the project aims to collect travel advice from hotels and publicly offer travel tips with advertising:

- Exponentially higher load
- Multiple domains — travel tips, orders, advertising
- Emphasis on data consistency
- Hundreds of users

7.2 Possible Approaches

Likely, the focus of MVP will change in the future, and its architecture will have to adjust. The higher the granularity of low coupled modules, the easier the application development will be. From the point of MPV iterative improvement, the application must support continuous integration — introducing changes quickly and automatically.

„Every architectural decision should be testable and should have a test written to accompany it. If a decision is not testable, then it is merely an opinion or a suggestion and not a decision.“ [19] Iterative software development brings the advantage of reactivity to project challenges. However, it puts more strain on ensuring already developed software quality since multiple refactoring is involved.

■ 7.2.1 Event-Driven

„Event-driven Architecture (EDA) is a software architecture paradigm promoting the production and consumption of events.“[20]. The main purpose of this architecture is to usually solve the real-time collection of data through event processors such as Kafka. From Kafka’s documentation „Technically speaking, event streaming is the practice of capturing data in real-time from event sources like databases, sensors, mobile devices, cloud services, and software applications in the form of streams of events; storing these event streams durably for later retrieval. “ [21]

In our example, it would be hooking into Booking.com’s new order event stream and issuing two more events:

- Welcome email: A general welcome email would be sent to the customers with an onboarding site upon new orders.
- Before arrival email: Guests would receive an email with the essential information two days before arrival. It would require an email job scheduler such as JobRunr [22].

However, this will not be implemented due to legal challenges in getting access to the booking.com API.

■ 7.2.2 Microservices

„Microservices are independently releasable services that are modelled around a business domain. A service encapsulates functionality and makes it accessible to other services via networks—you construct a more complex system from these building blocks. “ [23]

Each service encapsulates its small domain. The main goal of splitting the monolithic application is not the performance gain. The monolith can be distributed to multiple nodes, and if properly designed, committing database transactions should block only a portion of the whole database. The goal is to bring maintainability and higher separation of concerns to each service, leading to faster adoption of more modern technologies and more rapid improvement to the service itself. However, the price is the higher cost of operation. If we look at some advantages taken from [24]:

- It tackles the complexity problem by decomposing the application into a set of manageable services that are much faster to develop and much easier to understand and maintain.
- It enables each service to be developed independently by a team that is focused on that service.
- The microservices architecture enables each service to be scaled independently.

■ 7.2.3 Monolith

The monolith approach is the most common approach despite its negative connotation. Most applications and libraries are designed to be used in a monolithic environment. The application is designed to be one big unit. The unit can be modularised, but there are no network barriers between modules internally. It significantly simplifies development for a small project where the complexity of the whole ecosystem does not bring maintenance and refactoring issues. Advantages are:

- Simple to develop. The lack of network communication between internal modules significantly simplifies adding new features and refactoring.

- Simple to deploy. The deployment does not require any cooperation with other services. The application is usually packaged in a single container.
- Simple to scale horizontally by running multiple copies behind a load balancer.

Over time more domains are built into the monolith as the application and business grow. Introducing any internal changes that do not change the application's behaviour is challenging since the modules are usually highly coupled. Internal changes include changing the underlying database engine or event processor. Disadvantages are:

- Any minor change requires redeploying of the whole application.
- As the size, complexity and number of internal modules increase, it puts more strains on the individuals to correctly add or refactor code.
- Possible increase in technical debt. Monolith is tight down to one technology. If the technology becomes unmaintained or there are better alternatives for solving the same challenges, it is tough to introduce them to the monolith.

7.3 Monolithic Approach

As said in 7.2, this project has two stages. The goal is to propose an architecture that will cover both stages or will be able to transform from one stage to the other.

As for the first stage, a monolithic approach would be ideal since the project is small and very few requirements would need a special kind of architecture. The amount of users is very low. If authentication and user management are excluded, the data model bounds are within one domain (Travel information).

The second stage of this project will likely require more domains (Hotel management, Travel information, Advertisement,...) to be integrated. Since the first stage covers only one domain with a few exceptions in the future, it can become a single service for managing travel information, and the user and hotel management can be extracted into other services.

However, we need to pay attention to „ A monolith haphazardly decomposed into a handful of microservices could actually leave you in a worse state compared to where you started. We even have a term for that: a distributed monolith. “[20]. This problem is solvable since the monolith at its core is a single service.

Ktor web framework was chosen as the backbone for the application's monolith backend. There are two key advantages when using Ktor. First, Ktor is built using Kotlin, and with Kotlin, we can create multiplatform projects meaning we can reuse the same codebase for both frontend and backend. The second reason is that a Kotlin-GraphQL library from ExpediaGroup immensely simplifies the definition of GraphQL schema using Kotlin classes.

GraphQL, in our situation, is preferable over standard REST endpoints due to the simple integrability with MongoDB. Document databases are composed of nested objects. GraphQL allows us to query those nested relations defined on the data layer without creating many different endpoints for different needs.

7.4 Layered Architecture

There is no predefined structure for Ktor Projects. Ktor is a very simple yet extensible framework and gives the author the advantage and the responsibility to define the structure of his application. Our project follows standard layered architecture.

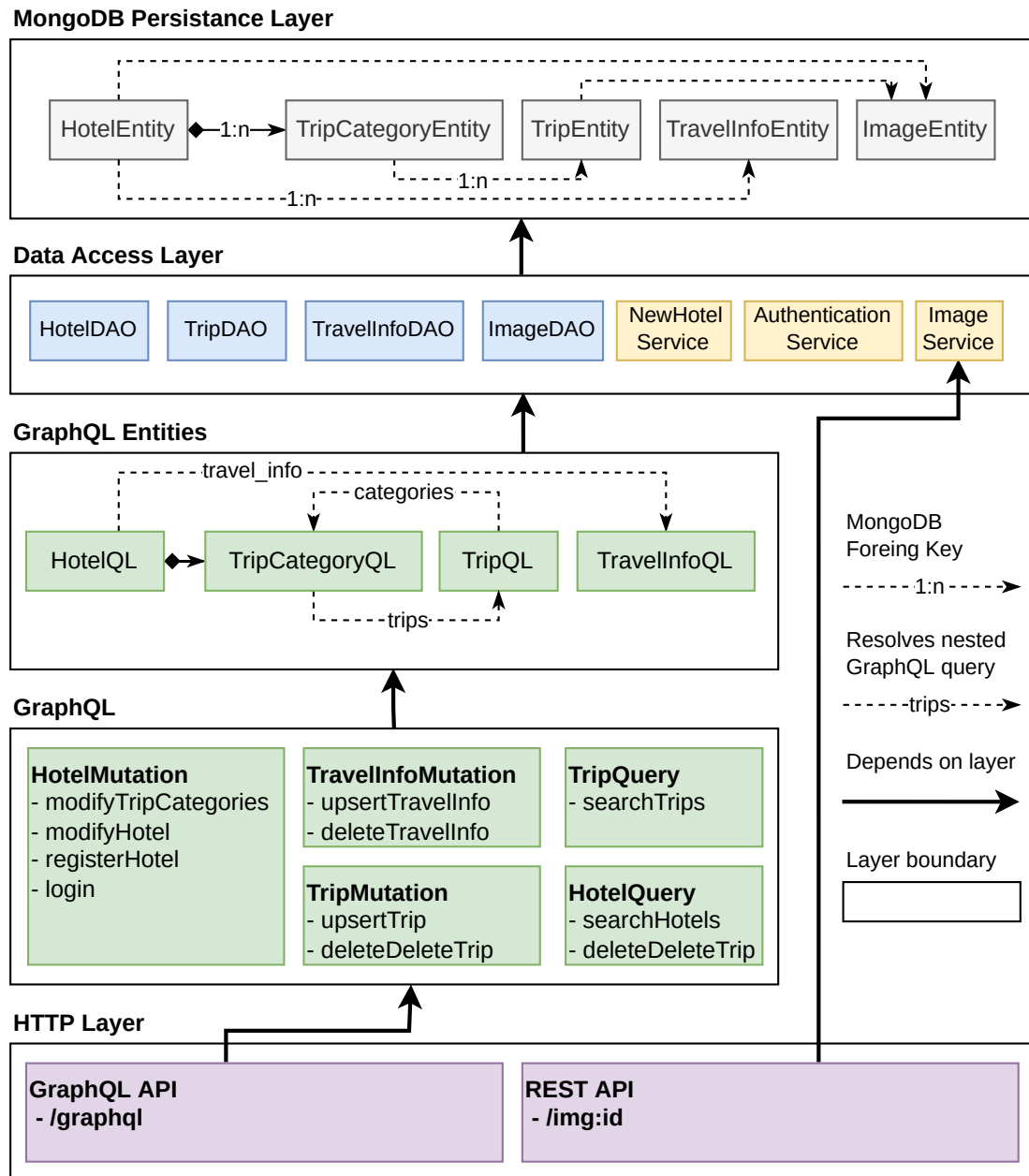


Figure 7.1. Layered architecture of the application.

7.4.1 Data Layer

The data layer was implemented using `KMongo`¹. It is an easy-to-use wrapper around Mongo DB Java driver, supporting Kotlin asynchronous programming using suspend functions. It is convenient since the underlying framework is built on Kotlin Coroutines (suspending functions). It also features `kotlinx.serialization`², which provides reflectionless serialization and deserialization of Kotlin's data classes from and to BSON.

The `data` Layer provides most atomic MongoDB operations such as `find`, `findMany` as well as helper functions for working with entities, for example, `save(e: Entity)` `save(e: Entity)`.

¹ <https://litote.org/kmongo/> KMongo - a Kotlin toolkit for Mongo

² <https://kotlinlang.org/docs/serialization.html> Introduction to `kotlinx.serialization`

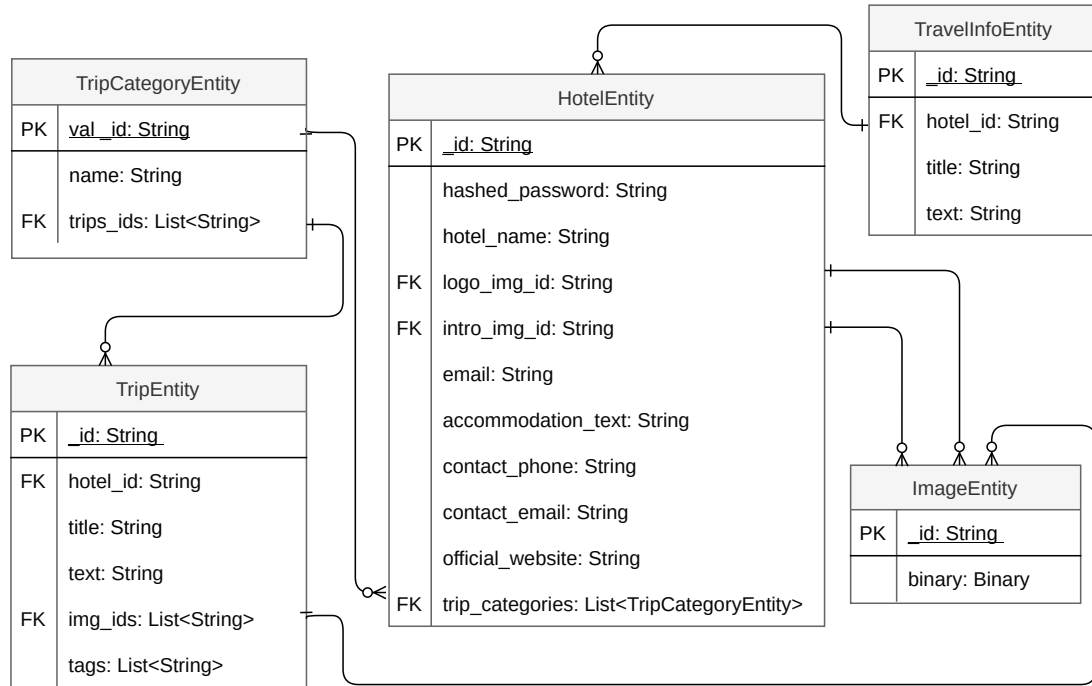


Figure 7.2. ER Model diagram.

7.4.2 Data Access Layer

The data access layer builds on the data layer by providing simple business queries. This layer validates the most significant application constraints, such as data ownership and prevents cross update operations. Neither Ktor nor KMongo provides a standard way to create DAOs. The most efficient is to create a single DAO component for each entity. Through dependency injection, all DAOs are available anywhere in the application.

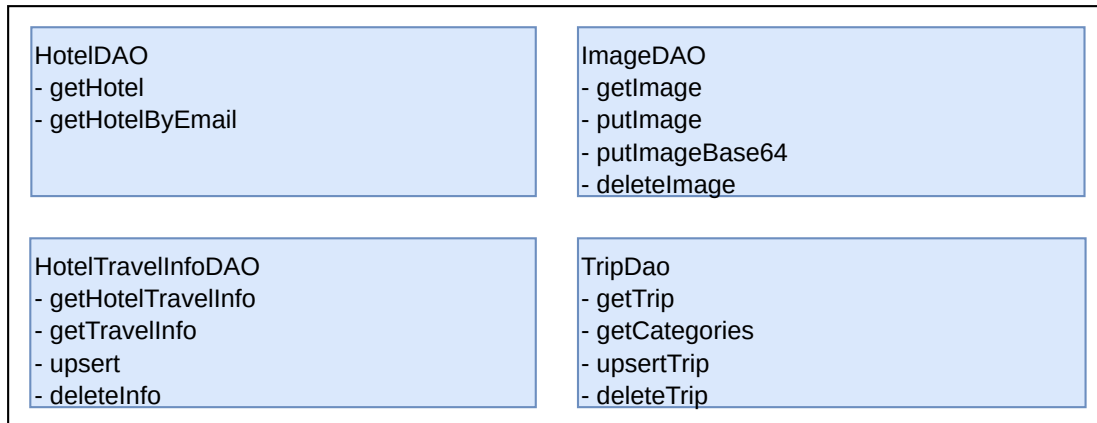


Figure 7.3. A sub list of operations on DAO objects.

7.4.3 GraphQL API

The GraphQL schema is depicted in the diagram 7.1. It was implemented using GraphQL Kotlin library from Expedia Group³. It is a standard implementation without any significant architectural challenges. The only problem was how to map GraphQL entities to database entities nicely, which is discussed in ??.

³ <https://opensource.expediagroup.com/graphql-kotlin/docs/>

■ 7.4.4 HTTP Layer

Takes GraphQL request, parses it, asks the GraphQL for the response, serialize it and sends it to the client. This layer also provides authentication for data mutations.

This layer is mostly about configuration. For additional details, visit provided links.

- **CORS**⁴: Allows HTTP post with `Authorization` header for CORS request from Vue frontend.
- **Authentication**⁵: Provides JWT authentication from email and password login.
- **GraphQL context**: Object with a map containing authorized user and other essential information for executing GraphQL query.

⁴ <https://ktor.io/docs/cors.html> How to set up CORS for Ktor.

⁵ <https://ktor.io/docs/jwt.html> How to set up JWT for Ktor.

Chapter 8

Implementation Details

The source code can be found here: <https://gitlab.fel.cvut.cz/veselj57/diploma-thesis>.

8.1 Backend

8.1.1 GraphQL Entity Mapping

One of the main challenges was developing an internally simple GraphQL and MongoDB connection, which would be easy to maintain. Using the exact entity definition for GraphQL and MongoDB is effortless, but that brings numerous minor problems of hiding unwanted fields in public API, such as password hash. Separating those entities brings more boilerplate code but better architecture and separation of concerns. Another problem is the need to map DB entities to API ones.

Entity mapping is done using the Kotlin extension functions ¹. It significantly simplifies the mapping code and allows us to put the mapping to the GraphQL entity file. With the GraphQL entity and the mapping in the same place, we do not pollute the DB entities with code from other layers. In the following example, we can see the definition of GraphQL entity with relation and mapping. The function `toGQL` in figure 8.1 shows the mapping between DB and QL entity.

```
@GraphQLName("TripInfo")
class TripInfoQL(
    var _id: String,
    var hotel_id: String,
    var title: String?,
    var text: String?
): KoinEntity {
    suspend fun hotel(dfe: DataFetchingEnvironment): HotelQL? {
        return get<HotelDAO>().getHotel(hotel_id)?.toGraphQL(dfe)
    }
}

fun TravelInfoEntity.toGQL(dfe: DataFetchingEnvironment): TripInfoQL {
    return TripInfoQL(_id, hotel_id, title, text)
}
```

Figure 8.1. Defining GraphQL entity from DB entity.

¹ <https://kotlinlang.org/docs/extensions.html>

8.1.2 GraphQL Authentication

JWT² is used to carry authentication between the frontend and backend. Because GraphQL uses one route for queries and mutation, authentication is optional to prevent redirects on the HTTP layer. Nevertheless, it is evaluated, and the result is passed to GraphQL.

Two factors must be checked to prevent cross update of data, for example, inserting a trip into another hotel. The first one is successful authentication, and the second one is verifying the ownership of the resource to the authenticated entity.

„GraphQL directives can be used to transform the schema types, fields and arguments as well as modify the runtime behavior of the query (e.g. implement access control, etc). Common use cases involve limiting functionality based on the user authentication and authorization.“ [25]

All queries are publicly available, but mutations are protected with a directive `@AuthHotelDirective` it checks whether the user was successfully authenticated when processing the field with this directive. Lastly, in the implementation of the mutation, its resource is checked against the authenticated user.

The following example 8.2 shows the injection of authentication into `DataFecher` of the protected resource. A custom GraphQL exception handler then catches the unauthorized exception. The injection works like a `proxy design pattern`, mediating the data retrieval.

```
val originalDataFetcher: DataFetcher<*> = environment.getDataFetcher()

environment.setDataFetcher { dfe ->
    val role = dfe.graphQlContext.get<GQLRole>(ROLE_KEY)

    if (role !is GQLRole.Hotel)
        throw UnauthorizedGraphQLRequest()

    originalDataFetcher.get(dfe)
}
```

Figure 8.2. Extending data fetcher with authentication.

8.1.3 Input Validation

There are multiple options for backend validation. The first is constraining the field by adding directives³ such as `@NotBlank`. It fits into the GraphQL ecosystem but cannot be reused in our multiplatform client.

The other option is to put the validation into the entity’s constructor and adjust the GraphQL error handling. GraphQL, when creating entities, throws a reflection error despite whatever the cause. If a `ReflectionExcecion` occurs, we can look for the cause in the stack trace and nicely print the error output and ignore the reflection error.

The following example 8.3 shows the constructor of `TripQL` entity which performs the validation.

² <https://ktor.io/docs/jwt.html#configure-jwt> Ktor JWT configuration.

³ <https://github.com/graphql-java/graphql-java-extended-validation> Extended Validation for graphql-java

```

init {
  require(title.length < TITLE_LENGTH)
  require(text.length < TEXT_LENGTH)
  imgs.forEach { require(it.length < IMAGE_LENGTH) }
  tags.forEach { require(it.length < TAG_LENGTH) }
}

```

Figure 8.3. Validating multiplatform entity.

8.1.4 Storing Images

In our application, we need to store images for trips and hotels. The standard choice is to save them on disk and serve them using a web server. However, introducing second storage would make harder the application deployment and backup. We can leverage Mongo DB's ability to store raw binary data. There are two options, store files using GridFS and create a document with binary data. „If your files are all smaller than the 16 MB BSON Document Size limit, consider storing each file in a single document instead of using GridFS. You may use the BinData data type to store the binary data.“ [26]

The second problem is transferring the files through GraphQL. GraphQL is purely text-based. The only option is to encode the binary image into a base64 string. However, it increases the file size by 33% [27]. The retrieval of the image is done using standard HTTP image transfer done by Ktor public route `/img/{image_id}` as depicted on the picture 8.4.

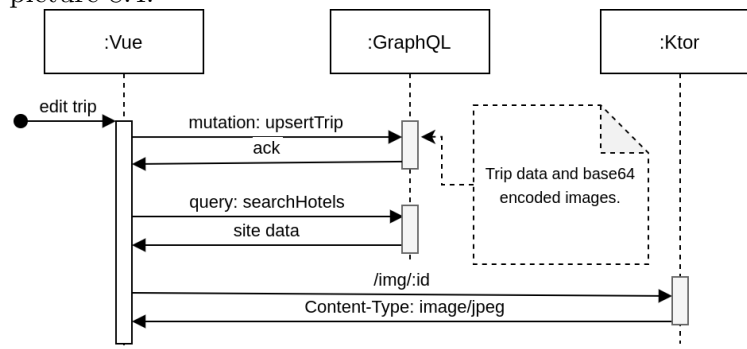


Figure 8.4. Image persistence diagram.

8.2 Multiplatform client

Programming multiple client libraries for different languages is expensive and error-prone, for example, if a field is present in the JS variant but not in the JVM one. Using loosely typed languages has some advantages, especially since there is no need to define DTOs (Data Transfer Objects). However, on the other hand, a programmer must always refer to the documentation instead of using IDE hints and error warnings about types.

This effort aims to generate a client library from the GraphQL definition, which is code-first. The opposite approach is to generate⁴ java DTOs from the schema first approach.

⁴ <https://github.com/graphql-java-generator/graphql-maven-plugin-project> GraphQL in a schema first approach.

The model is composed of all unit actions the GraphQL specifies. GraphQL mutations are easy to model; they have clearly defined input. GraphQL queries are hard to model and unify under OOP. We can either query more data to bind all subentities or increase the granularity of atomic top-level queries and make more calls. The client library chooses the first option.

The model contains these actions: `registerHotel`, `login`, `modifyCategories`, `upsertTrip`, `upsertTravelInfo`, `getHotelData`.

This library is not only for client implementation but can be used in integration testing; more on this in section 8.2.4.

■ 8.2.1 Kotlin Multiplatform

„Support for multiplatform programming is one of Kotlin’s key benefits. It reduces time spent writing and maintaining the same code for different platforms while retaining the flexibility and benefits of native programming.“ [28]

We can write standard Kotlin code, which is then compiled to other platforms, in our case, to JVM for GraphQL testing and Typescript for frontend development. Kotlin native is still very experimental, and more stable support for JavaScript / TypeScript is in active development.

■ 8.2.2 Kotlin Symbol Processing

„Kotlin Symbol Processing (KSP) is an API we can use to develop lightweight compiler plugins. KSP provides a simplified compiler plugin API that leverages the power of Kotlin while keeping the learning curve at a minimum.“ [29]

KSP is necessary because Kotlin compiles data classes to JavaScript classes, but we need TypeScript interfaces. Our model is defined in data classes; they are needed for a JVM client. KSP compiler plugin generates interfaces with the same fields from the data classes. Those interfaces are then correctly compiled into TypeScript interfaces for the JavaScript module. Example of compiled DTO object 8.5.

■ 8.2.3 Compiling DTOs

The initial intention was to generate the schema by analyzing the definition of GraphQL entities. Unfortunately, I could not configure the Gradle Kotlin Multiplatform project to both run the application and generate the code because, in some cases, it would bring cyclic dependency to the project.

- **Common DTO definition:** For each GraphQL type, a Kotlin `data class` was defined in the common source in the multiplatform project. `data class` is needed to provide kotlinox serialization support.
- **KSP generation** A compiler plugin inspects the data classes and creates an interface with the same properties just for JS module.
- **Multiplatform compiling:** With the correct Kotlin code in all modules, the compiler can now compile the modules.

```

@TypeScriptInterface
@kotlinx.serialization.Serializable
data class TravelInfoDTO(
    val _id: String? = null,
    val hotel_id: String,
    val title: String,
    val text: String,
    val hotel: HotelLDT0? = null,
)

```

a)

```

export interface TravelInfoDTO {
    readonly _id: Nullable<string>;
    readonly hotel_id: string;
    readonly title: string;
    readonly text: string;
    readonly hotel: Nullable<HotelLDT0>;
}

```

b)

Figure 8.5. Transformation of Kotlin data class a) using KSP to TypeScript b).

8.2.4 Client Library

The JVM module also includes a client HTTP library that models unit actions of GraphQL. The model contains these actions: `registerHotel`, `login`, `modifyCategories`, `upsertTrip`, `upsertTravelInfo`, `getHotelData`

The following example 8.6 shows the mapping of the `upsertTrip` mutation to an object-oriented approach. It is a snippet from class `HotelActions`. The OOP approach significantly simplifies integration testing.

```

suspend fun modifyCategories(
    list: List<TripCategoryDTO>
): List<TripCategoryDTO> {
    return graphqlRequestTyped(
        query = "mutation (\$categories: [TripCategoryInput!]) {
            modifyTripCategories(categories: \$categories){
                _id name trip_ids
            }
        }",
        topLevelName = "modifyTripCategories"
    ){
        put("categories", json.encodeToJsonElement(list))
    }
}

```

Figure 8.6. Multiplatform client action example.

8.3 Frontend

This section briefly summarizes frontend state management, UI optimization for different screens, screenshots of the final product, and highlights from the actual implementation. The frontend comprises mainly of standard Vue 3 code which is well described in the documentation⁵; therefore, any Vue specific problems are not included.

8.3.1 State Management

Pinia⁶ is the recommended state management package for Vue. We will focus on the network communication since the actual state management is done by Vue Framework⁷. The state was split into two modules:

⁵ <https://vuejs.org/guide/introduction.html> Introduction to Vue

⁶ <https://pinia.vuejs.org/introduction.html> Introduction to Pinia

⁷ <https://vuejs.org/guide/scaling-up/state-management.html> Introduction to Vue state management

- **Data module:** manages travel data that are displayed on the site. The entities reflect the entity relationship diagram 4.2.
 - Hotel data
 - TripCategories
 - Travel information
 - Trips
- **User module:** handles authentication and data editing. Data in this module is preserved over the browsing session using `window.sessionStorage`
 - Authentication
 - Authenticated user

The `data` module is initialized with one GraphQL query containing all the data the application will need for a visiting guest user on the startup. This situation is depicted in diagram 8.7 before the `User Browsing` section.

The `user` module holds the authenticated user (admin) and JWT token. If a token is present, the whole site is set into an administrative mode.

The following diagram 8.7 shows the editing of a trip using a popup B.1. The popup needs additional information, such as which trip categories are available. Before the dialogue is displayed, the required data is fetched from the backend. Most of the other popups follow this schema.

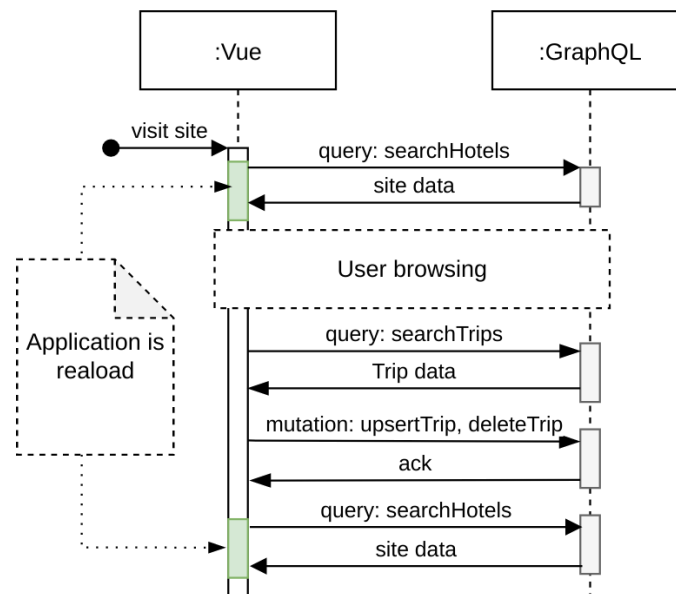


Figure 8.7. State management of `EditTripDialog`

8.3.2 Navigation

This section describes the thinking behind choosing UI/UX approach. Designing a good quality UI / UX is complex and could be extracted into separate research. The page structure follows the *single page* requirement. This structure keeps the consistency between the kiosk and the onboarding website:

- **Kiosk:** Kiosk navigation must be straightforward and shallow. Users do not spend much time in front of kiosks searching for nested information, for example, mall navigation kiosks. Popups are ideal because there are no forward and backward navigation buttons in the kiosk, only a touch screen.

- Onboarding web:** There are two cases to cover, mobile and desktop versions. Popups are not a good option for the desktop version and especially mobile. Content in our application is meant to be very simple; thus, displaying each travel info or activity on another page breaks the UX flow. The user would have to navigate from page to page to get a small amount of additional information. For MVP, navigation will be implemented in popups; it is likely to change in the future.

Following HTA diagram 8.8 shows which navigation actions a user can take on the onboarding site depending on privileges.

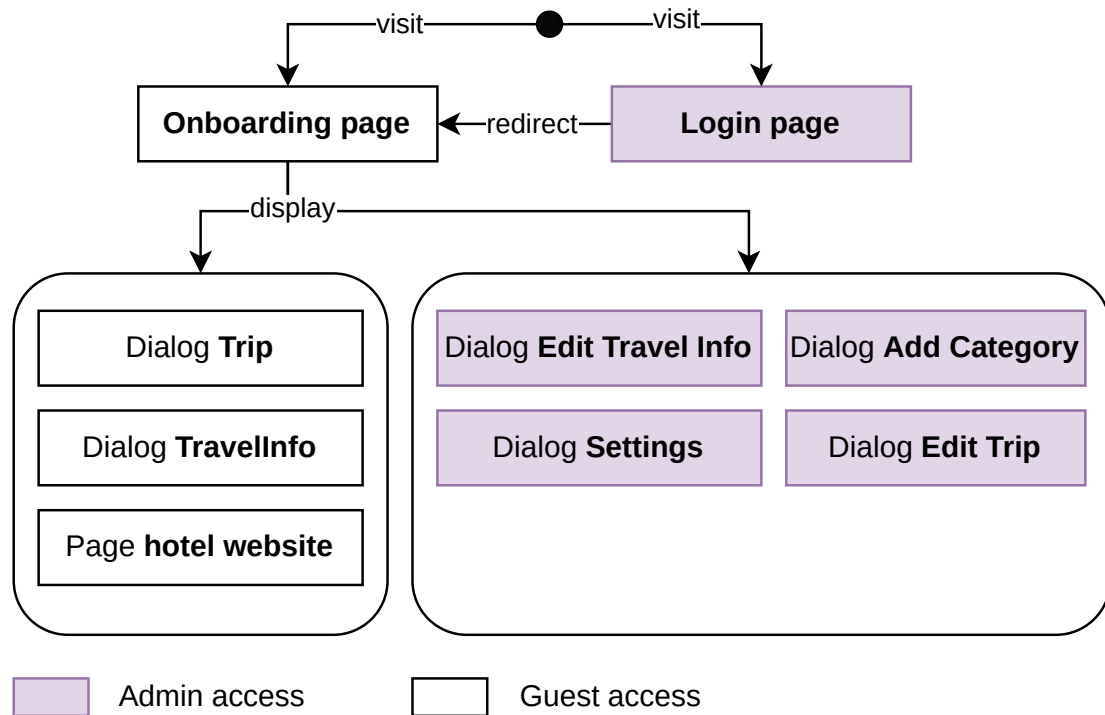


Figure 8.8. HTA diagram of user interface.

8.3.3 Onboarding Screen

The following screenshot depicts the onboarding site. There is contact information with the brand logo, name and a link to the official website at the top of the page. It is followed by rich text information about the hotel with an introduction image.

There is travel information in the middle of the page — The core advice for a guest about the location and attractions. At the bottom of the page, there are switchable categories with trips. By clicking on the category, the trips are replaced accordingly.

Penzion U Veselých
Informace pro Hosty

Karta pobytu Prozkoumej

Oficiální stránky ubytovatele

774 406 784

veselypenzion@email.cz

Informace o ubytování

Snídaně
Snídaně 8:00 - 9:30 formou bufetu. Prosíme hosty, aby neodnášeli jídlo ze snídaně na pokoje

Obědy, večere
V penzionu nepodáváme. Doporučujeme navštívit hotel Atlas nebo penzion Relax v centru Velké Úpy

Bar
Veškerá nabídka baru je na objednávku do domácích. Nápoje Vám přineseme do kulečnicku i na terasu

Wellness
Wellness je na objednávku, začínáme v celou hodinu v rozmezí 16-21:00. Minimální počet osob 2, maximální počet osob 5. Cena za hodinu pro 2 osoby 400Kč, pro 3-5 osob 600Kč. Děti mladší 10 let mohou být ve vířivce jen pod dohledem rodičů.

Fitness

Nápady pro vaši dovolenou

TourPAS – využijte lanovky bez omezeného počtu jízd

TourPAS je 1 - 4 denní jízdenka, která Vám dovolí využívat lanovky na Černou horu, Portášky a Hnědý vrch bez omezení za paušální cenu. Výhodné pro rod

Návštěva Sněžky lanovkou

Návštěva vrcholu Sněžky patří k základním bodům každého pobytu. Na Sněžku vede lanová dráha složená ze dvou úseků. Horní úsek je často mimo provoz z d

Relax Park v Peci pod Sněžkou

V Peci pod Sněžkou cestou na lanovku na Sněžku je umístěna bobová dráha o délce 900m a areál Lemurie pro malé i velké děti plný prolézaček, dlouhých a

Jízda na koloběžkách

Můžete si vypůjčit koloběžku a jet z Portášek naproti penzionu 5 km nebo z Černé hory 10,5 km dolů. Půjčovny koloběžek jsou na dolních i horních stani

Vyberte si podle

Autem v okolí

Turistika

Atrakce

Turistika

Pro děti

U Veselých
24. 01. 2022

Pevnost Stachelberg 20km

Pojedete asi 20 km autem, nejprve směrem dolů, před Trutnovem uhnete doleva na Žacléř, až se po stoupání dostanete nad obec Babi na parkoviště. Odtud

Hory Děti

U Veselých
24. 01. 2022

Adršpašsko-teplické skály 42 km

Pojedete autem do Trutnova, zde na 3 kr. objezdu doleva a dál směrem na Adršpach. V obci Chvalec se dáte doleva. Aby jste zaparkovali u skal, je nutné

Hory Děti

U Veselých
24. 01. 2022

Safari park Dvůr Králové 41 km

Pojedete směrem na Hradec Králové a v obci Kocbeře uhnete doprava na Dvůr. Pojedete podle šipek až k ZOO a za vstupní bránou je nejbližší parkoviště

Hory Děti

U Veselých
24. 01. 2022

Zámek Kuks 50km

Pojedete směrem na Hradec Králové a v obci Kuks uhnete doprava na parkoviště. Zámek Kuks je proslulý zejména svým exteriérem s velkolepými sochami

Hory Děti

Figure 8.9. The onboarding page screenshot.

8.3.4 Trip Screen

This screen 8.10 shows a TripPopup, which opens after clicking on a trip in a category. There is an introduction image and a small image gallery. The text can be edited in a rich text editor.

The screenshot shows a web application interface for 'Penzion U Veselých'. A modal window titled 'Adršpašsko-teplické skály 42 km' is open, displaying a large main image of a train track winding through a rocky landscape. Below the main image is a gallery of smaller images showing the rock formations and a cave entrance. The text in the popup provides details about the location, travel route, and ticket information.

Adršpašsko-teplické skály 42 km

Hory Děti

Pojedete autem do Trutnova, zde na 3 kr. objezdu doleva a dál směrem na Adršpach. V obci Chvaleč se dáte doleva. Aby jste zaparkovali u skal, je nutné si koupit online vstupenky s rezervací parkování, o prázdninách doporučujeme min 2-3 dny předem. Jinak se vrátíte bez prohlídky.

Adršpašsko-teplické skály jsou největší a nejmonumentálnější skalní město v Čechách. Jeho návštěva, zejména mimo hlavní sezónu je opravdovým zážitkem pro malé i velké návštěvníky. Dětskou pozornost přitáhne jak nejroztodivnější skály tak i možnost hrabat se a stavět z písku. Základní okruh v Adršpachu měří 3,5 km a trvá přibližně 3 hodiny. Návštěvu můžete zpestřit plavbou na lodičkách.

Pokud bude v Ádru plno, jedte dále asi 3 km a zastavte u parkoviště před Teplickými skalami. Zde na Vás čeká základní okruh 6 km s monumentálními Chrámovými stěnami o výšce až 60m. Spojit můžete i oba okruhy. Jestliže se nemůžete nabažit těchto pískovců, můžete dále navštívit další zajímavá místa: Broumovské stěny nebo Ostaš.

Zajímavé jsou i Jiráskovy skály se zbytky hradu na pískovcových věžích. Odtud je liduprázdný okruh přes rozhlednu Čáp a bludiště skal zpět k autu. S návštěvou hradu je 12 km.

Vstupné variabilní, zhruba: dospělí 180 Kč rodníne 2+2 500 Kč

Parkování: 150 Kč

Kocbeře uhnete doprava na Dvůr. Pojedete podle šípek až k ZOO a za vstupní bránou je nejbližší parkoviště

Kuks uhnete doprava na parkoviště. Zámek Kuks je proslulý zejména svým exteriérem s velkolepými sochami

Hory Děti

Figure 8.10. Trip popup Screen.

8.3.5 Touchscreen Kiosk

The following picture 8.11 shows the touch screen powered by Raspberry Pi. The screen will be mounted on a wall.



Figure 8.11. Picture of the touchscreen kiosk.

Chapter 9

Kiosk - Hardware / Software

Buying a standalone kiosk without software is a matter of tens of thousands of Czech crowns, which was not acceptable for the kiosk use. A much cheaper option is to buy a regular Desktop PC screen and connect it to a mini PC such as a Raspberry Pi. Hardware requirements are low except for running a single Chromium tab.

9.0.1 Screen

The main factors for the screen:

- Durability - The kiosk will be placed in a hallway, and frequent touching can damage the touchscreen.
- Low power consumption - The kiosk will run 16 hours a day.

The 22 ViewSonic TD2223 Screen was chosen due to its very affordable price and big frame around the screen, which should protect it.

9.0.2 Computing Unit

There are several requirements the computing unit must meet:

- WiFi support - There is no ethernet outlet in the intended location.
- Low power consumption - The kiosk should run 16 hours a day.
- Easy cable management - There will not be any casing in the early prototype.
- Linux - Linux is easily configurable as a kiosk.

An ideal mini PC was selected, Raspberry Pi 4. It is straightforward to use with its distribution which provides compatibility to all its features should this project extend.

Another factor was power boot. The lack of a power switch allows plugging the mini PC and the screen into a socket timer to control the operation time.

9.1 Operating System

A Raspberry Pi OS Lite was chosen as an underlying operating system because it is bare bone Linux installed with all Raspberry Pi drivers. Manual installation of all necessary software for the kiosk, such as the Display server, produces a much cleaner installation.

9.1.1 Touch Screen Correction

The kiosk orientation is portrait. The screen is configurable to the portrait mode; however, the touch screen did not adjust its orientation, and all inputs were upside down and left and right switched. Since the Raspbian OS use X.org as a display server, The Coordinate Transformation Matrix was set to compensate for the actual orientation with a matrix composed of up and down and left and right rotation matrix.

```
xrandr -o right
```



```
xinput set-prop "iSolution multitouch" \
    'Coordinate Transformation Matrix' 0 1 0 -1 0 1 0 0 1
```

9.1.2 Autorun Configuration

We need to install chromium and a Linux display server:

```
sudo apt-get install --no-install-recommends xserver-xorg
    x11-xserver-utils xinit xinput openbox chromium-browser
```

Next we need to set up an autorun script:

```
xset -dpms          # turn off display power management system
xset s noblank      # turn off screen blanking
xset s off          # turn off screen saver

# Remove exit errors from the config files that could trigger a warning
sed -i 's/"exited_cleanly":false/"exited_cleanly":true/'
    ~/.config/chromium/'Local State'

sed -i 's/"exited_cleanly":false/"exited_cleanly":true/;
    s/"exit_type": "[^"]\+"/"exit_type": "Normal"/'
    ~/.config/chromium/Default/Preferences

# Rotate screen to the right
xrandr -o right

# Rotate the touch screen to the right
xinput set-prop "iSolution multitouch"
    'Coordinate Transformation Matrix' 0 1 0 -1 0 1 0 0 1

# Run Chromium in kiosk mode
chromium-browser --noerrdialogs --disable-infobars --kiosk
    $KIOSK_URL --check-for-update-interval=31536000
```

Set kiosk website: `sudo nano /etc/xdg/openbox/environment`

```
export KIOSK_URL=https://example.com
```

Run script on user login: `sudo nano ~/.bash_profile`

```
[[ -z $DISPLAY && $XDG_VTNR -eq 1 ]] && startx -- -nocursor
```

Remove the splash screen from raspberry: `sudo nano /boot/config.txt`.

```
disable_splash=1
```

Remove Linux startup screen by appending this to `sudo nano /boot/cmdline.txt`

```
consoleblank=1 logo.nologo quiet loglevel=0 plymouth.enable=0
    vt.global_cursor_default=0plymouth.ignore-serial-console splash
    fastboot noatime nodiratime noram
```

9.2 Cost of Operation

The cost of operation is an essential factor. The screen has a typical consumption of 12.5W¹, the Raspberry Pi 4.3W² - Extended to a full year and 16 hours a day with a price of 6 Kč kWh, makes around 700 Kč per year.

It is important to note that the product was supposed to be deployed on Raspberry Pi. It is now deployed on a virtual private server on Wedos, which costs 1200kč per year.

¹ <https://www.viewsonic.com/ap/products/lcd/TD2223> Screen power consumption.

² <https://www.pidramble.com/wiki/benchmarks/power-consumption> RP pi power consumption.

Chapter 10

Deployment

This chapter will describe deploying our minimum viable product to a Wedos virtual private server as depicted on 10.1.

Originally this product was supposed to be fully deployed on the Raspberry Pi. However, the ISP provider did not offer us a public IP. The reason was to cut down the costs to a bare minimum. With the VPS, there are no problems with performance and bandwidth limitations.

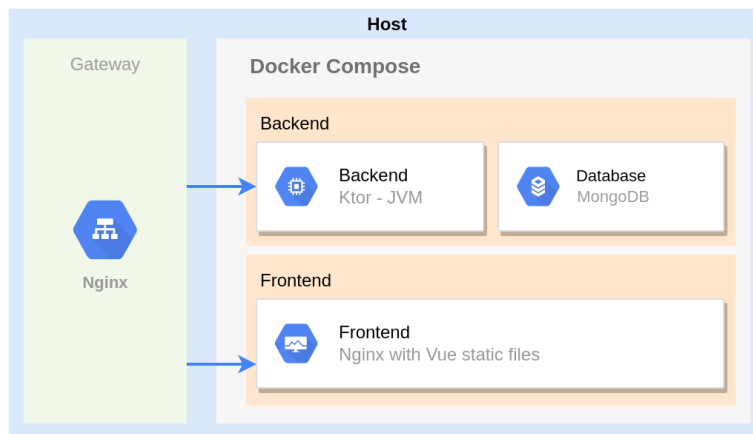


Figure 10.1. Deployment diagram

10.1 Pipeline

Setting up a fast turnaround pipeline is essential for speeding up development. Every action must be automated so that the developer can focus only on improving the application. Since this project is hosted on the faculty's GitLab, we can leverage GitLab's continuous integration pipeline and Docker infrastructure.

The GitLab's pipelines run on GitLab Runner, a service hosted outside of GitLab. The faculty offers a shared runner, but the waiting queue is too long — a custom shared runner was set up on Wedos for this project. The runner was set up to run the pipeline in a docker environment so `dind` image could be used. `Dind` is Docker image that can be run in Docker. In this way, we can access Docker build tools in our pipeline.

Another pre requisition was a `container registry` to store our production images. Gitlab offers this as well, but an application to SVTI is required, so I defaulted to using Docker Hub.

10.1.1 Stages

- **Build:** This stage takes our project repository with backend and frontend source code and builds appropriate docker images, and pushes them into the docker hub to be used in the deployment stage.

- Job: Build frontend
- Job: Build Backed
- **Deploy:** This stage established an SSH connection to our deployment server. On the server, pulls new images from Docker Hub and redeploys them using Docker Compose
 - Job: Deploy

10.2 Docker Images

The following two sections describe docker images 10.2, 10.3 used for deploying our application.

10.2.1 Vue frontend

Deploying Vue.js has two stages the build stage and the deploy stage. in the first phase Vue.js is built using Vite. The built code is copied into Nginx with configuration to rewrite URLs to support browser history mode.

```
FROM node:lts-alpine as build-stage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# production stage
FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
COPY prod_nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Figure 10.2. Dockerfile for Vue.

10.2.2 Ktor Backend

Deploying Ktor has similar steps as Vue.js. First, Ktor is built with a `shadow` plugin, which produces a fat jar with all dependencies in a single JAR using Gradle Docker image. Then it is run with an open JDK image.

```
FROM gradle:7-jdk11 AS build
COPY --chown=gradle:gradle . /home/gradle/src
WORKDIR /home/gradle/src
RUN gradle :ktor:shadowjar --no-daemon

FROM openjdk:11
EXPOSE 8080:8080
RUN mkdir /app
COPY --from=build /home/gradle/src/ktor/build/libs/*.jar
  /app/ktor-docker-sample.jar
ENTRYPOINT ["java", "-jar", "/app/ktor-docker-sample.jar"]
```

Figure 10.3. Dockerfile for Ktor.

10.3 Docker compose

The actual deployment is done using a plain Docker Compose file. The file diagram is depicted on figure 10.4. It includes three services: Mongo Database, Ktor Backend, and Vue Frontend.

The only data that must be persisted is the Mongo database. External volume to Mongo DB data folder was created and moved out of the container to ensure data persistence when the application is redeployed.

A separate backend network was created not to expose Mongo DB outside the Docker compose deployment.

The configuration is minimal but sufficient for the load expected. The application expects at most 30 visitors per day people in the early stage of a project.

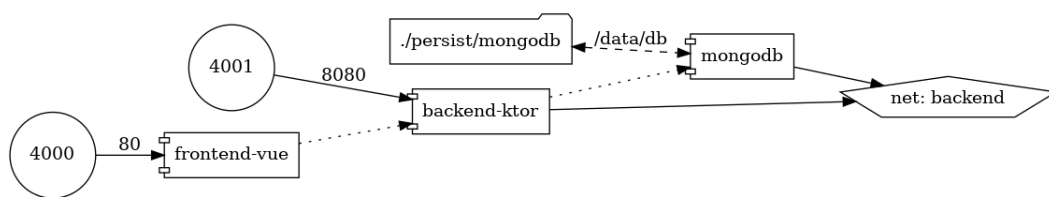


Figure 10.4. Service deployment diagram

10.4 Reverse Proxy

On the actual VM, there is a statically installed Nginx. This Nginx accepts all connections to the VM since there are multiple other services. For our domain, the Let's Encrypt bot was configured.

- Redirects `travel-info.cz/api` to port 4001 - Backend
- Redirects `travel-info.cz/` to port 4000 - Frontend

Chapter 11

Testing

This chapter describes 4 types of testing performed on this application: unit testing, integration testing, end to end testing and user testing.

The application mainly comprises CRUD (Create, Read, Write, Update) operations on the data layer and GraphQL communication. There is not much business logic to test except input validation; therefore, testing focuses more on the correctness of GraphQL API and its connection to the frontend. The following diagram 11.1 depicts the application architecture layers [todo REF] and test levels that cover them.

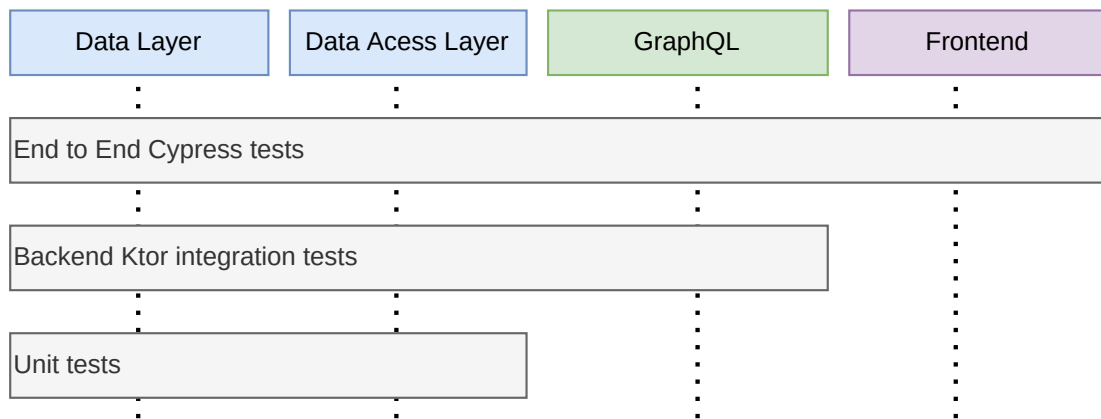


Figure 11.1. Test coverage of layered architecture.

11.1 Test Goals

- Test CRUD operations on the data model
- Test Security (Unauthenticated modification, data access)
- Test integration between frontend and backend
- User Kiosk Testing

11.2 Backend Testing

„Ktor provides a special testing engine that doesn't create a web server, doesn't bind to sockets, and doesn't make any real HTTP requests. Instead, it hooks directly into internal mechanisms and processes an application call directly. This results in quicker tests execution compared to running a complete web server for testing.“ [30]

The Ktor framework provides special integration test DSL called `testApplication`. It runs a new application for each test. This DSL was extended to clear the database just after the startup. The test usually requires application in a specific initial state, such as users are already registered or have some initial seed data. For this, we can use our multiplatform client 8.2, example 11.3.

The following list contains integration tests with respect to the entities. Some tests from the CRUD matrix were grouped into a single test since most of the create and update operations are implemented as a single `upsert` operation.

- **Authentication:** Test the GraphQL directive `@AuthHotelDirective`.
 - TEST: authenticated modification of categories
 - TEST: unauthenticated modification of categories
- **Hotel:** Test CRUD operations on `HotelEntity`
 - TEST: prevent trip insertion to another hotel
 - TEST: upsert categories
 - TEST: update hotel settings
- **TravelInfo:** Test CRUD operations on `TravelInfoEntity`
 - TEST: upsert, delete `TravelInfo`
- **Trip:** Test CRUD operations on `TripEntity`
 - TEST: upsert and modify trip
 - TEST: delete trip

A code coverage plugin was added to evaluate what portion of code runs in all tests. It does not indicate if the code is correct. The more code is run, the higher the probability that a bug is discovered. Therefore we try to achieve as extensive code coverage as possible. The following table 11.2 contains the code coverage from integration testing. However, the results might not be entirely correct. The official Kotlin coverage tool *Kover*¹ is in beta version 0.5, and there are cases where results are incorrectly computed.

	Class, %	Method, %	Branch, %	Line, %	Instruction, %
cz.cvut.veselj57.dt	44.4% (4/9)	52.2% (12/23)	50% (5/10)	96% (169/176)	98.2% (2519/2565)
cz.cvut.veselj57.dt.api	76.5% (13/17)	61.9% (13/21)	20.6% (7/34)	55.3% (21/38)	52.5% (379/722)
cz.cvut.veselj57.dt.entities	100% (15/15)	95% (19/20)	59.4% (60/101)	97.9% (46/47)	77.9% (1574/2020)
cz.cvut.veselj57.dt.graphql	85.7% (6/7)	76.5% (13/17)	50% (3/6)	56.5% (26/46)	59.6% (226/379)
cz.cvut.veselj57.dt.graphql.directives	100% (1/1)	100% (3/3)	100% (2/2)	100% (8/8)	100% (59/59)
cz.cvut.veselj57.dt.graphql.exceptions	50% (1/2)	50% (1/2)		50% (1/2)	50% (3/6)
cz.cvut.veselj57.dt.graphql.model	63.6% (7/11)	42.1% (8/19)	0% (0/17)	31.1% (14/45)	15.1% (126/836)
cz.cvut.veselj57.dt.graphql.mutations	66.7% (4/6)	61.5% (8/13)	28.9% (13/45)	58.4% (45/77)	44.8% (468/1044)
cz.cvut.veselj57.dt.graphql.queries	100% (3/3)	100% (5/5)	50% (1/2)	100% (7/7)	97.7% (85/87)
cz.cvut.veselj57.dt.graphql.security	100% (3/3)	100% (3/3)		100% (3/3)	100% (18/18)
cz.cvut.veselj57.dt.persistence	100% (3/3)	75% (6/8)	50% (2/4)	100% (21/21)	100% (187/187)
cz.cvut.veselj57.dt.plugins	85.7% (24/28)	78.1% (25/32)	0% (0/13)	84.4% (54/64)	58.6% (290/495)
cz.cvut.veselj57.dt.services	100% (3/3)	58.3% (7/12)	66.7% (8/12)	98% (50/51)	99.9% (802/803)

Figure 11.2. Code coverage table.

¹ <https://github.com/Kotlin/kotlinx-kover> Kover Github page.

```

val hotelActions = HotelActions(client)
hotelActions.registerHotel(hotel)
hotelActions.authorize(credentials)
val tripId = hotelActions.upsertTrip(data)._id
assertEquals(1, hotelActions.getTrips(listOf(tripId)).size)
hotelActions.deleteTrip(tripId)
assertEquals(0, hotelActions.getTrips(listOf(tripId)).size)

```

Figure 11.3. Using Multiplatform client for testing.

11.3 Frontend Testing

The last part of testing is to verify the correct implementation between frontend user actions, such as adding a trip and GraphQL API. There were two considered options. The first option was to mock the input and responses in the browser; the other option was to run the backend alongside the frontend and implement complete End to End testing. Mocking the responses would not verify the correctness of the GraphQL queries; therefore, developing full E2E is favourable for testing the whole application.

There are two main frameworks for frontend testing Selenium and Cypress. „Cypress was originally designed to run end-to-end (E2E) tests on anything that runs in a browser. A typical E2E test visits the application in a browser and performs actions via the UI just like a real user would.“[31] End to End tests were implemented using Cypress to keep our frontend stack in one programming language — Typescript.

- **General setting:**
 - TEST: Edit hotel settings
- **Test TravellInfo:**
 - TEST: Add new TravelInfo
 - TEST: Delete TravelInfo
- **Test Trip:**
 - TEST: Add new Trip
 - TEST: Delete Trip
- **Test TripCategory:**
 - TEST: Add new TripCategory
 - TEST: Delete TripCategory

The following example shows and Cypress test 11.4 Add new TripCategory for illustration purposes. It opens a popup for editing categories, types a name, saves the new category and checks if the result exists.

```

it('Add new TripCategory', () => {
  cy.get("#add-new-trip-category").click()
  cy.get("#popup-edit-category").should("exist")
  cy.get("#popup-edit-category-name").type(name)
  cy.get("#popup-edit-category-submit").click()
  cy.contains(".categories", name).should("exist")
})

```

Figure 11.4. Cypress test.

11.4 Small Usability Testing

This small usability survey evaluates the quality of the content and possible future improvements. The recommended number of participants is 5, according to Jakob Nielsen and Thomas K. Landauer[32]. The usability survey was conducted as an interview. The kiosk and the online mobile version of this product were briefly introduced before the interview. A more thorough analysis of the benefits of this product will be done after it is officially released and used in production, approximately in fall 2022.

11.4.1 Survey

- **Platform preference:** Which version of the system should be the priority.
 - Which version of the information system do you find more prompting to explore in the hotel, the touch screen kiosk or the online onboarding website?
- **Information quality:** This question collects what kind of travel information we should further include in the application.
 - What kind of trips, trip categories and accommodation information are you missing in the application?
- **Other information:** This question collects future nice to have features which would bring additional benefits to the guests.
 - What other information you would like to see in the system?

11.4.2 Results

- **Platform preference:**
 - 5/3 answered that the on-site touch screen kiosk is preferable.
- **Information quality:**
 - Trail map is missing
 - Too much nonformatted text
 - Not enough trips
 - Missing aquapalace activity
 - What to do on the first day
 - Information about local restaurants are missing
- **Other information:**
 - Weather forecast
 - Wellness reservation option
 - QR link to the official website
 - Add page with with a tourist map (mapy.cz)
 - Add indicator if an activity is free or not
 - Extract common attributes, such as opening hours, prices and highlight them in UI

There were three excellent remarks which will be added in future versions:

- Weather forecast
- Page with a tourist map
- Information about local restaurants

11.5 Testing Summary

The application was tested on all program levels using unit tests, integration tests and end to end frontend tests using Cypress. The purpose was to catch refactoring bugs when new features were being added. They helped verify that no features broke other features.

A small usability survey uncovered 2 missing features (Missing weather forecast and tourist map) and 1 important data quality issue (Missing dining options).

Chapter 12

Project Future

The project will be deployed and maintained in Penzion U Veselých for at least one year to collect insight into how guests search for travel information.

- Maintain the project.
- Generate more travel content for the system.
- Create QR leaflets that will be hung on the guest room door to increase the number of visits.
- Add tools to collect primary statistical data such as the number of views on a trip or total time spent in session.
- Evaluate the actual benefit after one year in production.

The ultimate goal is to build a website where hotel owners will put non-payable travel content in exchange for free advertisement. It was mainly described in the section future of travel platforms 3.6.

- Redesign the UI / UX from the collected feedback.
- Create a website where hotels will be able to register and participate in generating travel content.

Chapter 13

Conclusion

This thesis completed the assignment by first analyzing the problem, proposing a solution, validating the idea by user testing and implementing the minimum viable product.

The 2 chapter briefly described a minimum viable product and why it is essential for product development. The following chapter 3, outlined communication between guests and hospitality providers regarding travel information and typical user groups to describe the target audience and understand their pain points regarding holiday planning. An information system consisting of an on-site touchscreen kiosk and an onboarding website was designed to address those pain points in chapter 4.

Two product validation user tests using a high fidelity Figma prototype were designed in chapter 5 to verify that guests benefit from the system. The validation test showed a need for this product from the guest's point of view to gain travel information and from the hotel's point of view to decrease the time to serve the guests.

MVP concept was formally described using functional and nonfunctional requests in chapter 4. Chapter 6, 7 analyzed the software and architectural approach. The analysis determined that the web is the ideal technology for both the kiosk and the onboarding microsite. The application was implemented using a monolithic approach using the Ktor framework on the backend and Vue 3 on the frontend, chapter 8.

Chapter 9 described hardware setup and required adjustments in software to make Raspberry Pi work as a kiosk. A production environment with a deployment pipeline was set up on virtual private server, chapter 10.


The final product underwent integration testing and end to end testing using cypress 11. A small user survey of 5 guests evaluated the quality of content and proposed some future features.

In the summer of 2022, the MVP will be deployed in Penzion U Veselých. The plan is to add more content and analyze long-term benefits for the guests. The final user survey discovered additional useful features, such as adding an interactive map with trails. Those features will be added if the system proves to be viable.

References

- [1] *Tourism Satellite Account*. Czech Statistical Office. 2020.
https://www.czso.cz/csu/czso/satelitni_ucet_cestovniho_ruchu.
- [2] *2022 State of the Hotel Industry Report In Collaboration With Accenture*. The American Hotel & Lodging Association. 2022.
<https://www.ahla.com/sites/default/files/AHLA%20SOTI%20Report%202022%201.24.22.pdf>.
- [3] Eric Ries. *The lean startup : how constant innovation creates radically successful businesses*. London; New York: Portfolio Penguin, 2011 . ISBN 9780670921607 0670921602.
http://www.amazon.de/The-Lean-Startup-Innovation-Successful/dp/0670921602/ref=sr_1_2?ie=UTF8&qid=1396199893&sr=8-2&keywords=eric+ries.
- [4] *Interní informace a statistiky*. Penzion U veselých. 2022.
<http://veselypenzion.cz/>.
- [5] *Relative market share of major online travel agencies (OTAs) in Europe in 2019*. Statista. 2019.
<https://www.statista.com/statistics/870046/online-travel-agency-ota-market-share-in-europe/>.
- [6] *Portál kudy z nudy.cz – fenomén mezi portály cestovního ruchu*. CzechTourism. 2022.
<https://www.kudyznudy.cz/o-kudy-z-nudy>.
- [7] *Play Store - Alfred Key*. Previo. 2022.
<https://play.google.com/store/apps/details?id=cz.previo.alfred&hl=cs&gl=US>.
- [8] *Connect your application to Booking.com*. Booking.com. 2022.
<https://connect.booking.com/>.
- [9] *Price List for Previo*. Previo. 2022.
<https://www.previo.cz/cenik-systemu-previo>.
- [10] *What is Prototyping?* Interaction Design Foundation. 2022.
<https://www.interaction-design.org/literature/topics/prototyping>.
- [11] Avroora Shuhalii. *What is the difference between low and high fidelity prototypes?* Medium. 2020.
<https://bootcamp.uxdesign.cc/what-is-the-difference-between-low-and-high-fidelity-prototypes-b1f3612f85f7>.
- [12] *Visualize user behavior*. Hotjar. 2022.
<https://www.hotjar.com/product/heatmaps/>.
- [13] *Meet Django*. Django Software Foundation. 2022.
<https://www.djangoproject.com/>.
- [14] Adrian Scholes. *Apps That Need ACID*. Volta Active Data. 2016.

- [15] *The 2021 State of JS survey*. State Of JavaScript. 2021.
<https://2021.stateofjs.com/en-US/>.
- [16] *RESTful web API design*. Microsoft. 2022.
<https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design>.
- [17]
- [18] *GraphQL Experience Over Time*. State Of JavaScript. 2019.
<https://2019.stateofjs.com/data-layer/graphql/>.
- [19] Kyle Brown. *On Architectural Testability*. Medium. 2019.
<https://kylegenebrown.medium.com/on-architectural-testability-4078459c5a90>.
- [20] Kacey Bui. *Introduction to Event-Driven Architecture*. Medium. 2021.
<https://medium.com/microservicegeeks/introduction-to-event-driven-architecture-e94ef442d824>.
- [21] *What is event streaming*. Medium. 2022.
<https://kafka.apache.org/intro>.
- [22] Joan Goal. *Email Scheduling Application with JobRunr*. Medium. 2021.
<https://medium.com/edreams-odigeo-tech/email-scheduling-application-with-jobrunr-d37fd18b4898>.
- [23] Sam Newman. *Building microservices*. Second edition ed.. Beijing: O'Reilly, 2021. ISBN 978-149-2034-025.
- [24] Ramesh Mhetre. *When to choose Microservices Architecture Over Monolithic? Why?* Medium. 2018.
<https://medium.com/@mheteramesh/when-to-choose-microservices-architecture-over-monolithic-why-794aed04d8db>.
- [25] *GraphQL Kotlin*. Expedia Group. 2022.
<https://opensource.expediagroup.com/graphql-kotlin/docs/schema-generator/customizing-schemas/directives/>.
- [26] *GridFS Documentation*. Mongo DB. 2022.
<https://www.mongodb.com/docs/manual/core/gridfs/>.
- [27] *Base64 - Encoded size increase*. Mozilla MND. 2022.
https://developer.mozilla.org/en-US/docs/Glossary/Base64#encoded_size_increase.
- [28] *Kotlin Multiplatform use cases*. Kotlin Foundation. 2022.
<https://kotlinlang.org/docs/multiplatform.html>.
- [29] *Kotlin Symbol Processing API*. Kotlin Foundation. 2022.
<https://kotlinlang.org/docs/ksp-overview.html>.
- [30] *Testing Ktor Framework*. Kotlin Foundation. 2022.
<https://ktor.io/docs/testing.html>.
- [31] *Why Cypress*. Cypress.io. 2022.
<https://docs.cypress.io/guides/overview/why-cypress#What-you-ll-learn>.
- [32] Jakob Nielsen, and Thomas K. Landauer. *A Mathematical Model of the Finding of Usability Problems*. In: *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for



Computing Machinery, 1993. 206–213. ISBN 0897915755.
<https://doi.org/10.1145/169059.169166>.

Appendix A

Thesis Assignment



ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Veselý** Jméno: **Jan** Osobní číslo: **474405**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávající katedra/ústav: **Katedra počítačů**
Studijní program: **Otevřená informatika**
Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Informační systém pro prezentaci služeb hotelu a aktivit v okolí v rámci pobytu a objednávky

Název diplomové práce anglicky:

Information system for presentation of hotel services and activities in surroundings as a part of the onboarding process

Pokyny pro vypracování:

Seznamte se s problematikou informování hostů o možnostech trávení dovolené a produktového vývoje. Analyzujte jaké jsou nejdůležitější informace, které host potřebuje vědět. Navrhněte řešení [1,2], jak tyto údaje digitalizovat a prezentovat zákazníkovi v průběhu objednávky formou jednoduchého Event Driven systému [4]. Vytvořte MVP (minimum viable product), který zahrne digitální prezentaci služeb penzionu a aktivit v okolí, administraci z pohledu ubytovatele a komunikační kanál mezi ubytovatelem a zákazníkem. Otestujte aplikaci z hlediska softwarového návrhu a implementace (Stabilita, škálování). Změřte její přínos. (Uživatelské testování, návštěvnost) [2, 3].

Seznam doporučené literatury:

[1] OSTERWALDER, Alexander, Yves PIGNEUR a Tim CLARK. Business model generation: a handbook for visionaries, game changers, and challengers. Hoboken, NJ: Wiley, c2010. ISBN 9780470876411.
[2] OSTERWALDER, Alexander, Yves PIGNEUR, Gregory BERNARDA a Alan SMITH. Value proposition design: how to create products and services customers want. Hoboken: John Wiley, [2014]. Strategyzer series. ISBN 9781118968055.
[3] BLAND, David J. a Alexander OSTERWALDER. Testing business ideas. Hoboken, New Jersey: John Wiley & Sons, [2020]. ISBN 9781119551447.
[4] BELLEMARE, Adam. Building Event-Driven Microservices. 1. O'Reilly Media, [2020]. ISBN 9781492057895.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jiří Šebek kabinet výuky informatiky FEL

Jméno a pracoviště druhého(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **11.02.2022** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **30.09.2023**

Ing. Jiří Šebek
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

Appendix B

Screenshots

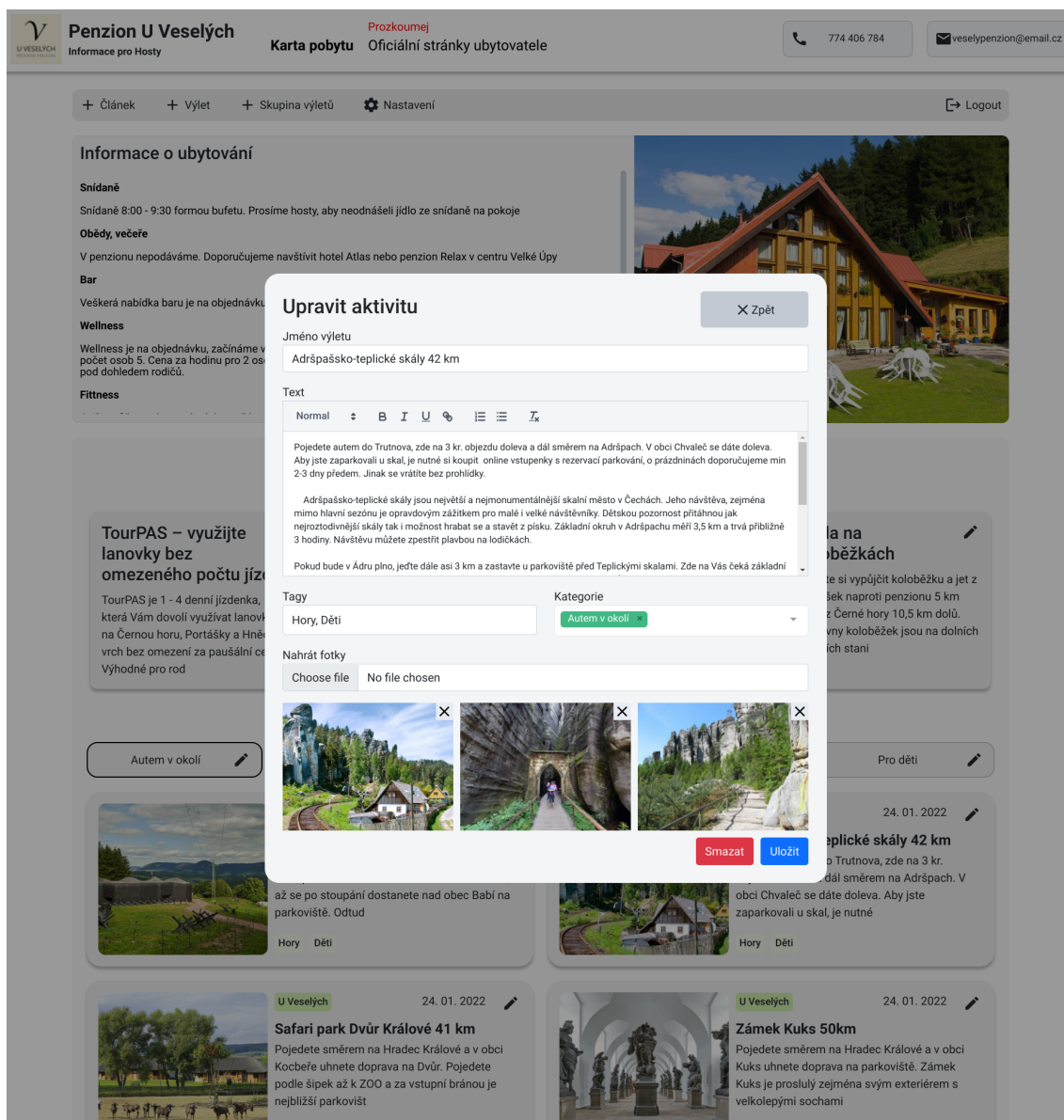


Figure B.1. Screen edit trip popup.

The screenshot displays the website for 'Penzion U Veselých'. The main header includes the logo, name, and contact information. A navigation bar at the top contains links for 'Článek', 'Výlet', 'Skupina výletů', and 'Nastavení'. The 'Nastavení' modal window is open, showing the following fields:

- Název zařízení:** Penzion U Veselých
- Oficiální stránky:** veselypenzion.cz
- Kontaktní email:** veselypenzion@email.cz
- Kontaktní telefon:** 774 406 784

The modal also includes a rich text editor for 'Informace o ubytování' with a toolbar (Normal, Bold, Italic, Underline, Link, Unlink, List, Unlist, Indent, Outdent) and a preview of the content. Below the editor are two file upload sections: 'Nahrát Logo' and 'Nahrát úvodní foto', both with 'Choose file' and 'No file chosen' options. An 'Uložit' (Save) button is located at the bottom right of the modal. The background shows a list of activities like 'Pevnost Stachelberg 20km', 'Adršpašsko-teplické skály 42 km', 'Safari park Dvůr Králové 41 km', and 'Zámek Kuks 50km'.

Figure B.2. Screen edit hotel settings.