GetawayGo

Project Plan

Fontys University of Applied Sciences

Date	**:**	**23/09/2024**
Version	**:**	**1.0**
State	**:**	**First draft**
Author	**:**	**Anna Kadurina**

Version history

Version	Date	Author(s)	Changes	State
1.0	23/09/2024	Anna Kadurina	Creation of the document	First draft

Distribution

Version	Date	Receivers
1.0	30/09/2024	Tülin Erçelebi Ayyildiz, Leon Schrijvers

Table of Contents

1. Project Assignment

1.1 Context

GetawayGo is a company focusing on providing a platform for travellers to find short-term accommodation worldwide, much like existing platform such as Airbnb and Booking.com. It will allow users to list, search and book properties while providing hosts with management tools.

1.2 Goal of the project

The goal of the project is to develop a scalable, user-friendly application for travellers and property owners. The aim is to create a platform thar will please the needs of both hosts and guests. The preferred situation is a fully functional application that connects travellers and hosts, providing both parties with a seamless experience for short-term stays. The advantages of this project are satisfying the travellers' needs, promoting tourism, and generating income for the owners of the properties. From an ICT point of view, the project offers a lot of possibilities, such as global scaling, user-centric designs and a lot of challenging functionalities such as external API integrations, AI integration, search features, filters, reviews and statistics.

1.3 Scope and preconditions

Scope

Inside Scope	Outside Scope
User Management	Integration with other travel services like car rentals or flights
Property Management	
Booking Management	
Analytics for property owners Service	
Property reviews to provide feedback	
Chat Service with AI integration	
Notification Service	
External API integration (Google Maps)	

Preconditions

- C# .NET backend
- React frontend

- Azure as a cloud platform
- Architecture with microservices
- Deadline: 18th of January, 2025

1.4 Strategy

The project will follow an agile approach using Scrum. This method is ideal for the dynamic nature of the project, allowing for regular feedback, iterative improvements, and rapid adaptation to changes. The Scrum approach can be easily adapted for a solo developer. By using this strategy, problems can be easily identified to improve the efficiency of the workflow.

1.5 Research questions

Main Research Question:

How can a scalable microservices-based platform effectively support short-term accommodation bookings while ensuring high performance, user-friendliness, and security for global travellers?

Sub-questions:

1. How can the platform dynamically scale to handle varying loads of users and bookings during peak times?
2. What architectural patterns can ensure high availability and fault tolerance in a microservices-based platform?
3. What security measures are necessary to protect sensitive user data on a global platform?
4. How can communication between different microservices be efficiently managed to avoid bottlenecks and ensure real-time data flow?
5. What strategies can be employed to effectively monitor and manage microservices to prevent service disruptions?
6. How can cloud-native tools be used to automate the scaling, deployment, and management of services to improve efficiency?

	Question	Strategies	Method(s)
Objective 1	How can the platform dynamically scale to handle	Library Research	Available product analysis, Best good and bad practices,

	varying loads of users and bookings during peak times?		Literature study, Expert Interview
Objective 2	What architectural patterns can ensure high availability and fault tolerance in a microservices-based platform?	Library Research, Workshop	Literature Study, IT architecture sketching, Expert Interview
Objective 3	What security measures are necessary to protect sensitive user data on a global platform?	Library Research, Lab Research, Showroom	Literature study, Security test, Guideline conformity analysis
Objective 4	How can communication between different microservices be efficiently managed to avoid bottlenecks and ensure real-time data flow?	Library Research, Lab Research	Literature study, System test
Objective 5	What strategies can be employed to effectively monitor and manage microservices to prevent service disruptions?	Library Research	Available product analysis, Literature Study
Objective 6	How can cloud-native tools be used to automate the scaling, deployment, and management of services to improve efficiency?	Library research, Field Research	Literature study, Available product analysis, Document analysis

1.6 End products

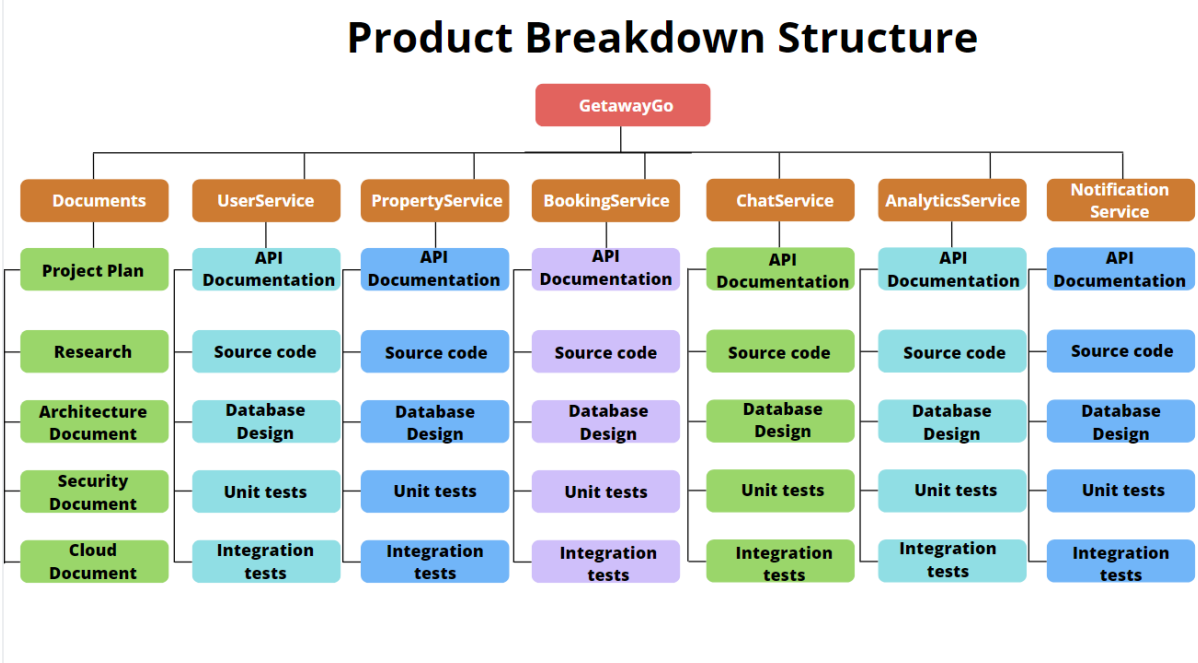

Figure 1 - Product Breakdown Structure of GetawayGo

2. Project Organization

2.1 Stakeholders and team members

Name	Abbreviation	Role	Availability
Anna Kadurina a.kadurina@student.fontys.nl	A.K	Developer	3 days a week
Leon Schrijvers l.schrijvers@fontys.nl	L.S	Stakeholder/Coach	2 days a week
Marc Grootel marc.vangrootel@fontys.nl	M.G	Stakeholder/Coach	2 days a week
Tülin Erçelebi Ayyildiz t.ercelebiayyildiz@fontys.nl	T.A	Stakeholder/Coach	2 days a week

2.2 Communication

The communication between the developer and the stakeholders will happen at the end of each sprint to get feedback on the end product, as well as throughout the sprint to discuss features and the progress of the application. All feedback sessions will be written down in Feedpulse to keep track of the communication and the progress of the project.

3. Activities and time plan

3.1 Phases of the project

GetawayGo project can be divided into several key phases:

- **Problem Analysis and Initial Research** – Understanding the code problem, analysing the requirements, conducting initial research on technology and user needs
- **System Design and Architecture Planning** – Designing the system architecture, including identifying the microservices, cloud infrastructure, and security measures
- **Implementation** – Developing the microservices and frontend
- **Testing** – Performing unit testing, integration testing, system testing, and security testing
- **Handover** – Submitting the completed project with all documentation

Disclaimer: Due to the Agile methodology employed in this project, phases may evolve or overlap as the project progresses.

3.2 Time plan and milestones

Time plan

Phasing	Effort	Start date	End date
Sprint 0 – Start of the project, Azure setup, Azure DevOps setup, UserService setup	Low	02/09/2024	22/09/2024
Sprint 1 – UserService Implementation, Documentation creation, PropertService setup, Frontend setup	Medium	23/09/2024	13/10/2024
Sprint 2 – PropertyService Implementation, BookingService setup, Documentation continuation, Frontend implementation	High	14/10/2024	10/11/2024
Sprint 3 – NotificationService, ChatService, AnatyticsService setup and implementation, Documentation continuation, Frontend implementation	High	11/11/2024	30/11/2024

Sprint 4 - NotificationService, ChatService, AnatyticsService implementation, Documentation continuation, Frontend implementation	High	02/12/2024	22/12/2024
Sprint 5 – Finishing up the project, Submission	High	23/12/2024	18/01/2025

Milestones

- Project Pitch Submission Deadline: 15/09/2024
- Design Oriented Research – Plan Submission Deadline: 29/09/2024
- First Portfolio Delivery: 06/10/2024
- Second Portfolio Delivery: 03/11/2024
- Third Portfolio Delivery: 08/12/2024
- Design Oriented Research Submission Deadline: 12/01/2025
- Final Portfolio Delivery: 19/01/2025

Roadmaps

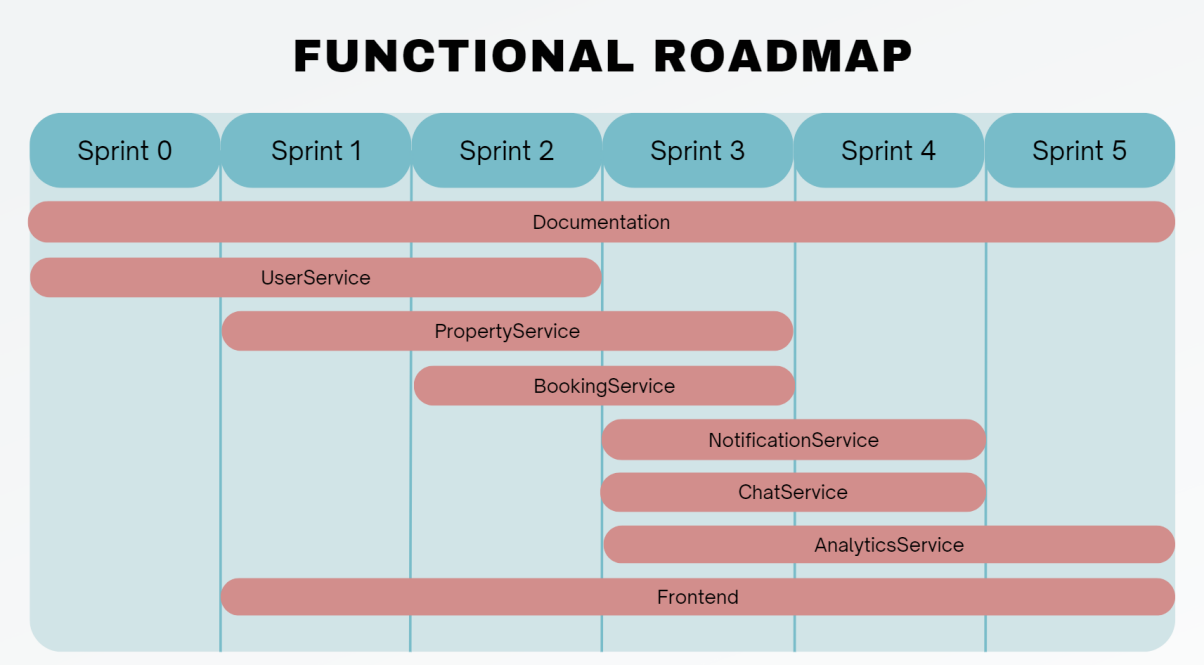

Figure 2 – Functional Roadmap of GetawayGo

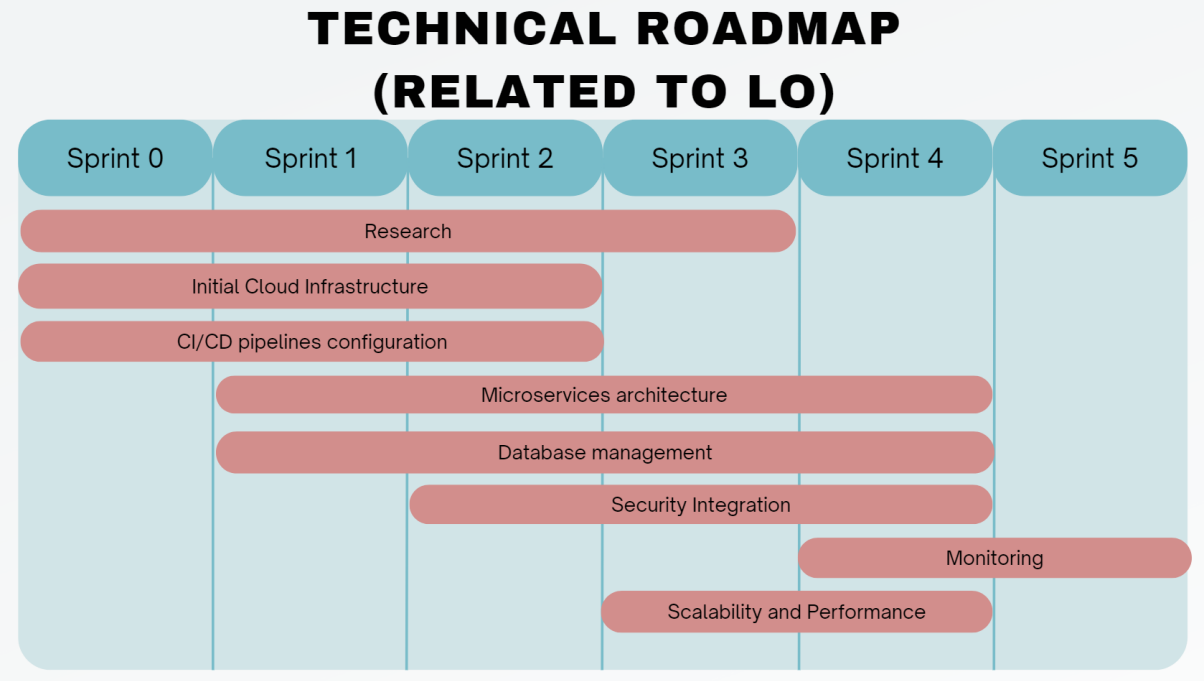

Figure 3 – Technical Roadmap of GetawayGo

4. Testing strategy and configuration management

4.1 Testing strategy

The testing strategy of the GetawayGo project will include multiple levels of testing: unit testing, integration testing, system testing, and security testing.

- Unit testing will focus on individual microservices to ensure that each service functions as expected. The goal is to achieve at least 80% code coverage with automated unit tests using tools such as xUnit for C#.
- Integration testing will verify that microservices work together correctly, especially in scenarios involving communication between services. These tests will focus on API calls, database interactions, and message queues.
- System testing will ensure that the overall platform functions as a cohesive unit, simulation real user scenarios to test features end-to-end.
- OWASP testing will also be incorporated to focus on the security of the application. These tests will focus on identifying vulnerabilities like SQL injection, cross-site scripting (XSS), and other common security risks, ensuring that the platform is secure.

The unit, integration and OWASP tests will be integrated within the CI/CD pipeline, ensuring they run with every new commit. For quality analysis, tools like SonarQube will be used to monitor code quality, security vulnerabilities, and technical debt.

4.2 Test environment and required resources

For the GetawayGo project, the test environment will primarily consist of two stages: a local development environment for implementing and testing during development, followed by direct testing in the production environment for final deployment and validation. The main reason for this is the limited scope and budget of the student subscription in Azure.

- Local Environment: The development and unit testing of individual microservices will occur in local environment. A local database will be used also. This setup will allow for thorough integration testing before deploying any changes. All tests will be ran locally to ensure the code meets the required standards.
- Production Environment: Once the local tests are successful, the code will be deployed to Production in Azure App Services. Production testing will include system testing to validate the correct functionality in real-world scenarios.

Since the project will use the cloud resources provided by the Azure Student Subscription, minimal additional hardware is required.

4.3 Configuration management

Version management for the project will be handled using Git, with Azure DevOps repositories being the primary tool. The branching strategy will follow GitFlow, with separate branches for features,development, and production releases.

- Feature branches will be used for individual features and will merge into the develop branch after passing unit and integration tests.
- The main branch will represent the production-ready state.

Change requests and bug reports will be managed via Azure DevOps' issue-tracking system, allowing for organized tracking of any changes, with each request linked to a specific branch and version control.

5. Finances and Risk

5.1 Project budget

The budget primarily relies on the $100 in Azure credits provided through the student subscription. This budget will be allocated toward essential cloud services such as hosting, storage, and databases, as well as any necessary scaling of microservices. The credits should be sufficient for the development and testing phases, especially with careful monitoring of resource usage and the adoption of cost-efficient services like serverless computing and free-tier options. Additional costs are not anticipated, but if more resources are needed, the project will prioritize optimizing usage within the given credits.

5.2 Risk and mitigation

Risk	Probability	Impact	Prevention activities included in plan	Fall-back activities
1 Sudden leave of a teacher	Low	Medium	Regular communication with all teachers; Documenting project progress and tasks	Contact alternate teacher; Seek help from other teachers
2 Communication problem	Very low	Extremely high	Scheduled weekly meetings with university mentor to ensure that the progress of the internship is clear; Daily meetings with company mentor	Scheduling more meetings
3 Technical challenges with Azure	Low	High	Thorough research and testing	Switch to an alternative cloud provider
4 Microservices integration issues	Low	High	Regular integration testing; clear API documentation	Temporarily consolidate services if needed
5 Security vulnerabilities	Low	High	Following best security practices and guidelines	
6 Budget overruns in cloud usage	Medium	High	Monitor cloud resources closely; use Azure cost management tools	Limit non-essential cloud services temporarily

7 Performance bottlenecks	Low	High	Stress testing and monitoring during development	Optimize performance; scale cloud resources

References

ICT Research Methods (2024). Retrieved from:

https://ictresearchmethods.nl/