University for Applied Sciences

Informatics Department

Applied Informatics

No More Waste

Documentation for the Architecture of an Mobile Application for Preventing Food Waste

Bruno Macedo da Silva
676839
inf3645@hs-worms.de

Supervisor	Prof. Dr. Volker Schwarzer
Working Period:	Summer Semester 2022
Due Date:	31.Juni 2022

Contents

1 Introduction and Goals

According to the Food and Agriculture Organization of the United Nations (FAO) in 2019, 931 millions tonne of food were wasted [FAO (2013)]. This has environmental, but especially social consequences. In a world where approximately 9.9% of the [AAH (2022)] population suffers from hunger that waste percentage sounds paradoxical.

As reported by the United Nations (UN) 5% of the global food loss and waste comes from restaurants [UN (2022)]. The solution for this problem must be locally applied so its effects can be seen in a global structure. To do so we propose to develop a mobile application that connects restaurants, bakeries and or pastries to clients. The former would offer their remaining products, which are still consumable, prior to the closing time, to a small price and the latter would browser in the app to find which shops are offering products.

We as "Clean Up the World ®" are a rising StartUp whose main concerns is to find environmental solutions to daily problems. Our portfolio includes projects about management of waste and optimization of household water usage. This product we want to develop targets small communities, like small cities or regions within a big city, to reduce the amount of wasted consumable food.

With our project we want to achieve the following goals:

- Connect providers with clients, so the former can offer products that the latter can purchase
- Collect statistical data about waste reduction within the providers
- Promote reduction of food waste that still could be consumed
- Allow clients to have a different dining experience.
- Allow providers to promote their products and gather new clients.

To make the easy to read we will use the pronouns "'he"' and "his" every time we refer to a single person.

After having an general idea on what the App should do, we create the following sketch to promote a simple visual example of our product when in use. This picture was shown to our intended users and to the boarding committee so they could have a idea of what to expect.

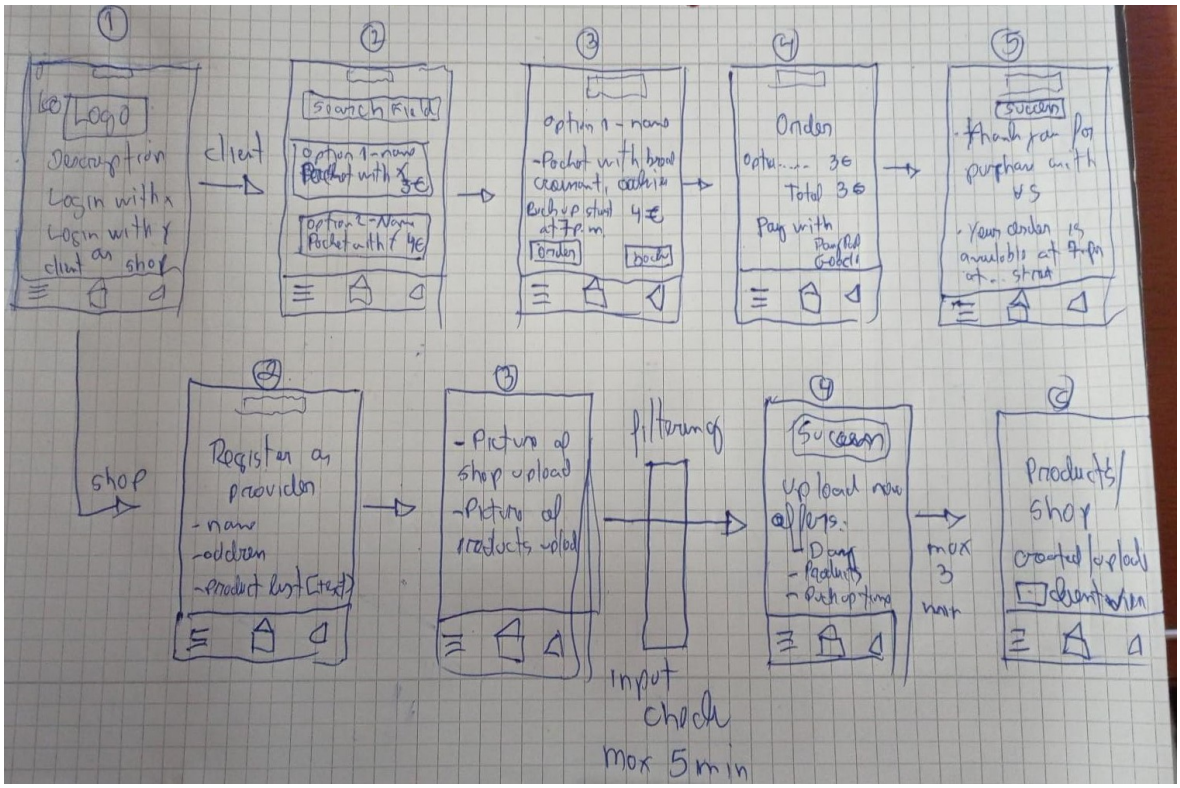


Figure 1: Sketch of the first idea

1.1 Design Purpose

The main purpose of this architecture is creating an exploratory prototype of an App. We aim to test it with potential stakeholders and regions to analyze their general acceptance and wishes [Cervantes and Kazman (2016)] and get a fast feedback.

This prototype will also make it feasible to identify unknown needs an wishes of the potential stakeholders, so we can eventually increase the scope of functionality. Exploring this domain will also provide us with information regarding the behavior of our target group when it comes to buying and serving food that would be wasted, but is still consumable.

1.2 Functional Requirements

The following functionalities describe the basic requirement for the App:

Id	Requirement	Description
F-1	Register as Client.	A Client can register to the app with its e-mail.
F-2	Login	After registration Client can login into the app.
F-3	Purchase option	A registered Client can purchase an available offer (see F7).
F-4	Filter/search options	A Client can perform filter and search actions for products.
F-5	Register as Provider	A Provider can register his store and add logos and pictures.
F-6	Create offer	A registered Provider can publishes what products they are offering with price and amount.
F-7	Upload offer	A registered Provider can add, edit or remove offers to his catalog.
F-8	Check orders	A registered Provider can check all existing orders targeting his/her shops.

ID	Motivation
F-1	The entry door of the App, where our Client get an overview of all available offers
F-2	In order to place purchases our client need to be registered. It will also provide statistical information about consumer behavior
F-3	Since we are dealing with a business relationship we have on one side a client willing to pay and for a product and on the other side a provider willing to offer a product/service
F-4	Like any other online-shop it is important that our Client can browse through the available possibilities
F-5	In order to make a product available a Provider needs to register his/her shop. This information will also be used for statistical analyzes about providers, products and consumer behavior
F-6 - F-7	A registered Provider can make an offer available according to his/her daily planning. For future development of this app, this will be helpful to identify tendencies regarding dates, periods and availabilities.
F-8	Also registered providers can get an overview about how often their products have been sold. This may open a different kind of business orientation.

The following Use Case Diagram should give our stakeholders an overview of the primary functionality of the app:

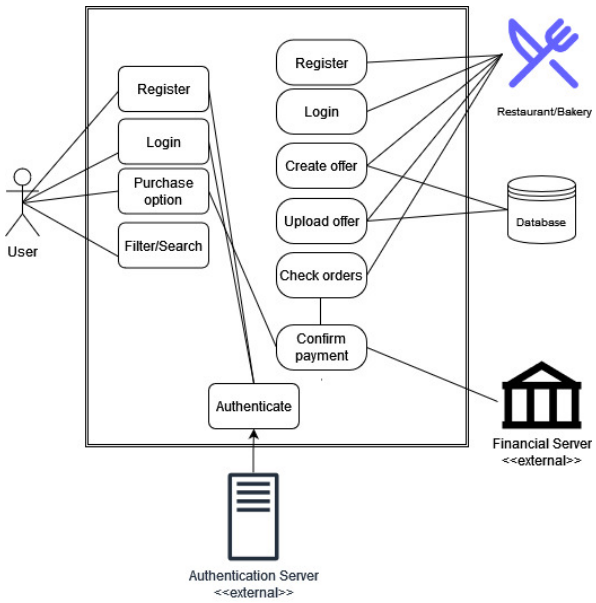


Figure 2: Preliminary functions

1.3 Quality Goals

The key qualities of this app are described in the table below:

Quality	Priority	Motivation
Usability	1	Since we are working with a prototype it is important the usage is easy as possible, to attract more users and to gather information about consumer behavior. clients and providers should have a simple interface where they can quickly interact without any burdens.
Interoperability	2	To reduce programming burdens and accelerate the delivery of a working product the registration and payment process will rely on third party providers. For that reason the developed features should work faultless in combination with the external Application Programming Interface (API) (i.g Mobile Payment Gateway and Federated Login).
Security	3	To guarantee a secure and easy payment process we will handle the API of the Mobile Payment Gateway within the development process. The possibility of outsourcing this service would cause a big damage to the first priority.

1.4 Stakeholders

The main stakeholders of this app are described in the table below:

Stakeholder	Description	Motivation
Providers	Owner of a restaurant, bakery or pastry.	One of the protagonist of this app. They will interact with clients using the app. From his usage we will gather valuable information about consumer behavior.
Clients	Person who wants to purchase last minute product from a provider.	The second protagonist of the app they will interact with the provider to search and to purchase product. The result of this interaction will provide us with statistical information to understand how food waste can be reduced.
Developers	Team in charge of creating the application using existing tactics and creating new solutions.	Responsible for guarantee that the main requirements of the app are fulfilled and fully functional. Since they will be dealing with the background of the product, it is important that they understand it very good so it can also be implemented in a final version.
Boarding Committee of "Clean Up the Word (R)"	Members of the management team who wants to delivery environmental solution do daily problems and at the same time develop a profitable product.	Group in charge of main decisions regarding what will be developed. Their decision are based on mark tendencies and on environmental issues.

2 Constraints

In this project we must distinguish between Technical (CT-T-#) and Business (CT-B-#) Constraints. The former describes specific elements of the project, like programming language, released platform (e.g. operational systems) and technical decisions related to the functionalities. The latter deals with management elements [Franzen and Thoms (2020)] (e.g time, budget and team). The following tables describes the business and the technical constraints of this project:

Business		
Id	Constraint	Reasoning
CT-B-1	Time to first prototype release - Maximal 1 year	To stay on the budget the first release should be ready within one year after the approval
CT-B-2	Development Team - 4 Dev + 1 Sys-Admin	Most of the development team members is allocated on other projects and cannot be changed. They should be in charge of the development and maintainability.
CT-B-3	Analytical Team - 1 person	During running phase of the prototype it will be necessary to have someone in charge of evaluating and interpreting the collected data, to find out if the goals are being achieved.
CT-B-4	Budget	For this kind of project the maximum budget is US$ 150.000. It should cover the development of the main functionality and the data analysis (see CT-B-3). "'middle app" according to [SPD LOAD (2019)]
CT-B-5	Trust of third party resources	Since login and payment will be processed by third party providers, Federated Login and API Gateway, a lack of security on their side can cause damage to the reputation of our company.

Technical		
Id	Constraint	Reasoning
CT-T-1	Programming Language - Kotlin	A multilanguage (Java, Kotlin, iOS, Swift) approach increases the maintainability burden and consequently the costs (see CT-B-4). It can also interfere with compatibility with different kind of device.s
CT-T-2	Platform - Android	Offering the application for different platforms (iOS and/or Android) increases costs for maintainability and requires a bigger team. Since the prototype should run during the first year mainly to gather information about consumer behavior the costs in this test phase can increase rapidly if we decide to develop for the most common platforms.
CT-T-3	Payment	One the one hand creating an own payment framework can gives full control of the application, but on the other hand it will required specialized team and increases costs and time (see CT-B-4).
CT-T-4	Payment gateway	Using existing Mobile Payment Gateway reduces development time, but demands fully Interoperability of the app with the existing gateways. It may also be a problem if the Client don't use this kind of payment method.
CT-T-5	Login	Using existing Federated Login decreases development time, but like CT-T-4 demands fully interoperability of the app with appliances. It may also be a problem if the Client don't trust this kind of login.
CT-T-6	Amount of servers	The existing budget allow us to keep only one full operating server to handle the circulating data.
CT-T-7	Lack of deep security measures	Due to low budget and limited time security within the app is handled mostly by the third party operators. Within the app we have only filtering measures to process inputs.

3 Context and Scope

Since this system relies on the correct working of external elements, it is important that their interaction is corrected displayed.

3.1 Business Context

This graphic is addressed to the following stakeholders: providers, clients and Boarding Committee of "Clean Up the Word (R)". It should give a black-box view of the interaction the app and its external components.

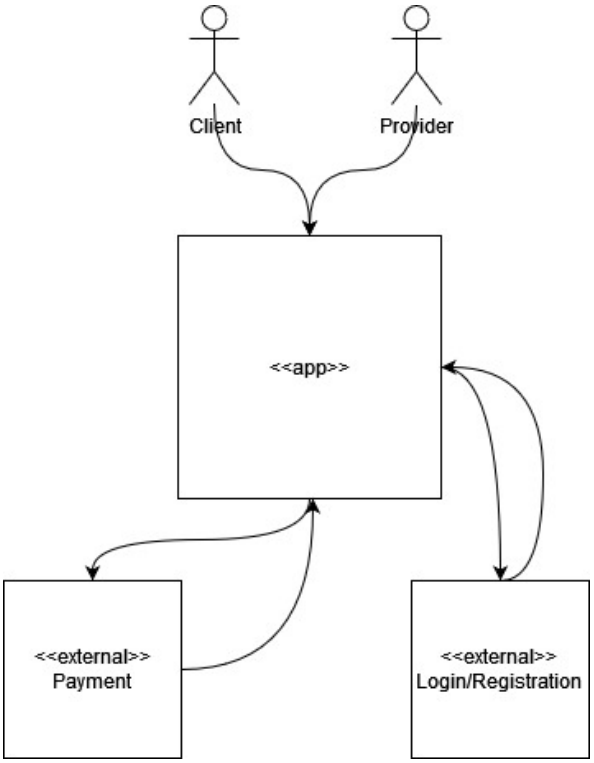


Figure 3: Diagram to describe the business context

Artefact	Description
Client	Searches for a last time offer from a restaurant, bakery or pastry.
Provider	Offers a still consumable product that was not sold during normal working time.
Payment	Deals with the payment processing using registered information from another payment platforms.
Login/Registration	Authenticated users using logins from other platforms.

3.2 Technical Context

This following graphic is addressed to our technical team. It provides a white-box view of the previous graphic, 3.

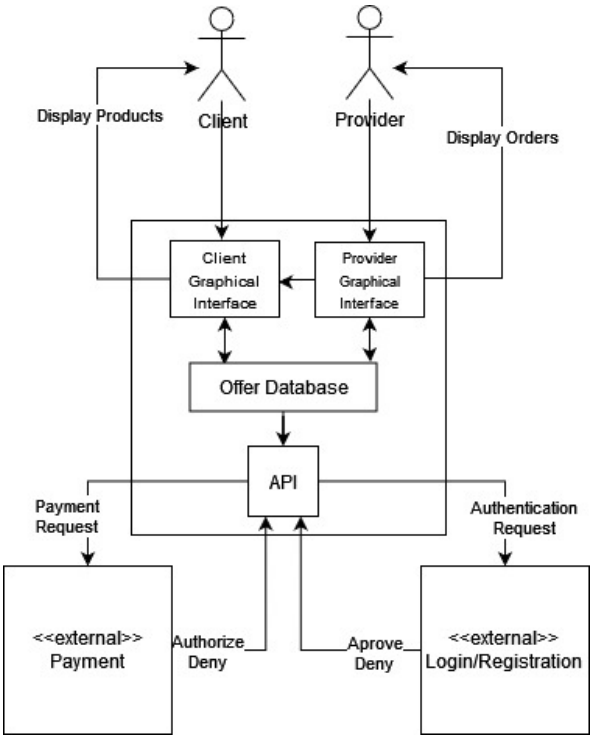


Figure 4: Technical Context

Artefact	Description
Graphical Interface	Client and Provider have an own interface to interact. Provider can access view their offer also with a Client's perspective.
Offer Database	Clients and providers can make requests to the database to inquire about its content.
API	For login and payment the authentication and authorization take places on the external service.

4 Building Block View

In this section we will describe the App using some elements of the 4+1 Architectural View Model. With this model we aim to target an understanding of all our main stakeholders.

We will use 4 different views, which should focus on specific elements of the project. Each view provides a different purpose Kruchten (1995). For this project we will provide the 3 following views of the 4+1 Architectural View Model:

- **Scenario view**: simple description for the end user
- **Structural view**: object-oriented decomposition
- **Behavior view**: description of the existing processes

4.1 Scenario view

Our first picture 2 provided our stakeholders a brief presentation of the basic functionalities of our app. Other elements addressed in this view were presented in the chapter where we discussed the business context, 3.1.

An example on how each feature of the app should work can be found in our use case in section 6.2. More elements of this view will be presented while discussing the internal decisions in chapter 5.

4.2 Structural view

For the following graphic we choose a Class Diagram to provide our development team a further view of the structural elements of the project. This first class diagram gives a simple description of the classes that can exist in the app.

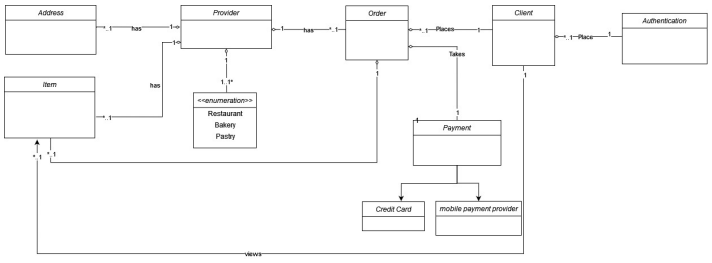


Figure 5: Level 1 - Class Diagramm

Zooming down the classes we presented before, we can see how they are build with its attributes, methods and interaction with other elements.

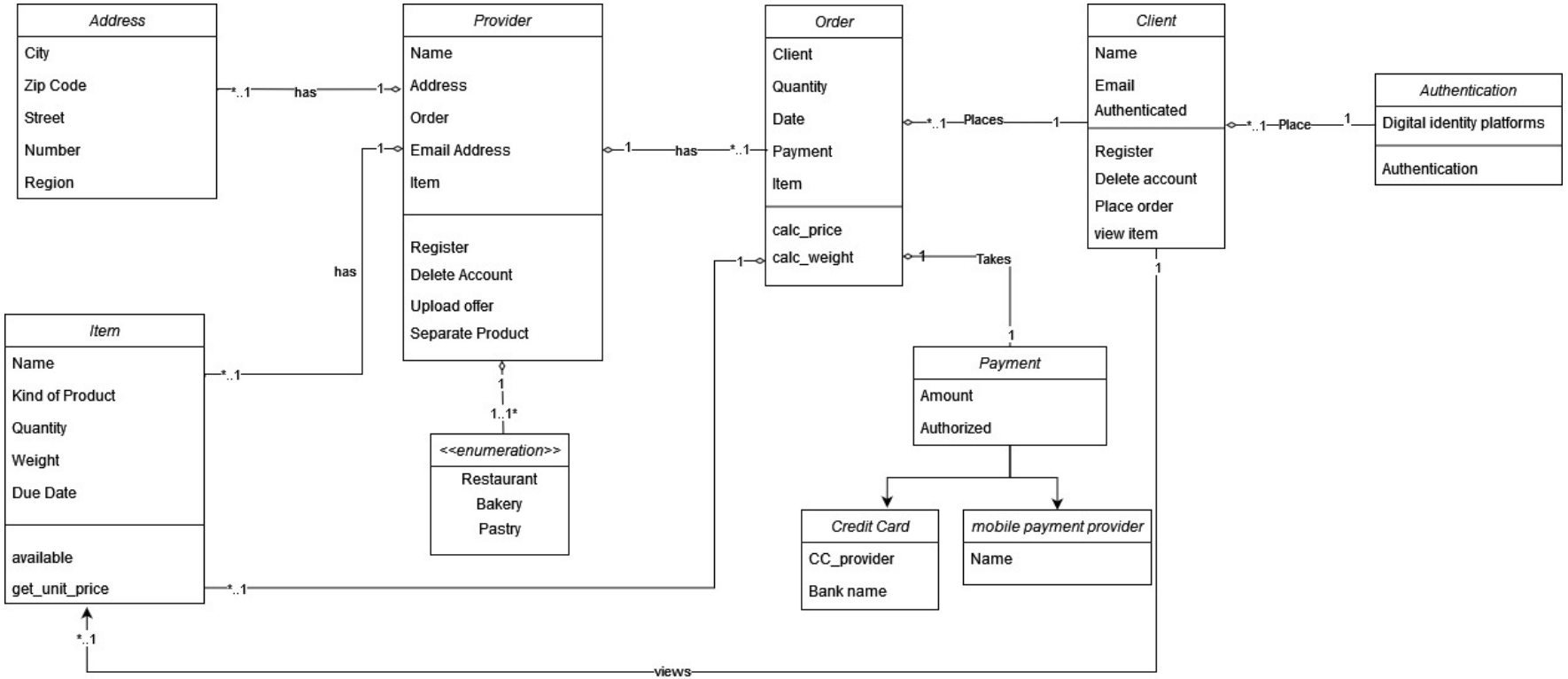


Figure 6: Classes Overview

4.3 Behavior view

The following Activity Diagram depicts the register and login procedure within the app. It should explain our main stakeholders, providers and clients, the starting process of the app.

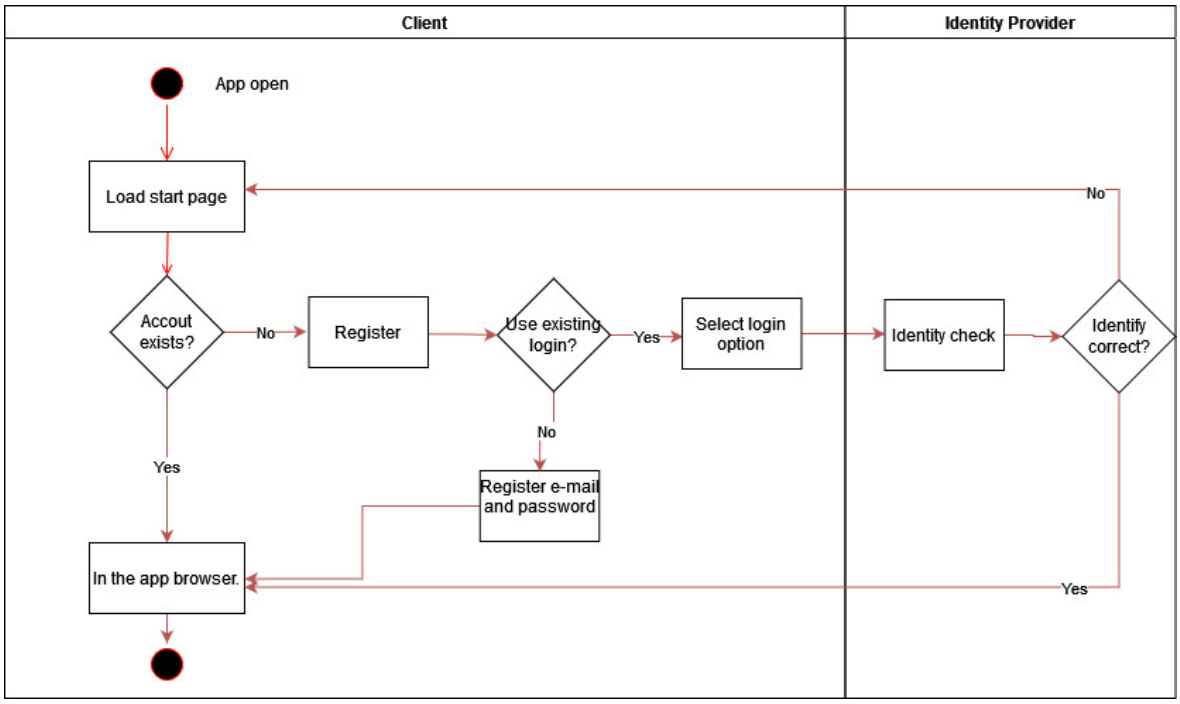


Figure 7: Login procedures

To help understanding the above process we created the following Sequence Diagram that depict the communication flow with the third party providers.

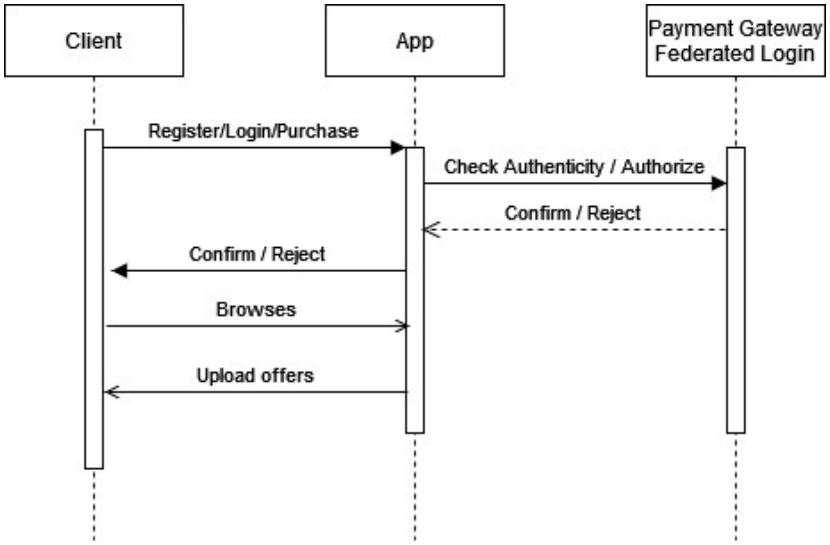


Figure 8: Sequence of actions with third party applications

5 Crosscutting Concept

In this chapter we will present the technical solutions that we will use to develop this project. For each quality attribute we will present the chosen tactics.

5.1 Solution for Usability

The core of our app is how easy it is to use. We want our user to navigate through it without being overwhelmed with information not related to the main objective: purchase a product or upload a product.

To guarantee that our clients could get an easy and fast update from the providers we choose the *observer* pattern over the wellknown Model-View-Controller (MVC). The former allow us to have focus on the implementation on the different sides of our app, provider and clients. The latter has an inherent development burden for simple user interfaces. Using Model-View-Controller (MVC) for this first prototype would increase the difficult level and the time needed to provide a functional. Below there is a simple depict of this pattern:

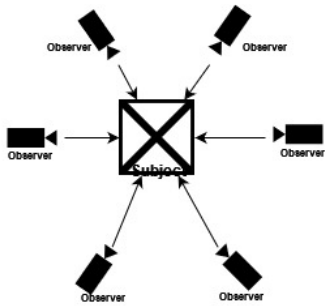


Figure 9: Observer Pattern simple explained

The next diagram shows a more detailed version of the of the *observer* pattern. This view intends to assist the development team with the implementation of this pattern.

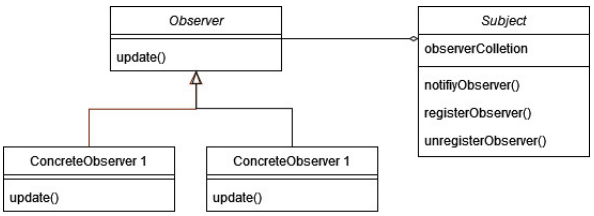


Figure 10: Observer Pattern with Classes

Tactict	Pattern	Motivation	QA
Support User Initiative	Observer	The interaction of the users is a main factor of our app. We want them to have fully control of their actions either by cancelling or by resuming an action. The updates by the provider should be directly sent to the clients in a simple fast fashion.	QA-1
	Lazy Registration	Avoid having to memorize another password and username may increase the acceptance of the user. With this pattern we allow them also to browse in the app and seeing what is available without being registered Interaction Design (2017). This may give a glimpse of what they get if they join us. The alternative is to have the user first register his credentials and then browser in the app. This burden would only make the get-to-know process slow and difficult.	
Support System Initiative	Observer	By each upload from the providers we want our clients to have it on his device, without having to "ask" for it.	

5.2 Solution for Interoperability

The communication with the third party components should during the whole lifetime of the App reliable. Since we are dealing with two different services, Mobile Payment Gateway and Federated Login, we will describe the integration processes according to each specification.

5.2.1 Payment Gateway

The usage of Mobile Payment Gateway offers three possibilities Zoho (2019):

- Redirection to payment processor's page
- Payment data and processing inside the application

- Payment data entered in the app, but processed with an API

The third option stays in direct contact with our top quality attribute, usability. Since we want to offer a easy shopping experience, the payment process should also be harmonic with other features.

Our main tactic here is to *limit dependency* using APIs. Since we want to delivery a functional prototype within short period of time, we want to concentrate our efforts on the elements of the usability. For that reason using APIs avoids also "reinventing the wheel" for situations where there ara consolidated solutions.

Tactict	Pattern	Motivation	QA
Limit Dependencies with an Intermediary	Wrapper	The API will be the intermediary for the payment process. For the clients all visible steps will occur in the app, without being sent to another page. On the background the API will receive the input and send it to the payment gateway. The verification takes place in gateway, which then communicate with the financial institute of the client and send the payment to the Provider Zoho (2019).	QA-2

5.2.2 Federated Authentication

Using of Federated Login reduces burden of saving user credentials locally. It also improves the usability so users do not have to create and remember another username and password. The authentication process takes place on the third party operator, as seen in the picture 8.

Tactict	Pattern	Motivation	
Microservice	API Gateway	Increase of security so the microservice is not directly exposed to the external world. It reduces the complexity of the microservice, since the gateway will have to deal with data transfer rate, tokens and other activities. Dealing with failures would also be handled and logged by the microservice javarevisted (2021).	QA-2

Following we present a graphic that depicts the login process using an API Gateway. This

graphic should assist our primary stakeholders to understand that with their existing credentials from other services they can access multiple services, including ours.

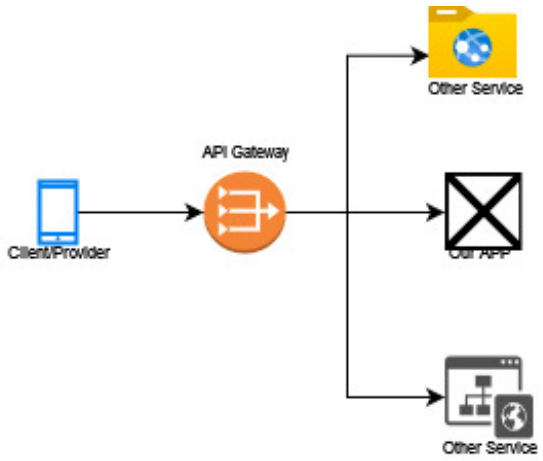


Figure 11: Federation Identity

When we white-box the previous graphic we get a deeper view on how the API Gateway should interact with the Client. This graphic should assist the development in the integration process of the app with the external element..

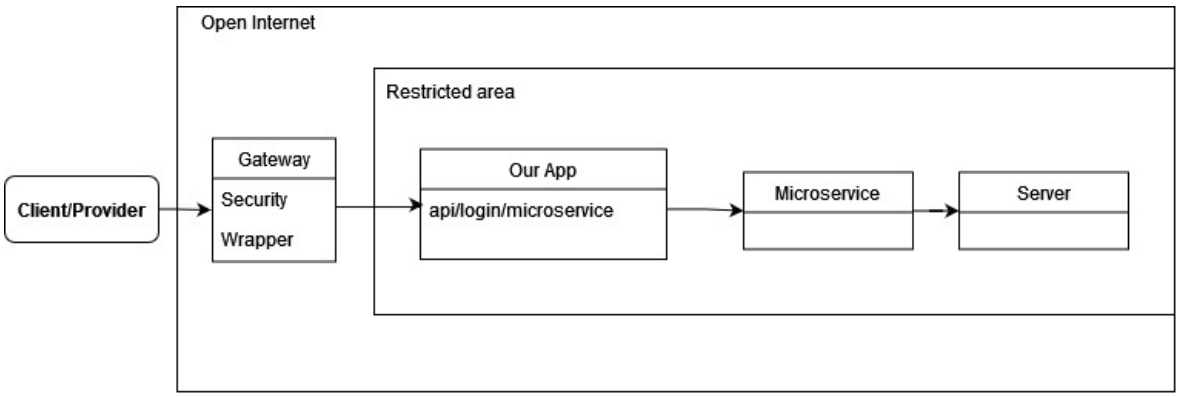


Figure 12: Environment Depict of Federation Identity

5.3 Solution for Security

There are two security concerns that need to be addressed to the users. The first one deals with the authentication and payment process. This will be managed by the third party providers. The second one involves the interaction of the providers with the app. Since this stakeholder can upload data and file to the app it is important that only approved data type is inserted. In the table below we will describe the tactics used for the these two concerns.

Below there is a graphic that explains in an easy fashion the flow process of the data input by the providers:

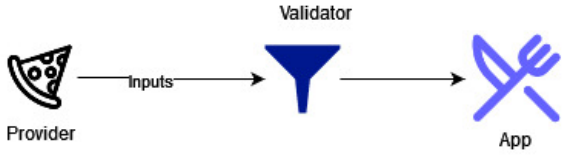


Figure 13: Input Validator

Tactic	Pattern	Motivation	QA
Validate Input	**Intercepting Validator**	providers has a big interaction with the app. They can upload files and texts. To make sure that only secure element a inserted into the app, it is important that every input is analyzed before reaching the app and the clients Steel et al. (2012).	QA-4
Authenticate Actors **Authorize Actors**	Authentication enforcer Authorization enforcer	To avoid the connection of bots we want to allow only registered users to interact with the functionalities of the app. This will be done with the third party operators [Wikipedia (2020)].	QA-4

The choice for this validator was based on its cost and on its easy implementation. A more complex solution like an Intrusion Prevention System would increase the overall security aspect of app but it would also increase the compatibility burden, demands a more complex structure and increases the maintainability costs.

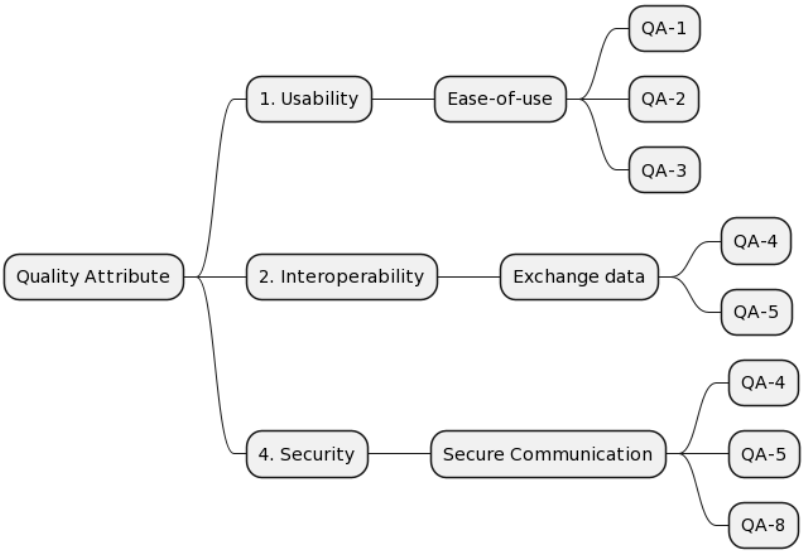
6 Quality Requirements

6.1 Quality Tree

The priority of each element will be expressed using the following notation:

- ([Customer view], [Architect view])
- H - High
- M - Medium
- L - Low

Users and the development team have different perspective of an app. The former think about how attractive and easy to use it is, the latter want to build something what achieves a goal. For that reason is the interpretation of the priority sometimes so different, according to which group has been asked.



The next table presents the reason for the previous categorization regarding the priority of each quality attribute.

ID	Reasoning for Users	Reasoning for Development Team
QA-1	All important information should be there so his shop can be well promoted	An initial registration with filter for the input is important, but aesthetically details are the goal now.
QA-2	Once he get something new, he wants to make it available	First we need to guarantee that no overrides occur than they see if it is promptly displayed.
QA-3	They want to browse and see all available options	Search engine can be very helpful, but filtering can wait a litte, since it does not affect the app itself
QA-4	The most important is that they can use and purchase	This integration must be done fast and careful so no mistakes shows up.
QA-5	They just want to easily and secure pay, it does not matter how it works	The compliance with payment regulations is a must, since any mistake can costs huge fines and damage to the image of the company.
QA-6	They don't want to wast time with loading pages	The loading time can be fixed once there is a structure that allows loading in the first place.
QA-7	They want to get confirmation that everything worked fine.	The communication between the API and the payment provider show comply with all existing regulations. Push notification can be added once the main feature works.
QA-8	That is something that they don't want to see, but want to make sure that it exists	Since the payment is processed by the third party operator, all concerns should be addressed to them and specified in the Service Level Agreement (SLA)

6.2 Evaluation Scenarios

From the requirements, 1.2, we could develop the following uses cases and depict the main quality attributes of this project.

Use Case	Description
UC-1: Register as Client	The Client registers an e-mail address.
UC-2: Login	The Client logins in to the system.
UC-3: Places an order	The Client chooses a Provider.
UC-4: Register payment	The Client registers a payment method.
UC-5: Register as Provider	The Provider registers their facility and products.
UC-6: Update availability	The Provider uploads their product catalog.

Table 1: Use Cases

With the following use cases we will be able to define the major quality attributes that are

involved in the development of this application. They should be measurable and testable so we can verify if the system meets the needs our stakeholders [Cervantes and Kazman (2016)].

ID	Quality Attribute	Scenario	Associated Use Case
QA-1	Usability	A Provider is able to register his company, to specify the kind of products he offers and upload a logo or picture of his shop and products in a easy and fast (within 5 Minutes) fashion.	UC-5
QA-2	Usability	A Provider is able to update the offers at any time.	UC-6
QA-3	Usability	A Client is able to search and filter options.	UC-6
QA-4	Interoperability	A Client can register his e-mail using another account (Google, Microsoft, Facebook) in a Federated Login	UC-1
QA-5	Interoperability	A Client can pay the order using a Mobile Payment Gateway (e.g. Stripe, Square, PayPay, SecurePay)	UC-4
QA-8	Security	The Client selects a payment method from an existing service. The client inputs no payment data in the app. Everything is processed between the API Gateway and the financial institute.	UC-4 & QA-5

The defined quality attributes are represented in the following scenarios:

Usability			
Scenario	**Value**		
ID	QS-1	QS-2	QS-3
Source	Provider	Registered Provider	Client
Stimulus	wants to register his shops	wants to make a last minute offer	wants to search/filter offers
Artifact	app	app	app
Environment	working time, during afternoon	peak period, between 4 and 7 pm on Friday	peak period, between 4 and 7 pm on Friday
Response	offer available in the app	immediate availability of the offer in the app	display of the filter/search output
Response Measure	How long did the registration and upload process take? How many and what kind of error messages did the Provider get?	How long did it take to upload an offer? How many and what kind of error messages did the Provider get?	What kind of inputs did the user has to place until he finds what he wants? Did he have to type anything or were filter/search options available? How long it takes until the client finds a product?

Interoperability		
Scenario	**Value**	
ID	QS-4	QS-5
Source	Client	Client
Stimulus	wants register using a Federated Login	wants to pay using existing mobile payment account
Artifact	app and Federated Login provider	app and Mobile Payment Gateway
Environment	peak period (on the context of the Federated Login provider)	peak period (on the context of the gateway)
Response	authentication succeed or failed	confirmation / declined
Response Measure	How much data was transmitted and how much was queued? Focus on System overload [Kasunic and Anderson (2004)]	Total amount generated data in the app that are transferred and processed and rejected by the gateway? Focus o connectivity and system overload [Kasunic and Anderson (2004)]

Security		
Scenario	**Value**	
ID	QS-6	QS-7
Source	Client	Client
Stimulus	clicks on registration using an existing login	click on pay using an existing mobile payment account
Artifact	app, API Gateway and Federated Login provider	app, Microservice and Mobile Payment Gateway
Environment	peak period (on the context of the Federated Login provider)	peak period (on the context of the gateway)
Response	authentication succeed or failed	confirmation / declined
Response Measure	Required time and effort to intercept and/or block requests (create Denial of Service (DoS))	Extension and impact to image/damage of the app/company in case of attack

7 Risk and Technical Debt

To measure the risks of this project we will use the following 3x3 risk matrix, which will help us develop the Risk Assessment:

Figure 14: 3x3 Risk Matrix Template
Source: [Smartsheet (2017)]

On the left side we see the risk table defined after several discussion with the team members. On the right side there are the elements to which the table refers to:

Risk Criteria	Element ID					
	1	2	3	4	5	6
Unproven technology	1	1	1	1	1	1
Performance	2	2	1	1	1	1
Scalability	1	1	2	1	1	1
Availability	1	1	4	2	1	1
Data loss	1	1	3	1	1	1
Single points of failure	1	1	4	4	1	1
Security	1	2	3	2	2	2

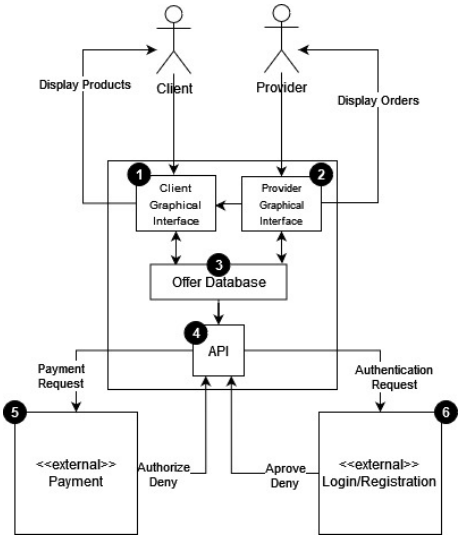


Figure 15: Modified from Figure 4

7.1 Motivation

Risk Criteria	1 Client Graphical Interface	2 Provider Graphical Interface	3 Offer Database
Unproven technology	*not applicable*	*not applicable*	*not applicable*
Performance	Concerns regarding a such dynamic shop. Data displayed	Concerns regarding a such dynamic shop. Data update.	*not applicable*
Scalability	*not applicable*	*not applicable*	One database can be overwhelmed if the number of user is not limited for this prototype version.
Availability	*not applicable*	*not applicable*	In case of intense traffic the latency can be more than expected.
Data loss	*not applicable*	*not applicable*	Nowadays no company can survive in case of data loss. Technical damages can be mostly easy fixed, but moral damage stays forever.
Single points of failure	*not applicable*	*not applicable*	Only one component gives always this risk. Increasing the number of components increases also the total development costs
Security	*not applicable* the client's input is always processed by the APIs and by the third party providers	If no strong data filtering exists, providers can upload malicious files or execute unwished commands.	The filtering of the input should occur mostly on the server side, Source no external access can figure out, how it is implemented.

Risk Criteria	4 API	5 External Payment Service	6 External Authentication Service
Unproven technology	*not applicable*	*not applicable*	*not applicable*
Performance	*not applicable*	*not applicable*	*not applicable*
Scalability	*not applicable*	*not applicable*	*not applicable*
Availability	The access to the products become unstable.	*not applicable*	*not applicable*
Data loss	*not applicable*	*not applicable*	*not applicable*
Single points of failure	In case of failure login, registration and payment are compromised.	*not applicable*	*not applicable*
Security	Lack of practical experience within the team about this topic. we rely on the service provided by the third party companies	SLA Less than 99.5% but equal to or greater than 95.0% [Paycore (2019)]	SLA < 99.99% - >= 99.9% auth0 (2014)

References

AAH (2022). World hunger: Key facts and statistics 2022. *actionagainsthunger.org*. https://www.actionagainsthunger.org/world-hunger-facts-statistics, Accessed 18th May 2022.

Amplify Docs (2020). Federated identities. https://docs.amplify.aws/sdk/auth/federated-identities/q/platform/android/. Accessed 9 June 2022.

AppDynamics (2020). 16 metrics to ensure mobile app success. https://www.hwe.design/theories-concepts/system-response-stability. Accessed 8th June 2022.

auth0 (2014). Service levels. https://www.smartsheet.com/all-risk-assessment-matrix-templates-you-need. Accessed 10th June 2022.

AWS (2017). Microservices. https://aws.amazon.com/microservices/?nc1=h_ls. Accessed 9th June 2022.

Baresi, L. (2009). *Activity Diagrams*, pages 41–45. Springer US, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_9, Accessed 26th May 2022.

Cervantes, H. and Kazman, R. (2016). *Designing Software Architectures: A Practical Approach*. Pearson Education, Boston.

FAO (2013). Food wastage: Key facts and figures. *fao.org*. https://www.fao.org/news/story/en/item/196402/icode/, Accessed 18th May 2022.

FAO (2022). 17 *fao.org*. https://www.fao.org/food-loss-reduction/news/detail/en/c/1378973/, Accessed 18th May 2022.

Franzen, E. and Thoms, M. S. (2020). Architectural drivers in modern software architecture. https://medium.com/@janerikfra/architectural-drivers-in-modern-software-architecture-cb7a42527bf2, Accessed 18th May 2022.

HWE.DESIGN (2020). Hardware engineering design. https://www.hwe.design/theories-concepts/system-response-stability. Accessed 8th June 2022.

IBM (2004). What is Class Diagram? https://developer.ibm.com/articles/the-class-diagram/, Accessed 18th May 2022.

Interaction Design (2017). User interface (ui) design patterns. https://www.interaction-design.org/literature/topics/ui-design-patterns. Accessed 9th June 2022.

javarevisted (2021). Top 10 microservices design patterns and principles - examples. https://javarevisited.blogspot.com/2021/09/microservices-design-patterns-principles.html. Accessed 9th June 2022.

Kaspersky (2021). What are bots? – definition and explanation? https://www.kaspersky.com/resource-center/definitions/what-are-bots. Accessed 9th June 2022.

Kasunic, M. and Anderson, W. (2004). Measuring systems interoperability: Challenges and opportunities. Technical Report CMU/SEI-2004-TN-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6871. Accessed 1st June 2022.

Kruchten, P. (1995). The 4+1 View Model of architecture. *IEEE Software*, 12(6):42–50. https://doi.org/10.1109/52.469759, Accessed 18th May 2022.

MuleSoft (2015). What is an api? (application programming interface). https://www.mulesoft.com/resources/api/what-is-an-api. Accessed 8th June 2022.

nginx (2021). What is load balancing? https://www.nginx.com/resources/glossary/loa

d-balancing/. Accessed 9th June 2022.

Overby, S., Greiner, L., and Paul, L. G. (2017). What is an sla? best practices for service-level agreement. https://www.cio.com/article/274740/outsourcing-sla-definitions-and-solutions.html. Accessed 11th June 2022.

Paycore (2019). Service levels. https://paycore.io/slas/payment-and-payout-gateway-sla/. Accessed 10th June 2022.

PayPal (2016). Paypal payments pro. https://www.paypal.com/us/webapps/mpp/paypal-payments-pro. Accessed 9th June 2022.

Richardson, C. (2020). Pattern: Api gateway / backends for frontends. https://microservices.io/about.html. Accessed 9th June 2022.

Robinson, M. (2019). What does federated login mean? a simple, detailed answer. https://carvesystems.com/news/what-does-federated-login-mean-a-simple-detailed-answer/. Accessed 1st June 2022.

Schwarzer, V. (2022). Lecture 11 – risk storming and how to be a good architect [powerpoint slides]. Software Architecture 506 SWA, Worms University of Applied Sciences.

Smartsheet (2017). Download free, customizable risk matrix templates. https://www.smartsheet.com/all-risk-assessment-matrix-templates-you-need. Accessed 10th June 2022.

Sparxsystems (2004). Uml 2 tutorial - sequence diagram. https://sparxsystems.com/resources/tutorials/uml2/sequence-diagram.html. Accessed 15th June 2022.

SPD LOAD (2019). How much does it cost to develop an app in 2022? cost breakdown. https://carvesystems.com/news/what-does-federated-login-mean-a-simple-detailed-answer/. Accessed 1st June 2022.

Steel, C., Nagappan, R., and Lai, R. (2012). *Core Security Patterns: Best Practices and Strategies For J2EE, Web Services, and Identity Management*. Pearson PTR, Boston.

techopedia (2018). Wrapper. https://www.techopedia.com/definition/4389/wrapper-software-engineering. Accessed 9th June 2022.

UN (2022). Stop food loss and waste, for the people, for the panet. *un.org*. https://www.un.org/en/observances/end-food-waste-day, Accessed 18th May 2022.

Vilmate (2019). Payment gateway integration in a mobile application. https://vilmate.com/blog/payment-gateway-integration-in-a-mobile-application/, Accessed 31st may 2022.

Waykar, Y. (2015). role of use case diagram in software development. *International Journal of Management and Economics*. https://www.researchgate.net/publication/322991847_role_of_use_case_diagram_in_software_development. Accessed 1st June 2022.

Wikipedia (2020). Security pattern. https://en.wikipedia.org/wiki/Security_pattern. Accessed 9th June 2022.

Wikipedial (2022). Model–view–controller. https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/. Accessed 15th June 2022.

Zoho (2019). Online payment gateway – definition, working & benefits. https://www.zoho.com/books/guides/payment-gateways.html. Accessed 9th June 2022.

Glossary

Application Programming Interface (API) Software intermediary that promotes the communication between different systems/applications/ softwares [MuleSoft (2015)].

Denial of Service (DoS) Intentional interruption or laming of network services.

Model-View-Controller (MVC) Pattern used in software architecture for the development of user interfaces. In this pattern information is presented to the user differently from how it is show internally Wikipedial (2022); **?** .

Service Level Agreement (SLA) Contract between a service provider and a consumer regarding the expected level of service that is accepted. This contract is based on defined metrics, measures to remediate occurring problems and penalties if the contracted level is not achieved [Overby et al. (2017)] .

Activity Diagram This kind of diagram shows the behavior of a system, it depicts in a graphical fashion the logic of a single use case [Baresi (2009)].

API Gateway Server used an single entry point into a system. It forwards requests to the used service. It is broadly used for authentication, auditing and logging services [Richardson (2020)].

App It refers to the mobile application to be developed.

bots Short for robot. It is a software that performs automated and repetitive tasks. It is usually controlled by a malicious actor who targets a network or service. They can be used consume resources and make a service unavailable, steal credentials and other attacks [Kaspersky (2021)].

Class Diagram This kind of diagram presents the structure of a system with its classes, attributes, methods and relationships [IBM (2004)].

Client Since we have two major stakeholders that will use the app, the word client will specify the one that places an order in the app.

Federated Login Authentication method in which users use existing accounts to gain access

to another domains or systems without the need of creating new credentials. The authenticity of a user is attested by service and granted to another [Robinson (2019)].

Intercepting Validator Mechanism used to check if user's input (data or file) corresponds to the criteria defined in the app. Non-conform input are discarded before even reaching the app [Kaspersky (2021)].

Microservice Software architecture approach made of small independent services used to communicate with other resources like APIs. The advantage of using this architecture is its scalability and maintainability, since each service is responsible for a very small group of correlated tasks [AWS (2017)].

Mobile Payment Gateway Those services works as an intermediary between customer, merchant and bank/credit card company. Here a payment request is sent to the gateway and forwarded to the approval instances. The core functionality of those gateways is the cryptography within the communication steps [Vilmate (2019)].

Provider The second major stakeholders are those who offer their products. They can be restaurants, bakeries, pastries and similar.

Risk Assessment Report used to identify risk/weakness that may exist in a project. [Schwarzer (2022)].

Sequence Diagram This kind of diagram presents the interaction of the elements over time as the communication flows from one direction to the other. It also displays which objects communicate with each other and the request that starts the process [Sparxsystems (2004)].

Stakeholder Describes all kind of potential person or entity that may have interest using the app.

Use Case Diagram This kind of diagram presents the main requirements and functionalitiesPa of a systems. It displays a simplified overview of core purpose of the application [Waykar (2015)].

User See stakeholder.

Wrapper Element used to encapsulate the complexity of one entity so it can be processed by another entity [techopedia (2018)].

Abbreviations

API Application Programming Interface.

DoS Denial of Service.

FAO Food and Agriculture Organization of the United Nations.

MVC Model-View-Controller.

SLA Service Level Agreement.

UN United Nations.