INFORMATION SYSTEM

for Covid-19 Test Center and Infection Detection

Chivu Vlad

KEA, Computer Science 2021 – 2nd semester Exam Project

Contents

Introduction

Motto: "Plans Are Worthless, But Planning Is Everything." - D. D. Eisenhower

For this project, we have chosen to follow the Rational Unified Process (RUP) as a development methodology framework which means that we are going to analyze, design, and develop our information system concerning the phases and techniques used in the Rational Unified Process.

RUP is use-case-centric and emphasizes starting with the most important functionalities that are most valuable for the stakeholders of the business of which we are considering developing a new or better solution.

Because RUP combines analysis and design at the beginning of a project yet still follows an iterative approach, it represents a sensible compromise between a purely agile approach and the waterfall approach.

Phase	Chapter	Step	Technique	Deliverable
Planning Focus: Why build this system? How to structure the project? Primary outputs: — System Request with feasibility study — Project plan	1 1 2 2 2	Identify opportunity Analyze feasibility Develop workplan Staff project Control and direct project	Project identification Technical feasibility Economic feasibility Organizational feasibility Time estimation Task identification Work breakdown structure PERT chart Gantt chart Scope management Project staffing Project charter CASE repository Standards Documentation Timeboxing Risk management	System request Feasibility study Project plan — work plan — Staffing plan — Standards list — Risk assessment

*Systems Analysis and Design 5th Edition, Author: Alan Dennis

I Inception Phase

1.1 Problem Statement

Why build this system?

Given the importance of public health and its ramifications on other industries, it is of great significance to have a system in place that will provide the latest information from the health department, and guidance to people on how to respond to this pandemic situation.

One of the latest requirements from the Danish Health Authority when it comes to indoor activities states that all persons must present a negative Covid Test performed within the last 72 hours.

The problem that arises here is how to manage testing centers without putting people in danger and offer enough alternatives so one location does not take more appointments than its full capacity. We must take into consideration that since summer is approaching more people will want to go out so testing centers will probably be very busy. To support this, demand a swiftly appointing system must be put in place that will be able to scale up during busy hours. Personal data is another important factor since we want to keep track of infection rates so we must design the system with the best practices of security in mind.

1.2 Problem Analysis

Who will benefit from this new system?

To better understand the ins and outs of the system considered to be developed we must put to test the benefits of this system for its users and other stakeholders. A Stakeholder Analysis will clarify this and bring more light on the possible system features before starting to think at the user level.

Does a satisfying solution exist already?

An investigation of an already existing solution to our problem is to be conducted so our efforts won't be in vain.

We have tried the current system (coronaprover.dk) and looking at the three measurements that we found as most important for this system, Reliability, Scalability, and Security we can say that it provides a sense of security by providing the NemID login, it is easy to navigate, important information (e.g., how to get tested) is highlighted, still, it can be improved in terms of time waiting to make an appointment to get tested.

What can be improved to the existing solution?

Aside from improving scalability, we could try to integrate the services of making an appointment to get tested, getting the result, and get vaccinated in one system composed of these three microservices.

Data coming back from all these services can be used with the consent of its owner to inform other people that they have been in contact with a positively tested person and suggest them to get tested. We should be offering an option for anonymity since many persons don't want to publicly share this. We could also gather the latest locations of a positively tested person at his indications, and we could inform not just the persons he knows he has been in contact with but many who transited the place or area. That information could fill up an interactive map that shows persons what areas should avoid.

How is the data handled and is it safeguarded?

We are working on building a secure application in terms of communication between client and server and more information of where the data is held or how is it used will be posted on the website and it is a matter of policy at the Health Department level.

1.2.1 Domain Analysis (SWOT & Stakeholder Analysis)

Analysis Focus: Who, what, where, and when for this system? Primary output — System proposal	3	Develop analysis strategy	Business process automation Business process improvement Business process reengineering	System proposal
	3	Determine business requirements	Interview JAD session Questionnaire Document analysis Observation	— Requirements definition
	4	Create use cases	Use case analysis	— Use cases
	5	Model processes	Data flow diagramming	— Process models
	6	Model data	Entity relationship modeling Normalization	— Data model

*Systems Analysis and Design 5th Edition, Author: Alan Dennis

Stakeholder Analysis

Main User: (a person who is trying to make an appointment to get tested):

- wants to have an easy way to make an appointment
- doesn't want to wait more than 5-10 minutes while trying to book this test
- wants to know how his data will be used further on
- wants information on what to do once arrived at the test center
- information on what is the current situation could be useful

Secretary: (using the system to make checks, validate data, and put new data in):

- wants a reliable system that will allow him/her to do the job
- has to be able to modify a mistake in the system

Administrator: (using the system to make checks, enable/disable users, and the only one authorized to delete data):

- wants to have well-structured data that is easy to maintain
- must have an error log file to track errors

The Health Department:

- wants an accurate overview of the situation in form of reports
- wants the system to offer these services without any errors
- wants the system to be able to cope with busy times
- wants to be well informed to take prevention methods and inform people ahead if unexpected situations occur

The local population:

- expects the situation to be managed in exemplary
- want to have the testing centers open and not close to schools or congested areas

Tourists:

- want to be informed of the current regulations
- want to have access to testing sites just as the local population
- be informed of where it is safe to travel

Businesses:

- want people to be able to come out so they can open for business
- want to prevent a potential spread to its customers or employees

1.2.2 Requirements Workshop (Use Cases & FURPS)

Use Cases (Corona test-center):

I User Registration

Preconditions: The user has successfully accessed the main page of the website, does not have an account already, and possesses CPR.

Success Guarantee: After completing the form if everything checks out (no duplicates or mistyped information) a success message will be sent back.

Main scenario:

1. User hits the main page of the website.

2. He decides to register for an account, so he hits the Register button.

3. A form is presented to the user that it's requiring some personal data to be sent back, like name, address, CPR, etc.

4. User completes the form and hits Continue.

5. The Server has validated the personal data and now it requires the user to choose a password and provide the email that will act as a username when the user logs in, so another form is presented.

6. User receives a confirmation message that the account was created.

Extensions:

2a. there is a problem at the server level, some data missing, or some connection problems that won't allow the server to send the proper response back. Admin will have to look at it and track the issue in the logs file. We must take precautions for this not to happen although and if it does it must be caught and handled.

4a. the Server notices there is already an account for this person or there is some conflict of data (e.g., CPR uniqueness), so it sends back an error message and restarts the process.

6a. if any problem occurs while processing the username and password the personal data mustn't be saved and the process resumes back from the beginning.

II User Logs in

Preconditions: The User has already signed up for an account and has a username and password.

Success Guarantee: User successfully logs in to his profile page where he can see his details and his history regarding testing and vaccination.

Main scenario:

1. User is on the main page of the website and hits the login option.

2. User is presented with the login page and provides his credentials.

3. User is authenticated and logged in.

Extensions:

3a. if the user couldn't be authenticated he is asked again for his credentials. If the problem persists a mechanism for recovering the password should be in place.

III User looking to book a test

Preconditions: The User is logged in

Success Guarantee: User has chosen his day and time for the test and a confirmation message has been returned to him after completing the process.

Main scenario:

1. User hits Book Test option.

1. User is presented with a quick option to book a test to the closest test center to his home and the next available time slot.

2. User accepts this option and hits Confirm option.

3. A confirmation message is presented with the details of the appointment.

Extensions:

2a. the User wants to select another time for the test, so he hits the Choose another option. A list of the available test centers will be displayed. Once a location is selected a calendar will be displayed. Selecting a specific

day in the calendar will prompt the display of the available times for that day and the user can then hit the Confirm option after going through these three steps (location-day-time).

2b. the Users spends too much time on the website (over 10 min) without booking a test, so he is asked to refresh the page for the latest available times.

IV User trying to cancel the appointment

Preconditions: The User is logged in.

Success Guarantee: The User has canceled the appointment and a confirmation message has been received.

Main scenario:

1. User goes to his Profile page.

2. User can see his appointments and hits the Cancel button next to them.

3. A confirmation message is received.

V Secretary wants Search and Edit options

Preconditions: The Secretary is logged in.

Success Guarantee: Secretary types in the name of the person are looking for and the person is presented back to the page. Each of the returned results has an Edit option that will open up the person's details that can be updated.

Main scenario:

1. Secretary has logged in and selects the option to show all persons in the system.

2. Secretary now has a search option and as letters are typed in only the names matching the query will display on the page.

3. Edit option is displayed next to each person.

4. Once the Edit option is hit a new page with the personal details opens up.

5. Information is edited, and the Update option is hit at the end.

6. A confirmation message is shown that the update was successful.

Extensions:

6a. the new information doesn't keep the consistency of the database, so an error message is displayed. The action is canceled.

VI Secretary trying to generate a report of the appointments

Preconditions: The Secretary is logged in.

Success Guarantee: A report with the appointments of the day is exported in a PDF or Excel file or just displayed on the screen for each test center.

Main Scenario:

1. Secretary goes to the Appointments page, information that can be filtered by location, by date, or both.

2. An option named Export is hit and an alert message for the selected option must be confirmed.

3. Secretary confirms the action, and an export file is generated that can be saved locally.

VII Administrator trying to delete some data

Preconditions: The Administrator is logged in.

Success Guarantee: The Administrator has entered the application, sees all the data that the Secretary has access to, and has permission to delete records in the database when needed. Once he chooses to delete an appointment that will trigger a response back with a confirmation message and the page should display the new information from the database. He has the same option for all the persons in the database.

VIII Administrator wants to disable a user

Preconditions: The Administrator is logged in.

Success Guarantee: The Administrator has successfully disabled a user.

Main Scenario:

1 Administrator is presented with his admin dashboard, and he can choose from options in the navigation bar to view the current users.

2. A view with the list of Users is presented and on the right side of each user, he has options to view this user's details or disable him. He chooses to disable him.

3. A confirmation message is received that the user has been disabled and the view is updated with the user's current status.

IX Administrator adding a new test center with the required info

Preconditions: The Administrator is logged in.

Success Guarantee: The Administrator has successfully added a new Test Center with an address and a number of daily available appointment places.

Main Scenario:

1. Administrator is presented with his admin dashboard, and he can choose from options in the navigation bar to view the current available Test Centers.

2. A view with the list of Test Centers is presented and below this list, an option to add a new Test Center is presented. He chooses that option.

3. A form is presented where he must fill up the Test Center's details, like address, number of available time slots, and the number of persons that can enter during one-time slot from which a capacity of the test center is established.

4. After successful completion of this form he is sent back to the list.

Extensions:

4a. if the Test Center is already in the list a warning message must be displayed and he is returned to the list.

Use cases for Test Results and Covid-19 prevention application:

I Secretary is updating the status of the tests by uploading an excel sheet or CSV file coming from the labs.

II User checks his/her result on his profile page.

III Administrator can check all the results and delete information if necessary.

IV A real-time map with current cases is shown on the front page as soon as new cases are confirmed.

V A notification is sent to persons who have been close to an infected person.

Use cases for Vaccine Center application:

I User checks books his vaccine at a Vaccine Center.

I Secretary checks and administers the vaccine list.

III Administrator can put in all the vaccines available.

FURPS

Functionality:

- has to be easy to use, application is intended for every citizen in the capital area that wants to get tested.
  ➔ important buttons should be placed at eye level and the color should spark attention.
- easy to navigate because we want users to sign in and schedule their test without much trouble.
  ➔ at all times it should be clear where you are in the application and the most important actions should stay visible at the top of the page.

Usability:

- the application would be used by at least three actors: the User who will make an appointment, the Secretary who will verify and maintain the user's details, appointments, and results, and the admin who can disable users, and delete records from the database.
  ➔ the right permissions must be set.
  ➔ they will access the application for different services so we would have a common interface for login and based on their permissions we will present them with their appropriate dashboard with the actions that they are allowed to perform.
- We would have a common interface for testing appointments, results, and vaccination.
  ➔ we could separate these functionalities later on in smaller applications (microservices) that make up a bigger eco-system so that we won't load one server with all the requests.

Reliability:

- we have to take into consideration that the application would serve a population of two million people in the metropolitan area of Copenhagen and as many as 200.000 daily visitors.

https://www.sst.dk/en/english/corona-eng/status-of-the-epidemic/covid-19-updates-statistics-and-charts

➔ choosing the right technologies to support that infrastructure would save us time and effort in the long run.

Performance:

- the application must respond to multiple requests per second whilst keeping the database consistent with the latest changes.
  ➔ We should send the important information in one request so that we save time on the trips made back and forth between client and server.

Supportability:

- the application must perform on all major web-browsers as well as different devices (e.g., phone, tablets, laptops)
  ➔ The application will render the information in HTML using Thymeleaf template engine and Bootstrap framework for structuring this format on the client's web browser. All major web browsers are capable of processing this at this time. CSS can be used to resize the view depending on the device that is accessing the application.

1.3 Project limitation (Identifying Project Scope)

1.3.1 Early Timeboxing

Task Name	Duration
1. Business Analysis	**4 days**
1.1 Identify High-Level Requirements	1 day
1.2 Detailed Use Cases	1 day
1.3 Prioritize Requirements & Define Scope	1 day
1.4 Work Report	1 day
2. Preliminary Design	**7 days**
2.1 Data models	2 days
2.2 System architecture	2 days
2.3 User Interfaces	2 days
2.4 Work Report	1 day
3. Implementation	**9 days**
3.1 Schema definition	2 days
3.2 Programming & Testing	4 days
3.3 Debugging	2 days
3.4 Work Report	1 day
Total	**20 days**

The activities here are not necessarily listed in the order they are worked on. The purpose of this estimate is to have an overall view of how much time each part of the project should take. Each day of work is approximate 8 hours of good work.

Human Resources:

- one full stack developer

1.3.2 Risk Analysis

RISK ASSESSMENT MATRIX

RISK RATING KEY	LOW	MEDIUM	HIGH	EXTREME
	0 – ACCEPTABLE OK TO PROCEED	1 – ALARP (as low as reasonably practicable) TAKE MITIGATION EFFORTS	2 – GENERALLY UNACCEPTABLE SEEK SUPPORT	3 – INTOLERABLE PLACE EVENT ON HOLD

	SEVERITY			
	ACCEPTABLE LITTLE TO NO EFFECT ON EVENT	TOLERABLE EFFECTS ARE FELT, BUT NOT CRITICAL TO OUTCOME	UNDESIRABLE SERIOUS IMPACT TO THE COURSE OF ACTION AND OUTCOME	INTOLERABLE COULD RESULT IN DISASTER
LIKELIHOOD				
IMPROBABLE RISK IS UNLIKELY TO OCCUR	loss of interest	integration with existing system	development costs	intangible benefits
POSSIBLE RISK WILL LIKELY OCCUR	internet service interruption	scalability	loss of source code	familiarity with technology
PROBABLE RISK WILL OCCUR	new domain for SWD	small team	time limit	data security

project complexity+small team+time limit+data security

1.3.3 Feasibility Study

We have defined our Use cases after a Stakeholder Analysis, a Business Analysis, and a Requirements Workshop. As we worked our way in marking down the requirements for this System we chose or were constraint to use by the context of this project a specific software development methodology and technologies that were affordable by cost and time.

Taking into consideration these factors we have worked out a Risk Analysis Matrix which revealed that time and security would be our biggest impediments in developing this System.

The time given to develop this System is four weeks and we have set it as a priority to follow certain guidelines and code standards that in the long term will prove useful yet in the short time will consume more resources.

We conclude that we can develop this System with some limitations in the features available in the first release, leaving room for improvements in the feature.

1.4 Proposed Solution

After analyzing the goals and stakes we've set our mission to have a minimum viable product with enough features to prove its worth. We will develop this web application that will sit on a remote server of a cloud provider and this way we do not have to worry about data loss because cloud providers have information backed up in multiple places.

The application will serve three users:

Regular User: → Register, Login, Make an appointment, Cancel an appointment, Update personal info.

Secretary: → Login, Check/Edit user's info or his appointments.

Admin: → Login, Disable user, Delete records from the database.

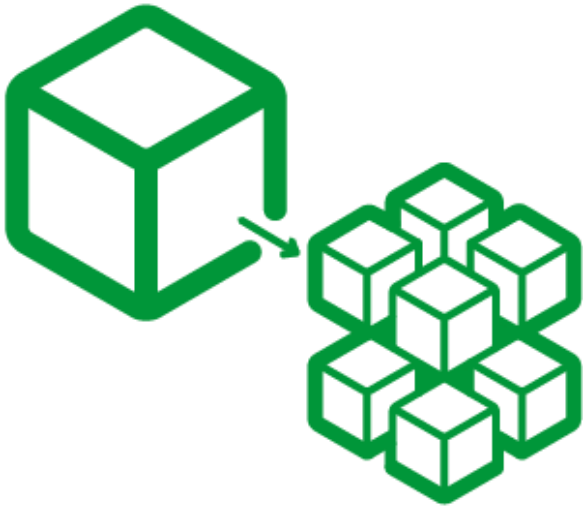
1.4.1 Features for another release

We can only differentiate if we manage to succeed in designing this System leaving room for future improvements like integration with other specialized services (e.g., NemID login).

We can also make use of the public data available to design an interactive real-time map of positive tests found in the Copenhagen area to keep people alert of where an undesirable place would be to travel. We also want to implement a future where people will get notified if they have been close to an infected person whilst keeping that person's anonymity.

Further on technologies like Kubernetes could bring an additional payoff in terms of scalability and profit since they can scale up computing resources to very busy times and scale down when requests are low. We don't have to buy hardware or spend money on maintenance this way.

We want to divide this application into smaller microservices that can interact with each other rather than having a single server responding to all the requests. We could containerize our microservices with a tool like Docker and implement a Continuous Integration and Continuous Development strategy with automated tests. This way we won't experience downtimes and we can be sure that our new code is following our current one. We have to mention that our current knowledge of these technologies is limited, and time does not allow us to implement them.



*decomposing an application in multiple services

II Elaboration

2.1 Software Design

Whilst developing the use cases into software components we will use some of the principles of GRASP.

Information Expert:

When making the transition from the Conceptual Model to the Software Objects Model we will use this concept to establish the responsibility of one Class to carry the logic of one action. That Class will now contain behavior that will perform that specific action. The actions of that class will modify the state of the Objects that are involved in that use case.

Low Coupling:

In general, is best to keep the behavior of one class to a strict minimum of what it needs to perform the logic of the required operation so that we don't rely heavily on just one class to perform all the logic. This means that we should encapsulate in one class exactly what that class is meant to be doing and use other classes that are suited to carry the other parts of the logic that are more suited for. In this way, we support low dependency and increased reuse.

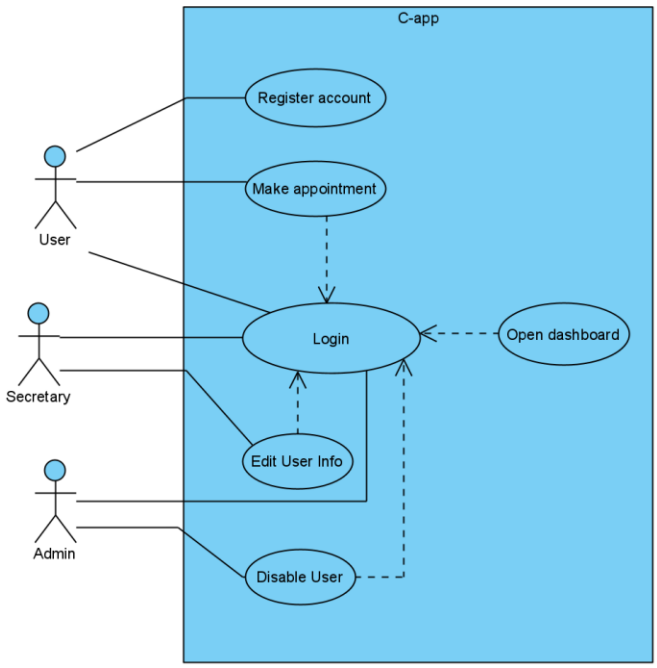
When two objects are loosely coupled they can interact but have very little knowledge of each other.

➔ In our project we will have domain model classes that are used as information experts in creating objects, and service classes that carry the possible behaviors of those objects depending on their state.

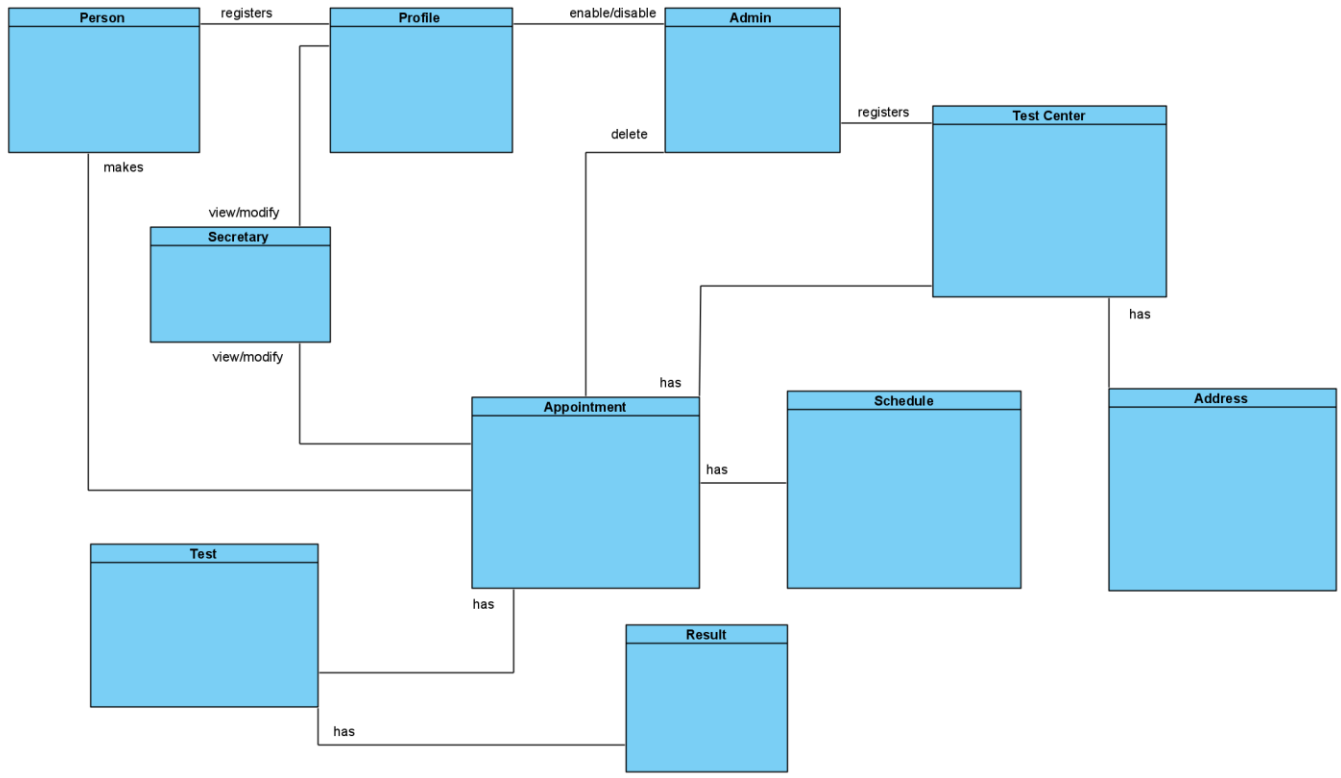
Other Design Principles to be used:

- Identify the aspects of the application that vary and separate them from what stays the same.
- Program to interfaces not implementations.
- Favor Composition over Inheritance.
- Classes should be open for extension but closed for modification.

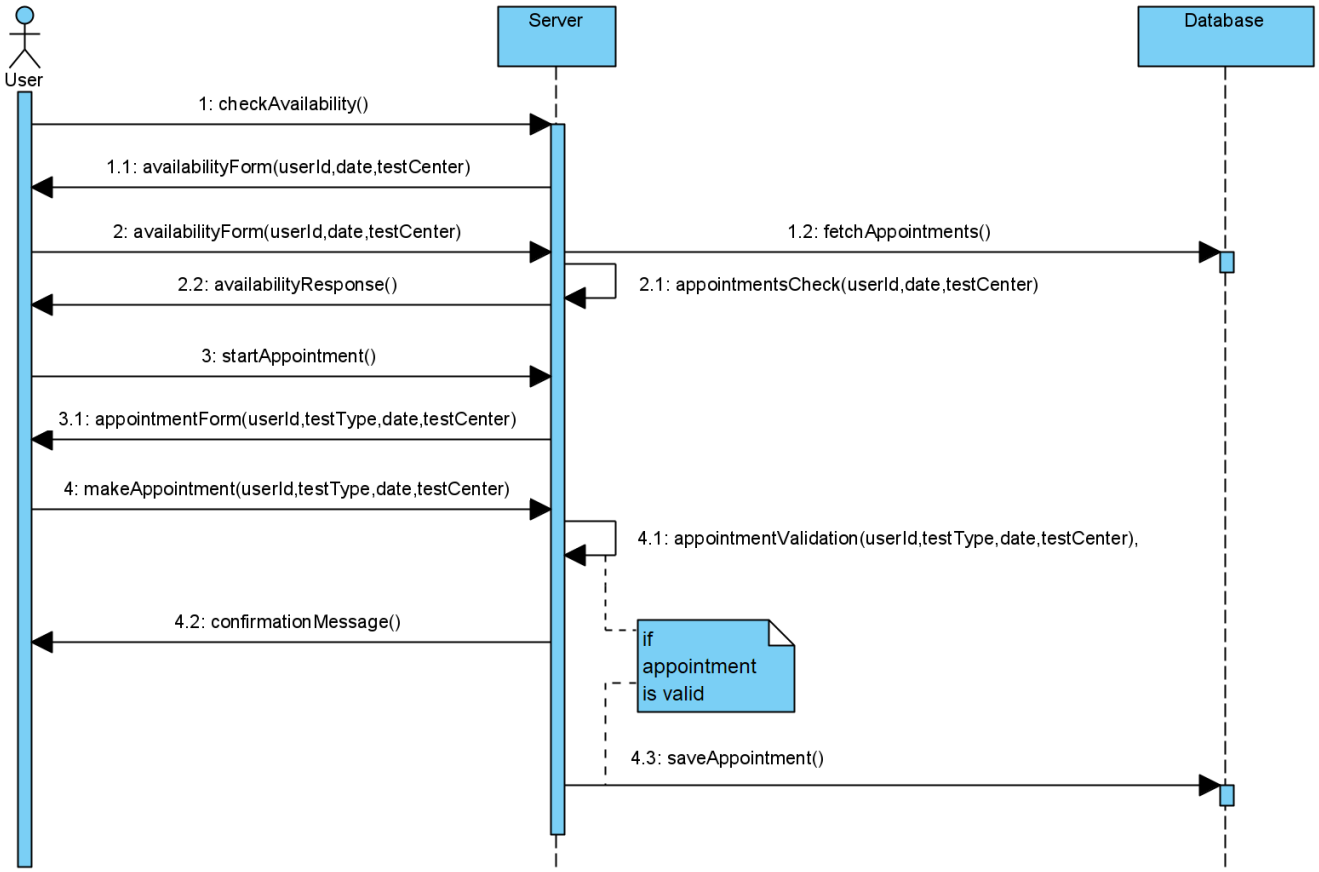
2.1.1 Use Case Diagram



2.1.2 Conceptual Model

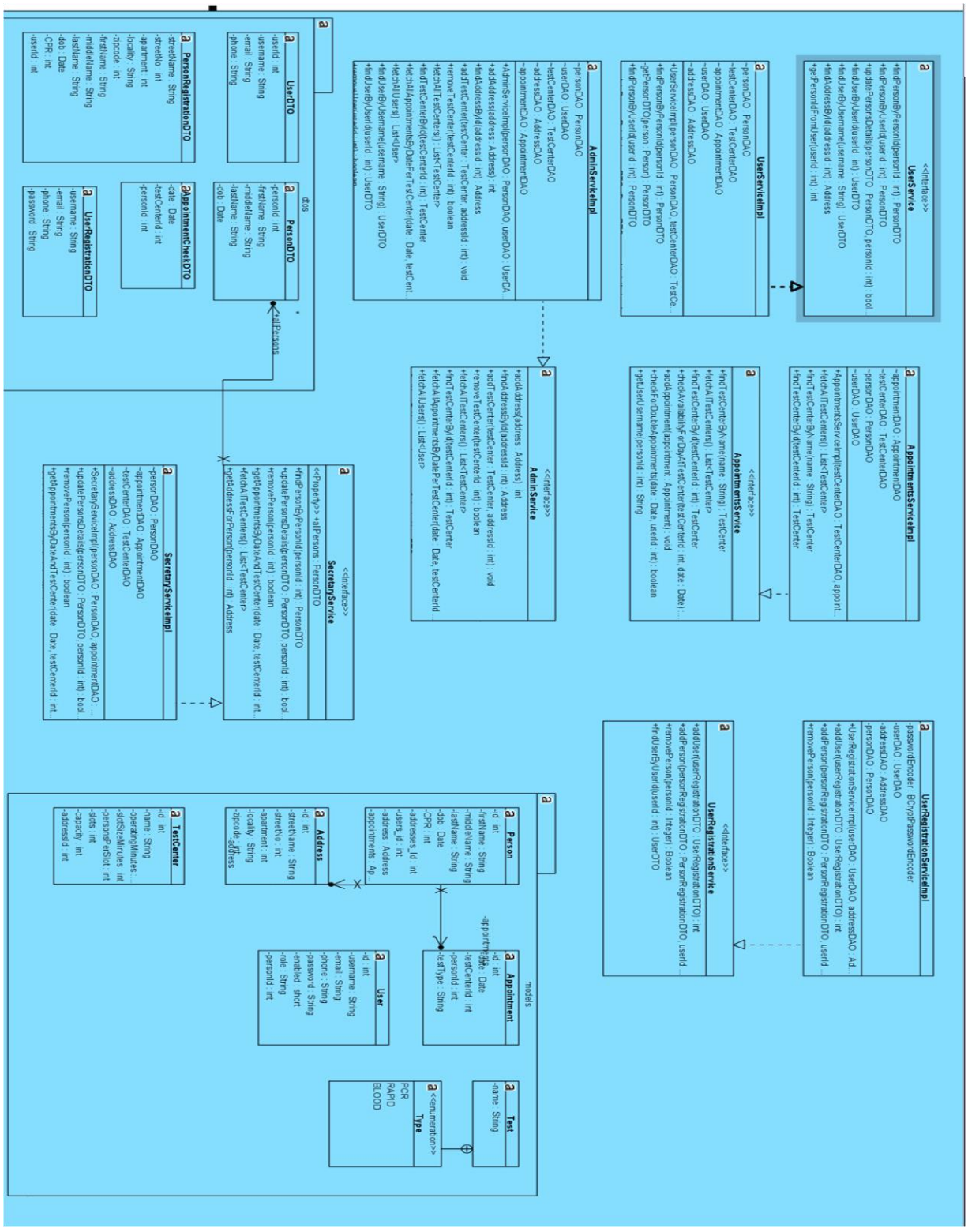


2.1.3 Make Appointment Sequence Diagram

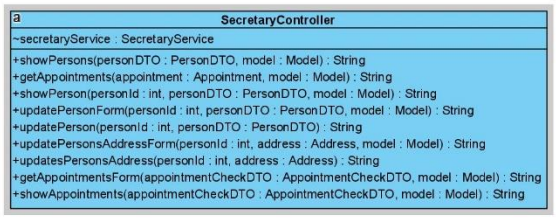


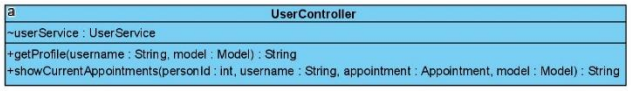
2.1.3 Package Diagrams

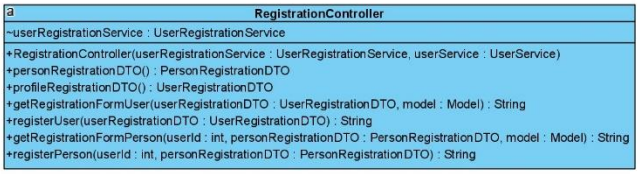
1. Business Layer

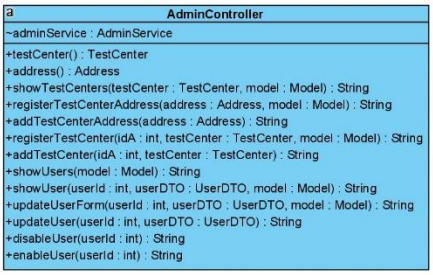


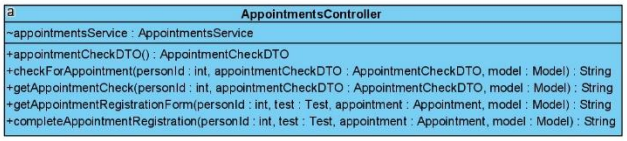
2. Presentation Layer



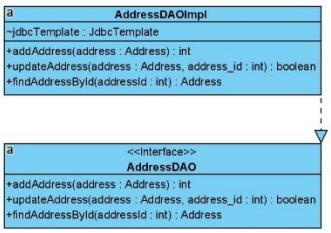


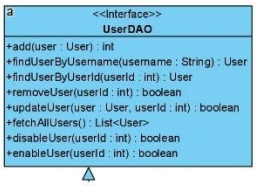


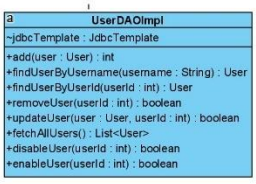


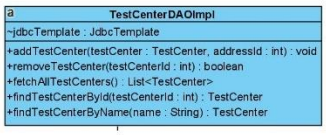


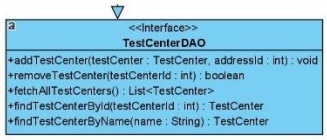
3. Data Layer

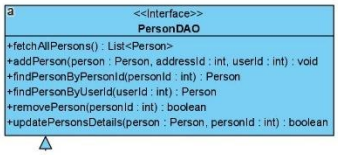


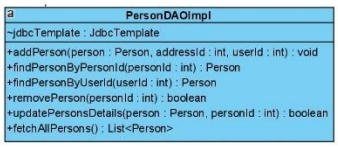


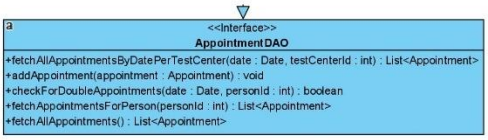








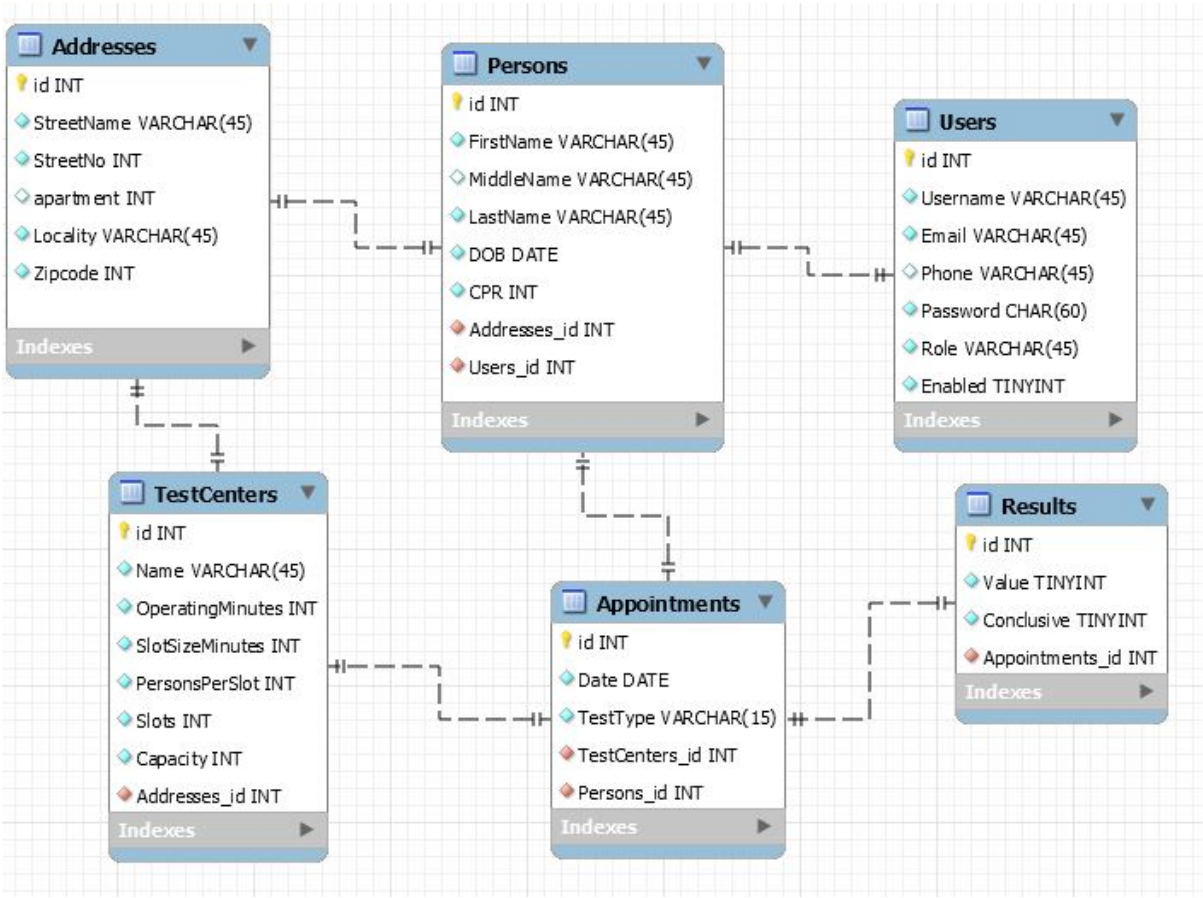




2.2 Database Design

The information that these tables hold was extracted with a Behavior Driven Design in mind thinking about exactly is the type of information that the user needs at the other end of the communication channel and what we need from him to persist in the database to be able to service them again.

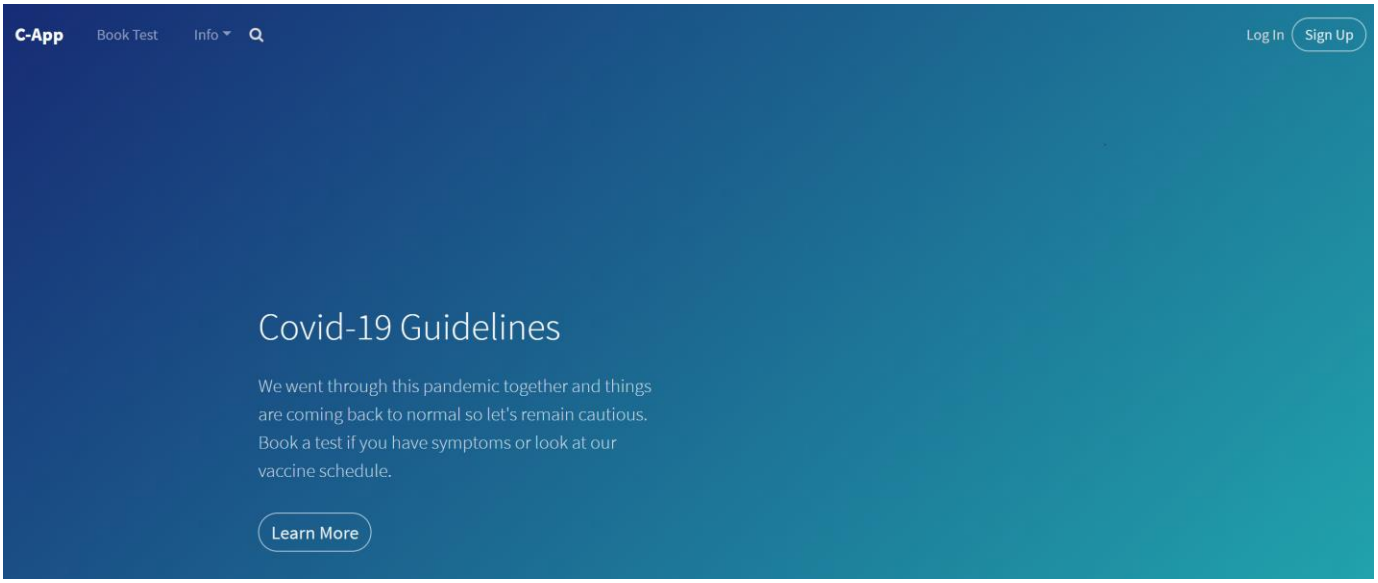
2.2.1 EER Diagram



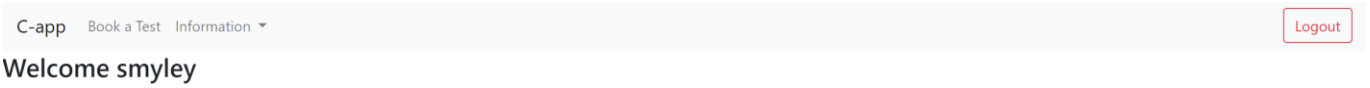
This schema went through a series of changes and now it is in a simple and stable form that our application can safely use. We can refer to this schema from other services that we might implement.

2.3 UI Design Prototypes

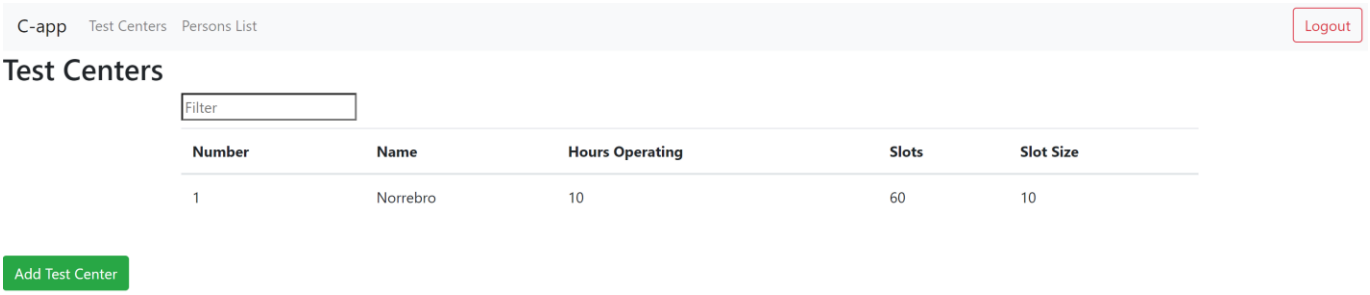
2.3.1 Login Interface



2.3.2 User Dashboard



2.3.3 Admin Dashboard



III Construction

3.1 Construction Plan

1. Implement a solution in a multitiered architecture with a production-ready look.
    - the Client-tier (front-end): this node is responsible for the user interface. The client will be using a web browser that will send HTTP requests to the application's Server. The communication is done through a Web API that will be able to process these requests.
    - the Application-tier (back-end): this node represents the business logic of the application and will hold all the interfaces that will provide the desired functions and the data interfaces for persistence.
    - the Data-store-tier: this is the database that keeps the application's data.

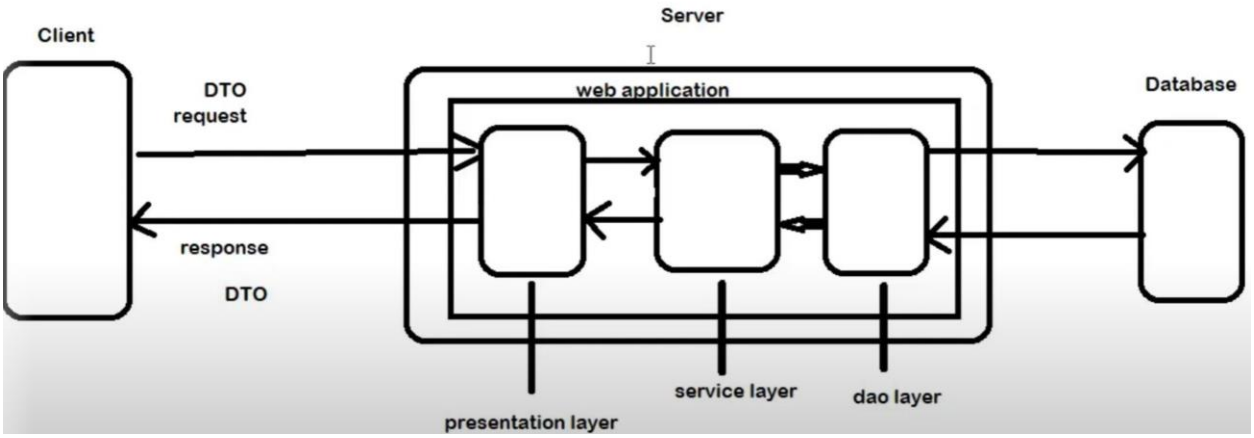
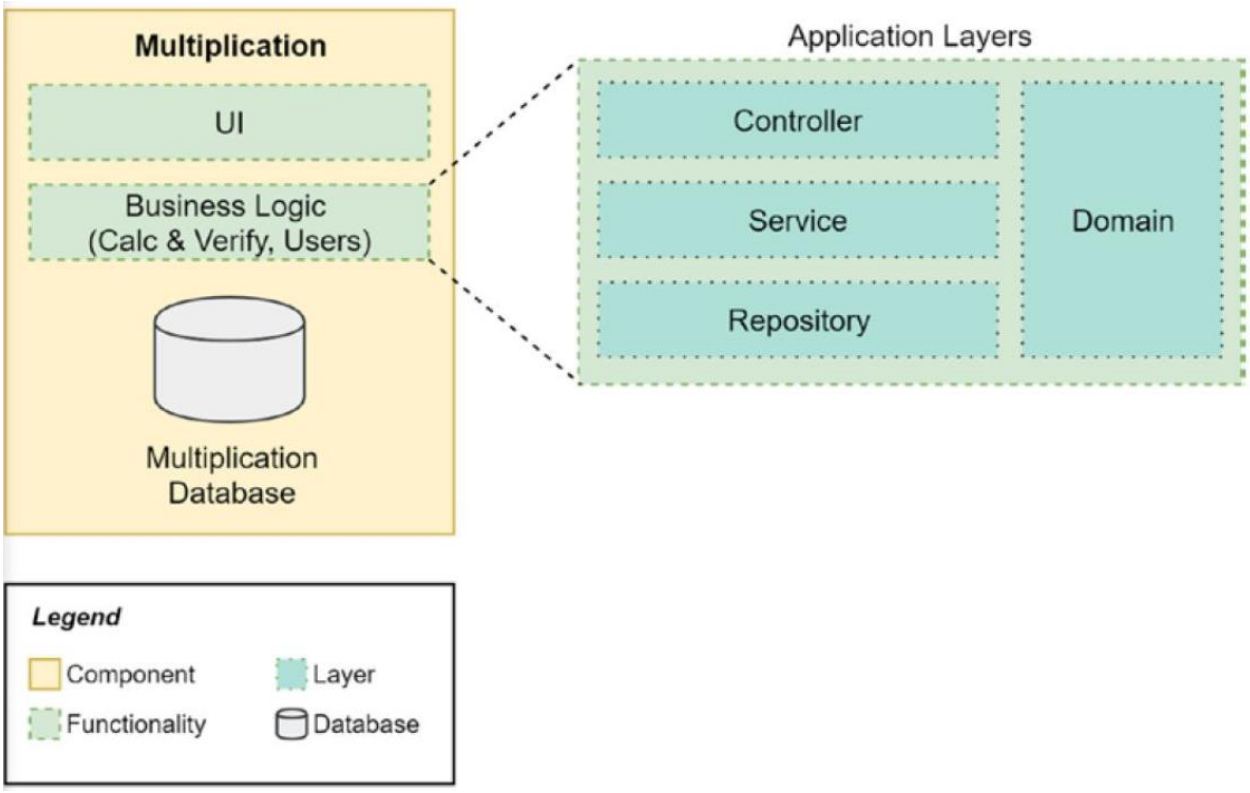


*image of a three-tier application

2. Develop the Application-tier with respect to the layers of the application:
    - the business layer where we include the classes that model our domain (entities) and the business logic (services).
    - the presentation layer represented by the Controller classes which will provide functionality to the Web Client. The API implementation will reside here.
    - the data layer that will be responsible for persisting the entities in a data storage and it can include Data Access Objects (DAOs),

which work with a direct representation of the database model, or Repository classes which are domain-centric. Repositories could use DAOs whenever the entity doesn't match the database model.



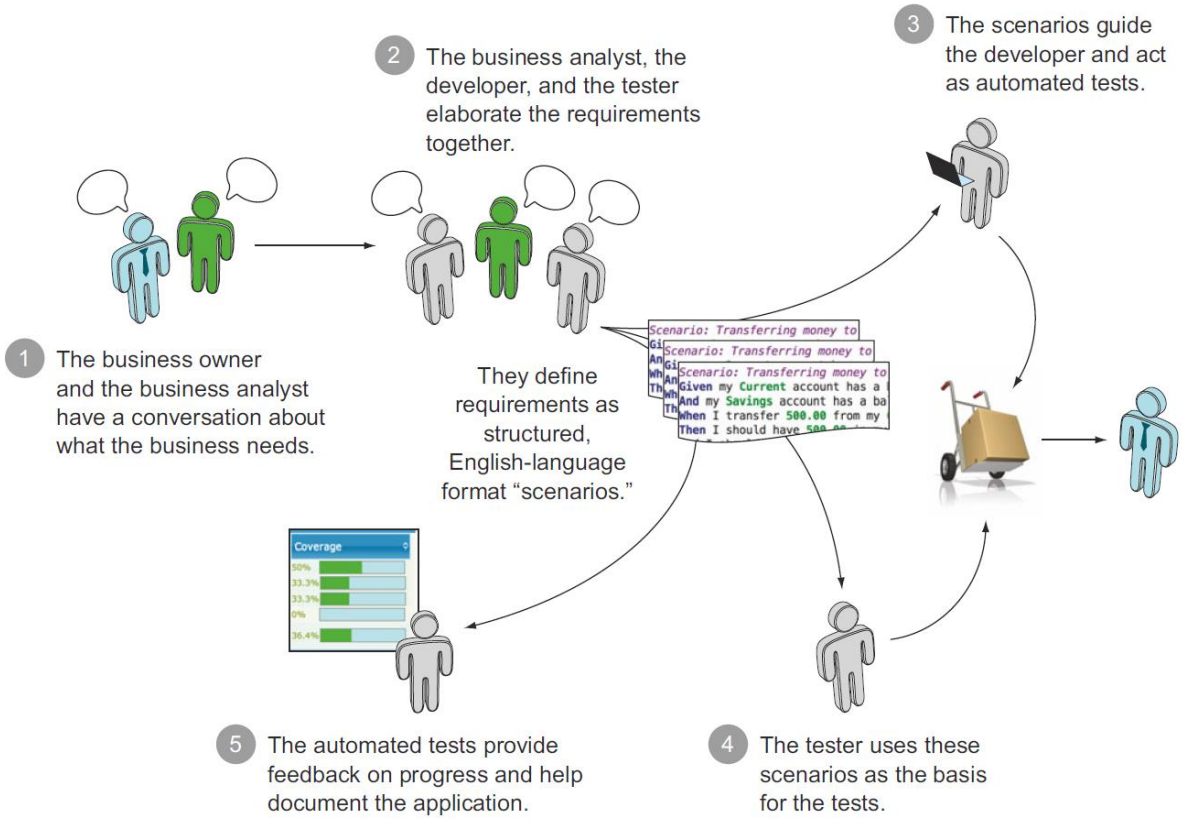
*Learn Microservices with Spring Boot, Author: Moises Macero

Advantages of using this software architecture are intrinsically related to the fact of decoupling layers.

- The domain part is isolated and independent from the solution, instead of mixed with the interface or database specifics.
- Non-business layers are interchangeable (e.g., changing the database for a file storage solution).
- Clear separation of responsibilities: a class to handle database storage of the objects, a class for the API implementation and another class for the business logic.

3. Prototype a UI to better understand what is to be done and test the user interaction with the application. It can be a paper prototype although a professional visualizing tool would be more efficient for the users and developers.

4. We will follow the guidelines of Behavior-Driven-Design (BDD) and Test-Driven-Design (TDD).
   - Start with the Service in mind which is the information that our application should provide considering what users want/need without thinking yet about how to implement that service.
   - Information is data put together so our domain models will start revealing once we know what the informational needs are.
   - To prepare that information some calculations will need to be done so we will use a service class that will put the pieces together and prepares that data in the desired format.
   - We expose our services through an endpoint that can return data in the needed format (HTML, JSON).



*BDD in Action: Behavior-driven development for the whole software lifecycle, Author: John Ferguson Smart

3.2 Test Driven Development Plan

OR AAA-pattern → Arrange, Act and Assert

- **Given** that we don't have an account
- **When** we want to register for one
- **Then** the system will return to us a form with the required information to be provided to set up an account

- **Given** that we provide the necessary information in the form presented to us
- **When** we want to register for an account
- **Then** the system will check it's database for duplicates or uniqueness constraints and return a response

- **Given** that we have successfully registered our personal information
- **When** we want to register for an account
- **Then** the system will guide us to choose a username which will be the email address and a password to secure that account

- **Given** that we have accessed the website
- **When** we want to see general information or the steps of getting tested
- **Then** the main page will present that information without being constrained to make an account

- **Given** that we have an account and we have been authenticated
- **When** we hit the Book Test button

- **Then** the application presents us a list of all the locations where tests are carried

- **Given** that we are signed in the application
- **When** we have expressed our intention to make an appointment by clicking the Book Test button
- **Then** based on our home address, the closest location with the next available timeslot will be provided as a shortcut to make the appointment. Choose another location option should also be presented.

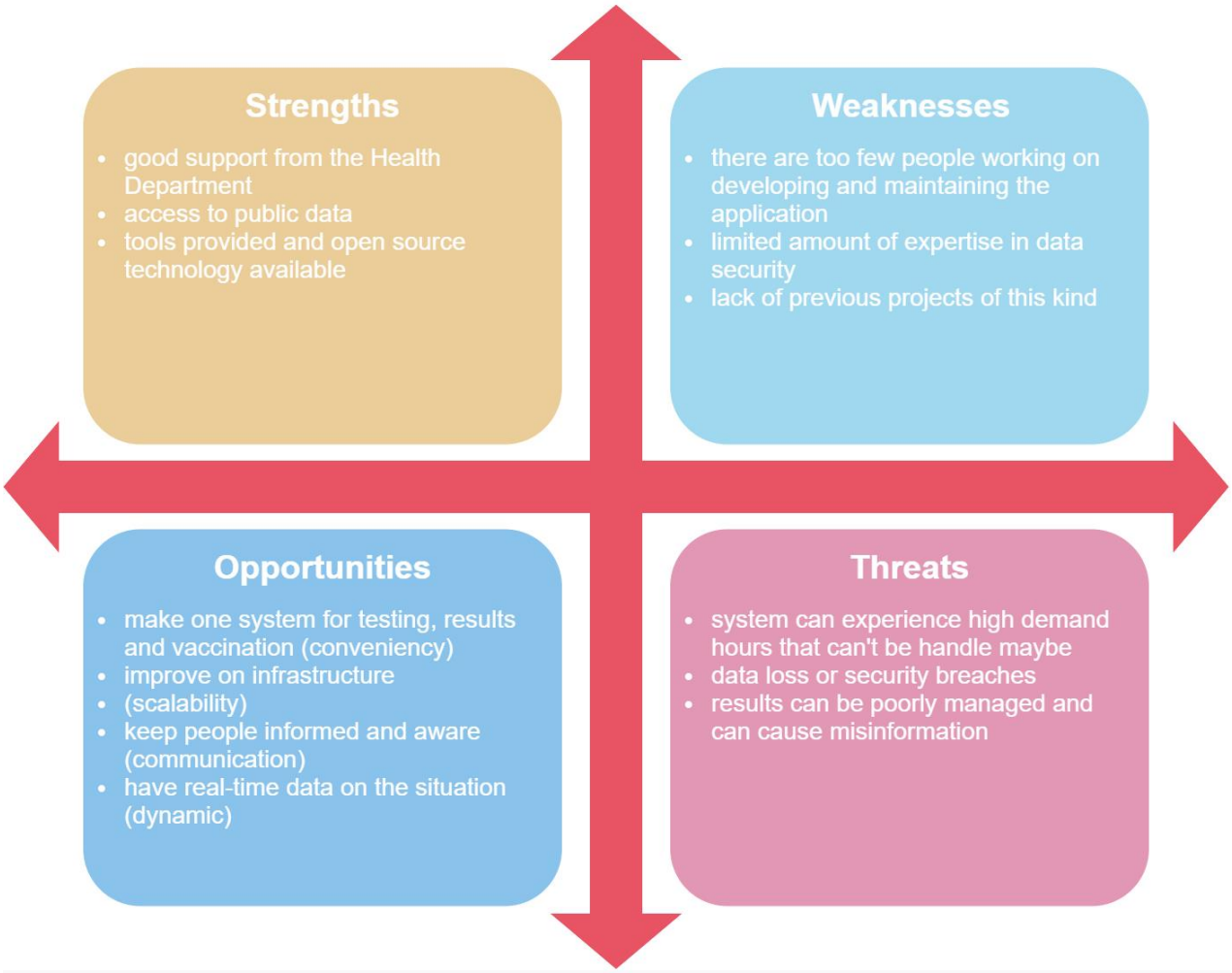
- **Given** that we chose to go to another location for testing
- **When** we are in the process of making an appointment
- **Then** all the locations for corona testing will be displayed to us and a map will present pinpoints for those locations

- **Given** that we have selected a location other than the suggested one
- **When** we are in the process of making an appointment
- **Then** a calendar will be displayed with the next available day selected, and a timetable next to it with the open timeslots for that day. We should be able to select another day also.

- **Given** that we have expressed our option for a location, a day, and a time for the corona test and hit the Submit button
- **When** we are in the process of making an appointment
- **Then** a success message will be presented to us along with a mail confirmation for the appointment made

Appendix

SWOT



Supplementary Specification

We must mention that we are not ready for production since this application does not adhere to all the security standards. It will be tested while running on a Tomcat web server provided by the Spring Boot framework along with some other dependencies of the project that will be mentioned in the *.pom* file. We will be using Maven for dependency management. We will also rely on a remote instance of a MySQL database running on gearhost.com. The general architecture of the program is similar to one that would be in production.

We are using Open-Source technologies like:

- database engine →MySQL
- database hosting →gearhost.com
- application framework →Spring Boot
- servlet container →Tomcat (included with Spring Boot framework)
- software development →JDK 11
- web application hosting service →Heroku
- unit testing →Junit
- integration testing →SpringBootTest
- repository hosting →Github

We are using Tools like:

- Maven for project structure and dependency management
- IntelliJ IDEA as editor
- Postman for API testing
- Bootstrap Studio for interface design
- Git for version control
- Visual Paradigm for software design

Work Reports

1st Iteration:

– business analysis is conducted (Artifact: Stakeholders Goal Analysis & SWOT Analysis).

– requirements are established, and the scope of the project is being defined with the features that we are going to design and implement (Artifact: Use Cases, FURPS, Risk Analysis Matrix).

– a decision is made regarding the technologies that we are going to use to implement our solution (Artifact: Supplementary Specification).

2nd Iteration:

– having the use cases in mind a list of services and of data that is needed to fulfill the requirements is drafted.

3rd Iteration:

– diagrams are being drawn, Use Case Model, fully dressed Use Cases, Sequence Diagrams, followed by a first draft of the Conceptual Model.

4th Iteration:

– the database schema is being drafted.

– models are translated to code and the general architecture of the System is prepared.

5th Iteration:

– prototypes of User Interfaces are designed.

6th Iteration:

– continued work in programming: DAO layer, service layer, and presentation layer.

7th Iteration:

– continued work in programming and debugging, testing services.

Acceptance Testing

Scenario	Passed Test
I User can register an account.	Y
II User can Log in with username and password.	Y
III User can check it's Profile where he can see a List with his appointments.	Y
IV User can check the availability of a specific Test Center in a certain day.	Y
V User can make an appointment to get tested.	Y
VI Secretary can Log in with username and password.	Y
VII Secretary can check and update personal details of users.	Y
VII Secretary can see all the appointments and filter them by date and Test center.	Y
VIII Admin can Log in with username and password.	Y
IX Admin can register a new Test center.	Y
X Admin can enable or disable users.	Y

Glossary

scalability: is the property of a system to handle a growing amount of work by adding resources to the system.

microservice architecture: a variant of the service-oriented architecture structural style, arranges an application as a collection of loosely coupled services.

server: a piece of computer hardware or software that provides functionality for other programs or devices, called "clients".

server request: a message from a client to a server that includes within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

cloud computing: represents the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user.

Kubernetes: an open-source container orchestration system for automating computer application deployment, scaling, and management.

API: a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other services.

database schema: describes the tables in a database in a formal language supported by the database management system.

References

1. Head First Design Patterns, Author: Eric Freeman
2. Domain-driven design Tackling Complexity in the Heart of Software, Author: Eric Evans
3. BDD in Action Behavior-Driven Development for the whole software lifecycle, Author: John Ferguson Smart
4. Systems Analysis and Design, Author: Alan Dennis
5. Patterns of Enterprise Application Architecture, Author: Martin Fowler
6. Spring Boot In Action, Author: Craig Walls
7. Learn Microservices with Spring Boot, Author: Moises Macero
8. The Java EE Architect's Handbook: How to be a successful application architect for Java EE applications, Author: Derek C. Ashmore
9. Building Java Programs, Author: Stuart Reges
10. Java The Complete Reference, Author: Herbert Schildt
11. Applying UML and Patterns, Author: Craig Larman