

**Report PiDev 2018**

WRITTEN BY:

* *BEN HAJ YOUSSEF Haythem*
* *MEGDICHE Hamdi*
* *MNASRI Haythem*
* *ZOUAOUI Ilyes*
* *ZOUGHLAMI Imen*

CLASS: *4TWIN2*

TOPIC: *Epione*

**Table of contents**

[General introduction 1](#_Toc528831557)

[CHAPTER 1 : Main conception & Used methodology (Phase 0) 3](#_Toc528831558)

[Introduction 4](#_Toc528831559)

[1. Study of existing solutions 4](#_Toc528831560)

[1.1. Issue 4](#_Toc528831561)

[1.2. Target audience 4](#_Toc528831562)

[1.3. Similar solutions 5](#_Toc528831563)

[2. Analysis 7](#_Toc528831564)

[2.1. Functional requirements 7](#_Toc528831565)

[2.2. Actors 8](#_Toc528831566)

[2.3. General Use Case Diagram 9](#_Toc528831567)

[2.4. Analysis Class Diagram 10](#_Toc528831568)

[2.5. System sequence diagram 11](#_Toc528831569)

[2.6. Non-functional requirements 12](#_Toc528831570)

[Conclusion 13](#_Toc528831571)

[3. Physical architecture 14](#_Toc528831572)

[3.1. Global architecture 14](#_Toc528831573)

[3.2. Environment 14](#_Toc528831574)

[3.2.1. Technologies 14](#_Toc528831575)

[3.2.2. Web service 15](#_Toc528831576)

[3.3. Deployment Diagram 15](#_Toc528831577)

[Conclusion 16](#_Toc528831578)

[4. Project Management 17](#_Toc528831579)

[Introduction 17](#_Toc528831580)

[4.1. Software Development Process 17](#_Toc528831581)

[4.2. Version control repository Manager (GIT) 19](#_Toc528831582)

[Conclusion 19](#_Toc528831583)

[5. Mockups 20](#_Toc528831584)

[General conclusion 25](#_Toc528831585)

[CHAPTER2: (Phase 1: JavaEE / REST API) 26](#_Toc528831586)

[Introduction 27](#_Toc528831587)

[6. Phase 1 Overview 27](#_Toc528831588)

[6.1. Introduction 27](#_Toc528831589)

[6.2. Approach 27](#_Toc528831590)

[6.3. Workflow 28](#_Toc528831591)

[6.3.1. Version Control 28](#_Toc528831592)

[6.3.2. Overall Process 28](#_Toc528831593)

[6.4. Conclusion 28](#_Toc528831594)

[7. Project structure 29](#_Toc528831595)

[Introduction 29](#_Toc528831596)

[7.1. Project Structure 29](#_Toc528831597)

[7.1.1. EJB Project 29](#_Toc528831598)

[7.1.2. WAR Web Project 29](#_Toc528831599)

[7.1.3. EAR Archive Project 30](#_Toc528831600)

[7.2. Specifications 30](#_Toc528831601)

[7.2.1. JPA 30](#_Toc528831602)

[7.2.2. JAX-RS 30](#_Toc528831603)

[7.3. Conclusion 30](#_Toc528831604)

[8. Design 31](#_Toc528831605)

[Introduction 31](#_Toc528831606)

[8.1. Class Diagram 31](#_Toc528831607)

[8.1.1. Persistence package 31](#_Toc528831608)

[8.1.2. Interface package 31](#_Toc528831609)

[8.1.3. Business package 31](#_Toc528831610)

[8.1.4. Resources package 31](#_Toc528831611)

[8.2. Object Sequence Diagram 31](#_Toc528831612)

[8.3. Strategies and Associations 32](#_Toc528831613)

[8.4. Conclusion 32](#_Toc528831614)

[9. EE Implementation 32](#_Toc528831615)

[Introduction 32](#_Toc528831616)

[9.1. Package 32](#_Toc528831617)

[9.2. Conclusion 32](#_Toc528831618)

[10. Web Services 33](#_Toc528831619)

[Introduction 33](#_Toc528831620)

[10.1. Nomenclature 33](#_Toc528831621)

[10.2. Conclusion 33](#_Toc528831622)

[11. Server-Side Control 34](#_Toc528831623)

[11.1. Introduction 34](#_Toc528831624)

[Conclusion 34](#_Toc528831625)

[12. General Conclusion 35](#_Toc528831626)

**Link of the application deployed on GIT:** https://gitlab.com/HamdiMegdiche/epione

**List of Figures**

[Figure 1 : Web site Tuotempo 5](#_Toc526804941)

[Figure 2 : Web site DocteurDirect 6](#_Toc526804942)

[Figure 3 : Web site Bonjour-sante 6](#_Toc526804943)

[Figure 4 : General Use case diagram 9](#_Toc526804944)

[Figure 5 : Analysis class diagram 10](#_Toc526804945)

[Figure 6 : Make appointment system sequence diagram 11](#_Toc526804946)

[Figure 7 : Add schedule sequence diagram 12](#_Toc526804947)

[Figure 8 : Global architecture 14](#_Toc526804948)

[Figure 9 : Deployment diagram 15](#_Toc526804949)

[Figure 10 : Home page 20](#_Toc526804950)

[Figure 11 : Search Result Page 21](#_Toc526804951)

[Figure 12 : Sign IN / Sign UP interface 22](#_Toc526804952)

[Figure 13 : Appointment booking page 23](#_Toc526804953)

[Figure 14 : Profile page 24](#_Toc526804954)

# General introduction

In the context of our integration and development project of the 4th year web and internet technology (TWIN), we were assigned the realization of a project in which we directly apply the skills acquired throughout this semester. We chose Health caring project, which is a platform that offers making appointments between doctors and patients.

Many people do not realize, and often underestimate the importance of good health.  
Health, as they say, is wealth and good health is necessary to carry out daily tasks.

The everyday choices you make can have a big effect on your health, your life and your wellbeing. Eating well, keeping active, not smoking or drinking too much alcohol and looking after your mental health can all help you stay well and enjoy life.

When discussing about health, we say doctors also, they save lives, but their importance goes far beyond that. Doctors also make a difference by helping patients minimize pain, recover from a disease faster or learn to live with a disabling injury. A patient's ability to enjoy life, even if they cannot be healed, makes a huge difference to them and to their families. If they can go back to work after an illness that benefits their employer too, and that is only part of what makes doctors important to society. Our platform will provide many services to facilitate the doctor and patient daily live.

“Health caring” is a project undertaken as a mandatory requirement for the course “PI-dev 4TWIN” that is being conducted mutually by MNASRI Haythem, MEGDICHE Hamdi, ZOUAOUI Ilyes, ZOUGHLAMI Imen and BEN HAJ YOUSSEF Haythem all of whom are members of 4TWIN2 class of 2018/2019 and the EPIONE team.   
The aim of the course is to provide a distributed environment to develop a Web application using multiple different technologies and synchronize the communication between them using APIs.

The project was carried out in four steps:

* The definition of the project, which consists of editing the functional specifications: The design and implementation of the project.
* The Implementation Business party and the exposition of web services using JEE technology
* The realization of a part of the project using .NET technology and the movable part using a cross platform Ionic
* The implementation Customer part respecting standards of Angular JS.

In the next chapters, we will mention details about the development of each of these parts being as specific as possible for the reader of this report.

CHAPTER 1 : Main conception   
&   
Used methodology (Phase 0)

Introduction

In this report we have to explain the initial phase of the project process, we will call it phase 0’s review. We will focus, as you will notice below on client’s needs, whishes, main features, vision, obviously technologies and environment.

The tutors proposed the theme and we are tasked to expand the main feature of the application, which can be resumed in facilitating the process of making an appointment.  
In this document, there is overall description of the Web Project. It includes what we aim to do. There is also description of the analyzing phase, whole descriptions of the design intended, scenarios and technologies are included in this document.

1. Study of existing solutions
   1. Issue

In everyday life, each individual is faced with a busy schedule. He has to wait for hours to make an appointment, there are days when we are afraid of having to make an appointment with a certain doctors, because he fears the waiting time and in the end, he does not find an appointment available or appropriate with their availability.

To overcome this lack of flexibility and speed, we thought to create this application, which consists of making an appointment online in just a few clicks in order to ensure better management of everyday life and change the spirit of our community to respect and understand the principles of the appointment. Together, we can make a difference.

* 1. Target audience

The main actors as we will see in detail later would be the patients seeking appointments. To describe the market better, we tried to understand the client’s needs, thereafter we were able to identify to which community the client will belong and it’s simply anyone with a Computer or Mobile Phone and obviously an Internet connection.

* 1. Similar solutions

This step is designed to start any IT project, define the operating context, or the business process, and identify the different imperfections in existing applications to correct and improve them in our new application. To detect existing problems, we have identified the following deficiencies in these applications:

* The confirmation of the appointment is not immediate and it requires the intervention doctor to confirm the date chosen by the client.
* The application may not be understandable to the user.
* Lack of options that provide comfort to clients to encourage them to choose to take online appointments.
* There is a lack of tracking appointments and sending reminder notifications.
* Non-computerized data (Customer records).



Figure 1 : Web site Tuotempo



Figure 2 : Web site DocteurDirect



Figure 3 : Web site Bonjour-sante

1. Analysis

During this phase, we will explain the main features, different actors and aimed interactions between users of the application according to the product owner .

* 1. Functional requirements
* **Manage Client Appointement**

The patient selects the specialty of the doctor and the city (for the place he can choose geolocation), he consults the doctor's availability and make an appointment with him, he can optionally write a message to that doctor. The confirmation of the appointment is done by sending an e-mail to the patient and by notification also the patient can cancel or update his appointments with doctors.

* **Manage Doctor Appointment**

The doctor can consult his daily program, specifies his availability by a 15 min slot, also he can cancel an appointment with a patient, and take a vacation.

* **Rate**

The patient can rate his visit to the doctor from 1 to 5.

* **Manage Patient Profile**

The patient can consult and add and update his personal information, also he can consult his paths.

* **Claims**

The patient can send claims about a doctor.

* **Search Doctors**

Visitors, patients and doctors, can search for doctors in our application.

* **Contact Support**

Visitors, patients and doctors, can contact support.

* **Manage Patient Path**

The doctor can create a patient path, and he can recommend other specialists for that patient.

* **Manage Doctor Profile**

The doctor can consult and add and update his personal information, also he can consult his dashboard and analytics.

* **Manage Accounts**

The administrator can consult, activate or deactivate patients and doctors accounts.

* **Manage Claims**

The administrator can consult the claims and send warnings to the doctors.

* 1. Actors

Indeed our application includes the following actors:

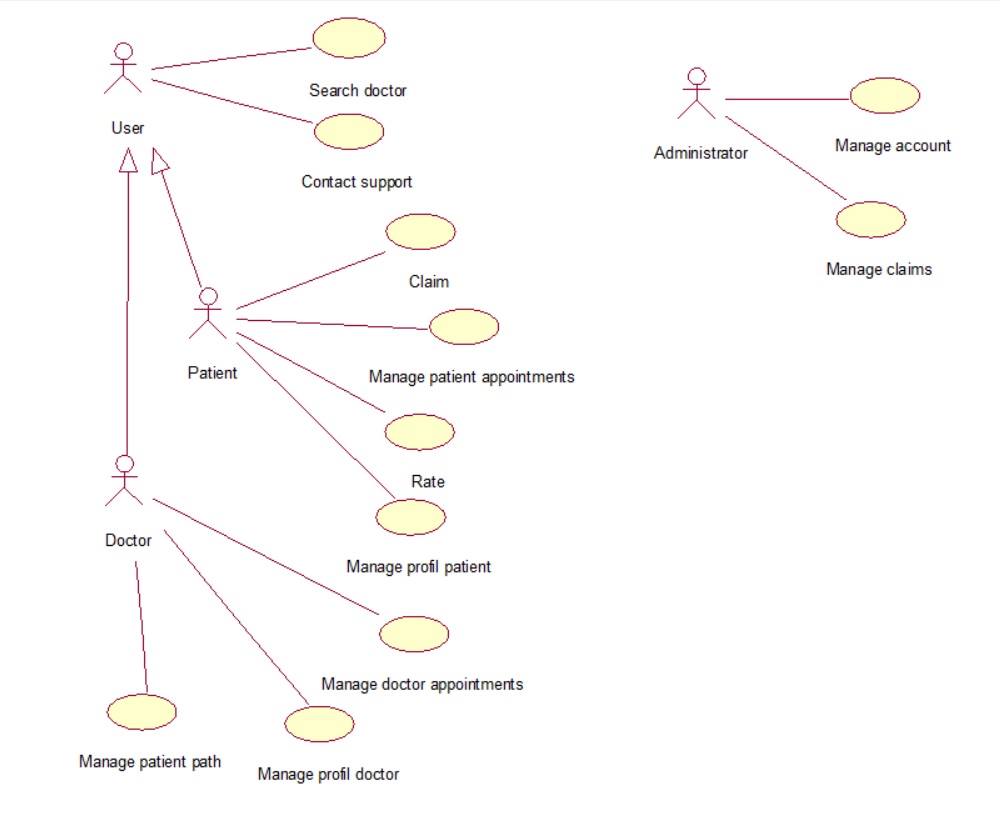
* **The simple user**  
  Who can only access services that do not require an authentication.
* **The administrator** It is the actor who manages the users of the system (activation / deactivation of accounts) and the different specialties and modules found in the application, visualize the statistics and has the right of access to the application.
* **Doctor**It's the user who has his own schedule to put his slots available to receive appointments and notification from the system, he is also the responsible for the patient's medical history.
* **Patient**   
  It is the user who makes appointments with professionals from freeway easy and immediate.
  1. General Use Case Diagram

Figure 4 : General Use case diagram

* 1. Analysis Class Diagram

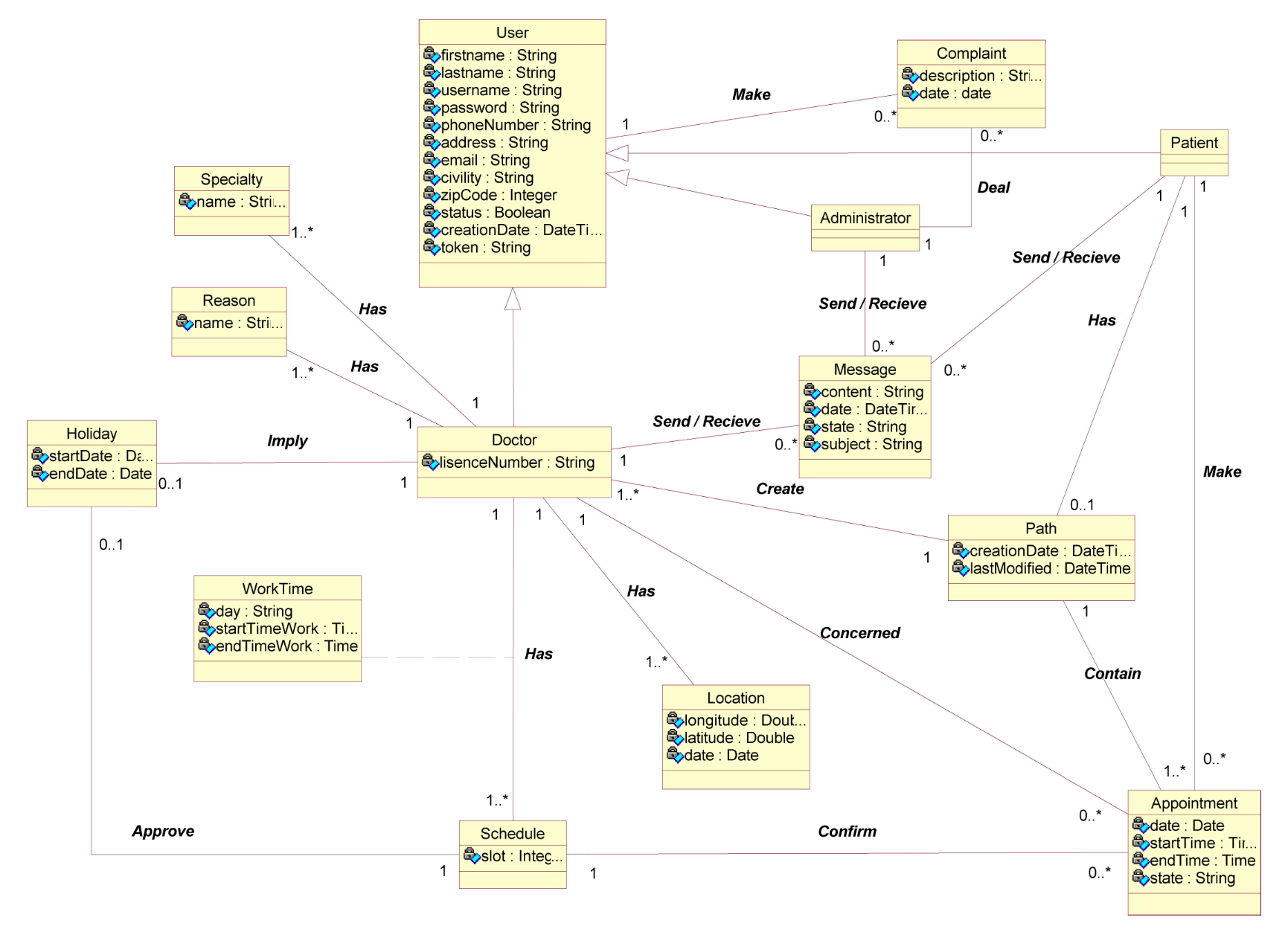


Figure 5 : Analysis class diagram

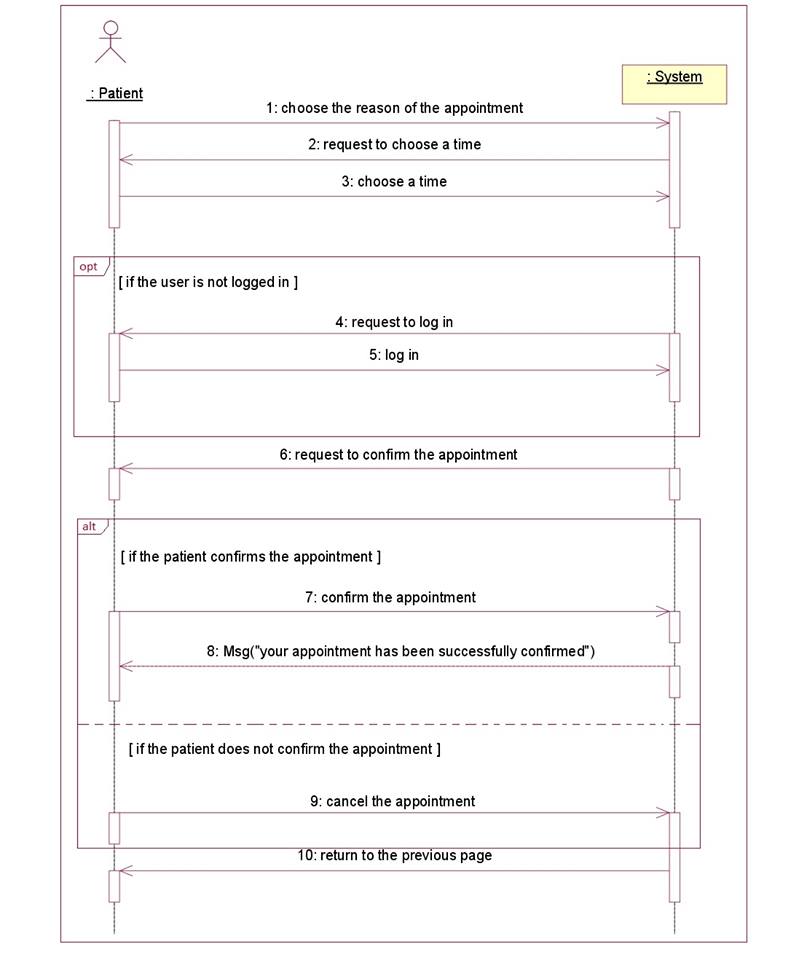
* 1. System sequence diagram

Figure 6 : Make appointment system sequence diagram

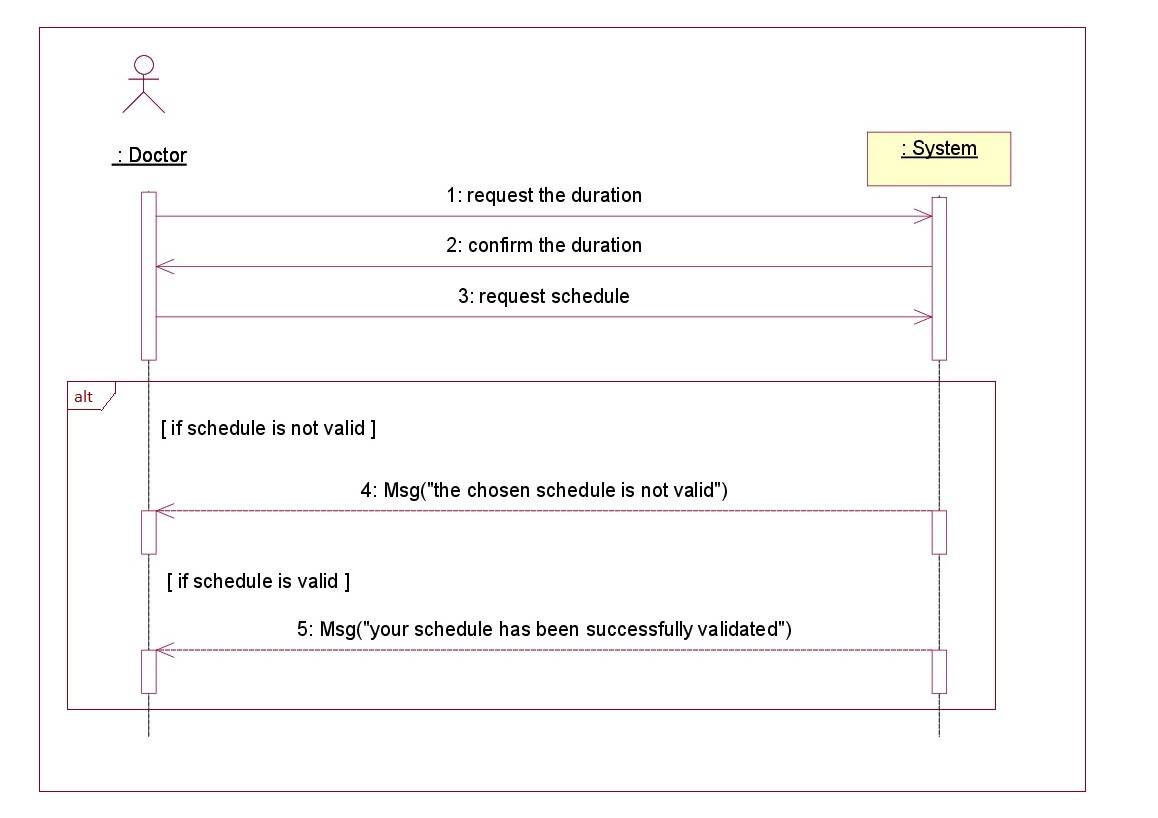


Figure 7 : Add schedule sequence diagram

* 1. Non-functional requirements

These are requirements that do not relate specifically to the behavior of the system but instead identify internal and external constraints of the system.

* **Simplicity**

The graphical interface of our application must be simple clear and easy to handle, to ensure easy and intuitive navigation and since users are not computer scientists.

* **Effectiveness**

The execution of the application must be done without errors and without crashes.

* **Performance**Attention should be paid to loading time, processing time and refreshment.
* **Availability**

The application must be available regardless of the information flows entered.

* **Reliability**

The application must work in a consistent manner without errors.

* **The validity**

It is the fact of responding to the functions expressed in the specifications.

* **Security**

The system in question must provide an authentication and access control function by specifying the access rights for each user to ensure the security and integrity of the data.

The application must respect the confidentiality of users’ data.

Conclusion

Throughout this chapter, we projected the purpose of the system to be developed, our application manages to save time by gathering all the existing solutions and providing several additional features all related to healthcare that satisfy most of the important needs. In the upcoming chapter, we start the development process, beginning with Java Enterprise Edition and the web service.

1. Physical architecture

Before the start of the project, we were informed about the technologies we will have to work with or use and according to these technologies a certain Agile Method will be used to better manage the work process.

* 1. Global architecture

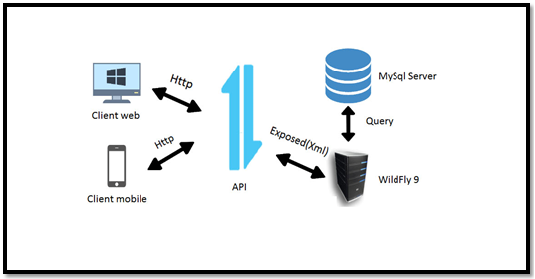


Figure 8 : Global architecture

* 1. Environment
     1. Technologies
  + **Java Enterprise Edition**

Java EE will be the back-end system of our application. Using Java EE to build all necessary functionalities regarding this phase, then exposing them as RESTful API that shall be consumed in an upcoming phases.

* + **.NET**

The .Net side of the application will be the Front-end product for the back-office of the main application as it will consume the exposed Metadata from the Wildfly server.

* + **Angular/Ionic**

The Ionic side of the application will be the Front-end product for the front office for mobile users and the Angular side of the application will be available for web users and both will consume the exposed Metadata from the Wildfly server.

* + 1. Web service

A client application can access remote distributed resources. There are several ways to access these resources, and web services are the most portable. We will use REST services (Representational State Transfer) with a Java API, which has an extraordinary evolution in the last versions of the Java Enterprise platform.

* 1. Deployment Diagram

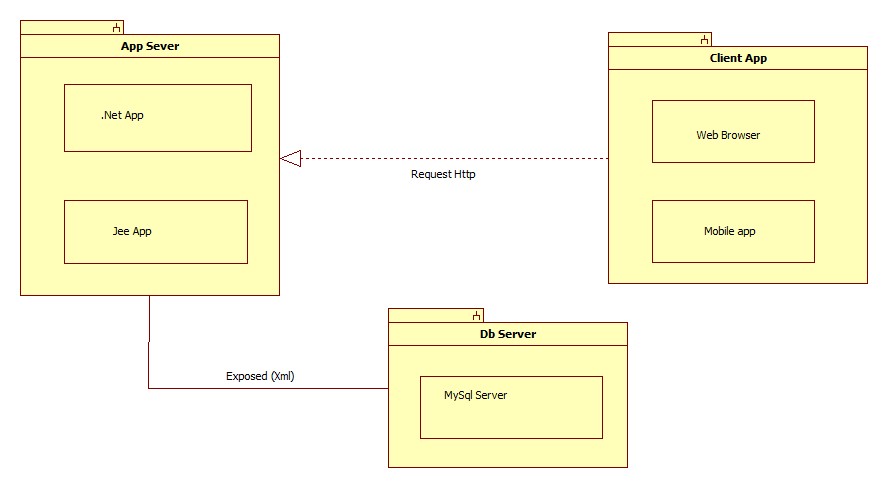


Figure 9 : Deployment diagram

Conclusion

This diagram resumes the global architecture of our application. During the phase 1, we will develop the Java EE resources, deploying in the WildFly server that shall be exposed via a REST API. In the phase after, we will tackle the ASP.NET application, deployed on IIS express and provides the back-office system. In the final phase , we will consume the previously exposed REST resources with Angular as a Front-end application along with an Ionic mobile application.

1. Project Management

Introduction

Since our project falls in the software criteria, we chose to follow a method of work that has proven to be highly successful in similar domains which is the Rational Unified Process used for software development to produce a functional product.

* 1. Software Development Process

RUP is a software development process from Rational, a division of IBM. It divides the development process into four distinct phases that each involves business modeling, analysis and design, implementation, testing, and deployment. The four phases are:

* **Inception** - The idea for the project is stated. The development team determines if the project is worth pursuing and what resources will be needed.
* **Elaboration** - The project's architecture and required resources are further evaluated. Developers consider possible applications of the software and costs associated with the development.
* **Construction** - The project is developed and completed. The software is designed, written, and tested.
* **Transition** - The software is released to the public. Final adjustments or updates are made based on feedback from end users.

Also it’s structured around six fundamental best practices, which are so-named due to their common use throughout the industry:

* **Develop Software Iteratively**: Encourages iterative development by locating and working on the high-risk elements within every phase of the software development life cycle.
* **Manage Requirements**: Describes how to organize and keep track of functionality requirements, documentation, tradeoffs and decisions, and business requirements.
* **Use Component-Based Architectures**: Emphasizes development that focuses on software components which are reusable through this project and, most importantly, within future projects.
* **Visually Model Software**: Based on the Unified Modeling Language (UML), the Rational Unified Process provides the means to visually model software, including the components and their relationships with one another.
* **Verify Software Quality**: Assists with design, implementation, and evaluation of all manner of tests throughout the software development life cycle.
* **Control Changes to Software**: Describes how to track and manage all forms of change that will inevitably occur throughout development, in order to produce successful iterations from one build to the next.

The RUP development methodology provides a structured way for companies to envision create software programs. Since it provides a specific plan for each step of the development process, it helps prevent resources from being wasted and reduces unexpected development costs.

* 1. Version control repository Manager (GIT)

The aim of GIT is to manage web development projects and files. As they change over time. Such a git repository contains a set of commit objects and a set of references to commit objects. A git repository is a central place where developers store, share, test and collaborate on web projects. We will be using GitLab over GitHub for the simple reason, which is Issue Tracking.

If you are using GitHub issues, you might think that it lacks some functionality. GitLab provides a powerful Issue Tracker that lets you change status and assignee for multiple issues at the same time. The repository is owned by Git Master “MEGDICHE Hamdi” as the rest of the group are assigned Developers.

Conclusion

The global view of the architecture shown above will help the team during the phases of designing and implementing methods and functions giving them a better understanding of the logical architecture of each of the applications especially when having to deal with RestAPI.

1. Mockups

The Mockups below resume the process to make an appointment. We opt for “Balsamiq mockup“ software for interface design.

This is the home page of our application, the visitor can easily search for a doctor by his name, specialty or address.

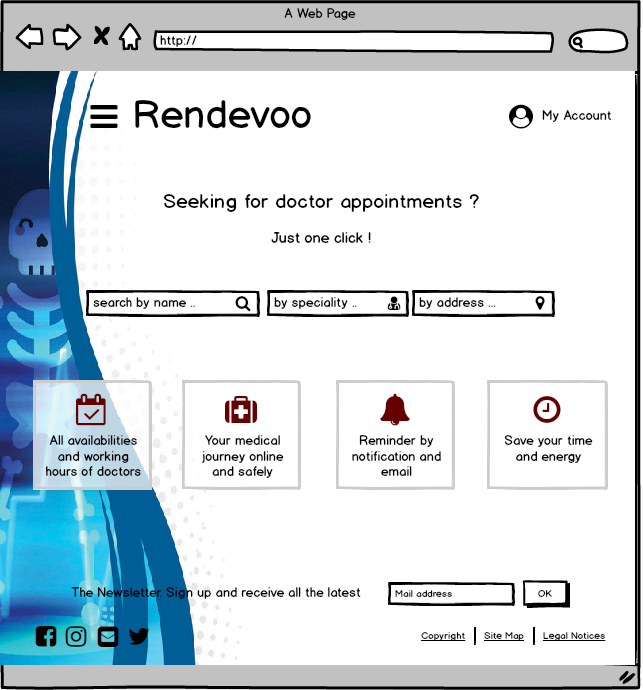


Figure 10 : Home page

This page displays the result of the search in a map and a list and allows the customer to choose a doctor to make an appointment.

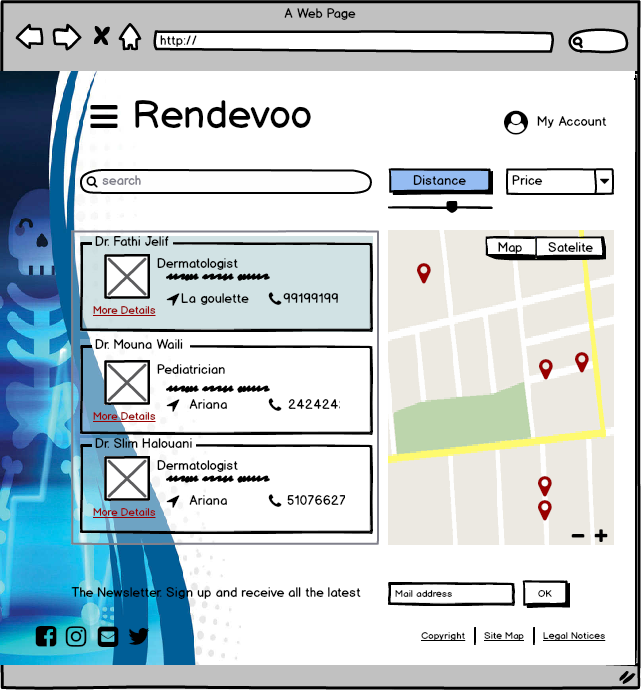


Figure 11 : Search Result Page

In this case, the visitor must register or sign in as a patient to be able to make one.

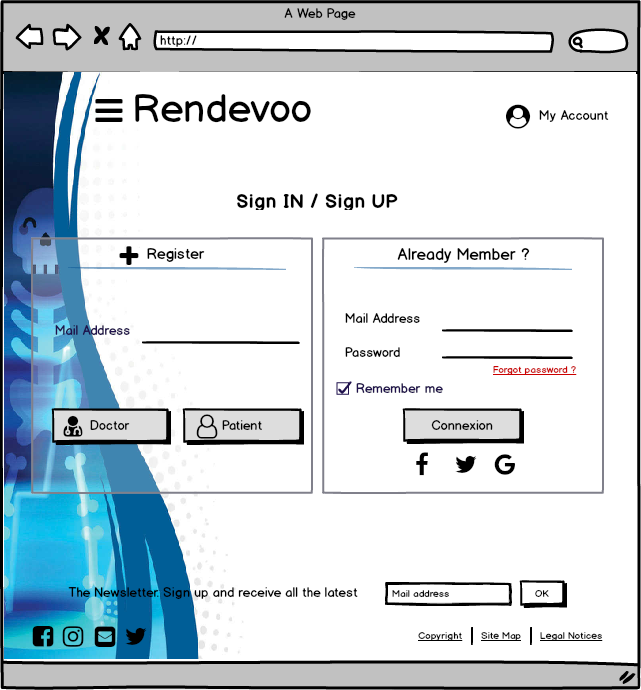


Figure 12 : Sign IN / Sign UP interface

This page allows the patient “Ahmed Hamden” to view doctor “Fathi Jelif”’s availability and choose a free slot to book the appointment online with him.

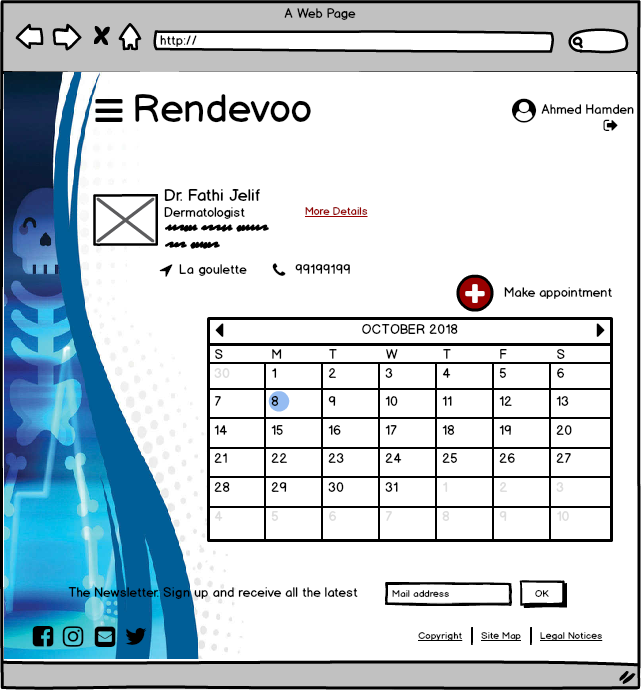


Figure 13 : Appointment booking page

Finally, the patient “Ahmed Hamden” can display his profile page to check the appointment that he made early, also he can manages his account.

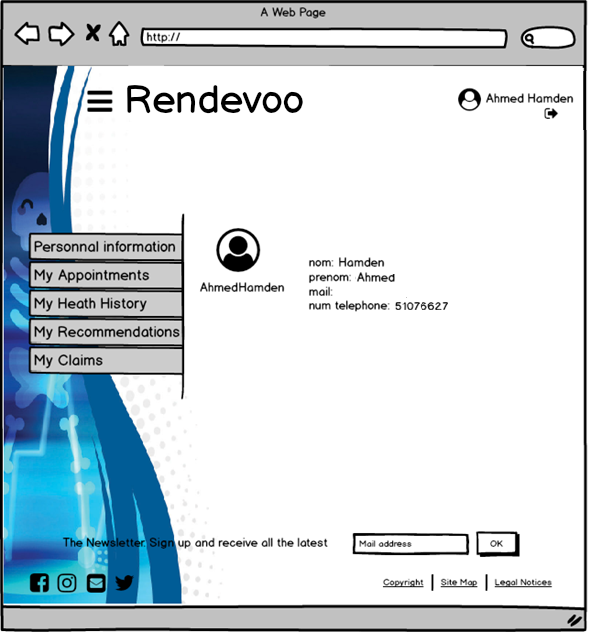


Figure 14 : Profile page

General conclusion

Throughout our project, we will develop an integrated platform dedicated to a wide community willing to help patients to make appointments the best way we can with whatever we can spare

Our mission is to develop an integrated platform (web, desktop and mobile) throughout this semester. Our project will be divided into 3 phase:

* Phase 1: Consists of developing a Java EE application to expose all of our functionalities as well as our web services, all of which will be used during the following two phases.
* Phase 2: Through which we will be using .Net framework to implement the back-office features for administrators to manage and supervise all of the application events that require special attention and/or authorization.
* Phase 3: Finally, we will be implementing with Angular and Ionic a client-side mobile application.

By the end of the semester, we will have put all of our knowledge together and we would have a totally integrated and fully functional platform. Together, we can make a difference. Together, we can make this world a better place.

CHAPTER2: (Phase 1: JavaEE / REST API)

Introduction

During this phase, we developed the back-end system of our application, using Java EE to implement all of the core functionalities. We then exposed them as a RESTful API (resources) that shall be consumed in the upcoming phases. In this chapter, we will detail the process and steps of the realization of this phase.

1. Phase 1 Overview
   1. Introduction

During this phase ,we aim to develop the back-end system of our application. Using Java Enterprise Edition 6 to build all necessary functionalities regarding this phase, then exposing them as RESTful API that shall be consumed in an upcoming phase. In this chapter, we will detail the process of realization during this phase.

* 1. Approach

There was tremendous features that should be dealt with, from adding users to making appointments , so after a big deal of time minimizing all of the work to make it easier to produce:

**1st features** were the “Offer/Service & Requests” system such as the accounts management, data extraction from Doctolib , appointment (add / smart adjustment ) and the patient path management.

All of these modules are to some degree similar instructure, design and logic as most of their scenarios represent posts waiting for a right answer with slight difference. They also represent better the theme of the whole project as they affect the patients directly so they hold the first priority among other features.

**2nd** are about chat , analytics , search , claims and rate.

Each of these are independent modules but for the most part they work mostly on dealing with providing distraction, collaborations and volunteer work among the Users of the application so they held second priority.

**3rd** were the Must-have functionalities of the application to be optimal and understood to be a spine for the other features such as User Identification and Appointment Management. We chose to work on the Modules stated as third and first with respective priority during this phase.

* 1. Workflow
     1. Version Control

Our Git repository contains a set of commit objects and a set of references to commit objects. As we said before we still use GitLab for the simple reason which is Issue Tracking. And we chose the Git Kraken Repository Manager to manage the project remotely. The repository is created by Git Master Hamdi Megdiche as the rest of the group are assigned Developers with master privileges. Currently 48 commits have been made by the 5 collaborators.

* + 1. Overall Process

We went through designing our Managed Entities focusing on the User, then we started to structure our Project as will be detailed in the next chapter. Once we had our project and Design we created our packages and assigned Tasks to each of the members. Once all is set we cloned the project and each worked separately until every review each end of week.

* 1. Conclusion

The workflow was smooth for all members as they aimed to do a good job varying from exposing to using bean sessions and even scheduled tasks on the server side to better control the application. All of these will be detailed each and separately on the chapters below.

1. Project structure

Introduction

During our project, we have implemented a Java EE solution that exposes a web service while respecting the structure of our project which consists of three parts:

* epione-web
* epione -ear
* epione -ejb

Adding the JPA specification that allowed us to communicate with the database through the hibernate ORM and the JAX-B specification to expose the web service.

* 1. Project Structure
     1. EJB Project

An EJB module is used to assemble one or more enterprise beans into a single deployable unit. It is deployed in a standard JavaTM archive (JAR) file. An EJB module can be used as a standalone application, or it can be combined with other modules to create a Java EE enterprise application. An EJB module is installed and run in an enterprise bean container. An EJB project must be referenced by an enterprise application project (defined as a module in an EAR) in order to be deployed successfully and run on a server.

* + 1. WAR Web Project

Dynamic Web projects are always embedded in Enterprise Application projects. The wizard that you use to create a dynamic Web project will also create an Enterprise Application (EAR) project if it does not already exist. The wizard will also update the application.xml deployment descriptor of the specified Enterprise Application project to define the Web project as a module element. If you are importing a WAR file rather than creating a dynamic Web project new, the WAR Import wizard requires that you specify a Web project, which already requires an EAR project.

* + 1. EAR Archive Project

Enterprise application projects contain references to the resources needed for enterprise applications and can contain a combination of Web modules, JAR files, connector modules, EJB modules, and application client modules. An enterprise application project is deployed in the form of an EAR file, and is therefore sometimes referred to as an EAR project. The modules in an enterprise application project are mapped to other J2EE projects. The mapping information is stored in metadata files within the enterprise application project. The metadata files are used for exporting the project to an EAR file and for running the project on the server.

* 1. Specifications
     1. JPA

The Java Persistence API (JPA) is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational database. JPA was defined as part of the EJB 3.0 specification as a replacement for the EJB 2 CMP Entity Beans specification. JPA is now considered the standard industry approach for Object to Relational Mapping (ORM) in the Java Industry.

* + 1. JAX-RS

JSON and Java technology are natural partners in helping developers exchange data and programs across the Internet. This partnership is particularly important for Web services, which promise users and application developers program functionality on demand from anywhere to anywhere on the Web. JSON and Java technology are recognized as ideal building blocks for developing Web services and applications that access Web services.

* 1. Conclusion

The structure is archived and detailed in the ear project as it has 2 containers one for the application (EJB) and one for the Web project which will be our Web services dealer and recipient.

1. Design

Introduction

During this chapter we will focus mostly on the design approach we viewed starting the print taking to mind the limits of the application and each of the architecture’s layers.

* 1. Class Diagram
     1. Persistence package

This package includes all the JPA entities in our application, including the Embeddable classes and the needed mapping relations.

* + 1. Interface package

In the package above, we added our Local and Remote interfaces, in which we set our main methods that shall be implemented in our service layer.

* + 1. Business package

Included above are all our services and business logic, these classes implement the interfaces set previously. Using the Entity Manager to perform all the needed transactions.

* + 1. Resources package

In the Web container, we developed our REST services using the interfaces developed in the EJB container part and tested them throughout a rest client.

* 1. Object Sequence Diagram

This sequence diagram describes the process of creating an adoption offer through which organizations will be able to enlist pets for adoption.

This sequence diagram describes the process of creating an adoption request through which members will be able to request pet adoptions.

* 1. Strategies and Associations

Inheritance is one of the key concepts in Java, and it’s used in most domain models. That often becomes an issue, if you try to map these models to a relational database. SQL doesn’t support this kind of relationship and Hibernate, or any other JPA implementation has to map it to a supported concept. We used Joined Columns Strategies for the reports to make queries more accessible and make the parent a Managed Entity on its own and to ensure data integrity, as for User we worked with the single table strategy to limit the access and make a discriminator column to differentiate each role.

* 1. Conclusion

The Design process helped the team get on the right course for the start of the implementation especially having a better understanding of what is exactly asked, as is said previously taking to mind the limits of the technology and the architecture.

1. EE Implementation

Introduction

The side of the EE container: Manages the execution of EJB run on the Java EE server. It

also takes care of transactions, security, and data sources (JPA resources) and other specifications.

* 1. Package

The Packages were divided to represent different logic layers as in the persistence and services packages represent the business layer. Along with an interface package composed of various local and remote interfaces to be implemented in the services. We also added a Schedule package on which our methods will be implemented and are executing as Singleton Instance during the life cycle of the Server Using Other EJB beans called upon in this package’s classes and methods making advantage of this specifications that EJB provides.

* 1. Conclusion

All of these beans will provide services for the Web container to work on and build web services

giving it a better understanding of the business layer.

1. Web Services

Introduction

Our client application can now access remote distributed resources. There are several ways

to access these resources, and web services are the most portable. We used REST services with a Java API JAX-RS which has an extraordinary evolution in the last versions of the Java Enterprise platform.

* 1. Nomenclature

The paths provide for the web service won’t be accessible for user clients but Front-end developers should work with them so we aimed to make the paths recognizable as each resource will start with the class path bearing the prefix as the name of an entity with no uppercase such as “Fundraiser” will have a Resource class path “/fundraiser/...” and so on followed by a verb with a significant meaning such as “/fundraiser/addfundraiser” is for adding a fundraiser.

* 1. Conclusion

The global view of the resources shown above will help the team during the phases of

consuming these web services giving them a better understanding of the nomenclature when they swap modules during the coming phase s especially when having to deal with RestAPI.

1. Server-Side Control
   1. Introduction

We needed community control and as we had to ease our clients’ needs we needed to maintain the servers whenever the admin was on vacation, so we thought of having many scheduled tasks for the server to serve without admin intervention.

Conclusion

The Scheduled tasks made more sense to the notion of a back-end system as it provided More flexibility on the server side of the application and let the process be as fluid as possible and made the business aspect give a proper good behavior for the application’s server side.

1. General Conclusion

During this phase, we split the features relevance-wise equally among the team members. We used the UP methodology to organize ourselves and GitLab as a version control tool to work together distantly. We ended up conceiving our application as an array of resources that allows us to respond to the needs established in the first phase .These resources, exposed as RESTful services, will be core parts of our future applications. In the upcoming phase, we will tackle the ASP.NET application build that shall have extra as well as advanced features and have the back-office services.