**General Introduction**

In the context of our integration and development project of the 4th year web and internet technology (TWIN), we are assigned the realization of a project in which we directly apply the skills acquired throughout this semester. We chose Animal caring project, which is a platform that offers animal-related services.

Animals are a wonderful part of our lives.  They bring us joy.  They make us laugh.  They

give us unconditional love.  They bring us sorrow when they leave us, but leave us with cherished memories of true friendship. But, many animals in our society are suffering.  They are homeless and unwanted, abused and neglected.  Some go hungry on our streets; others put to death in our shelters; and many suffer silently at the hands of their owners, who either don’t understand, or simply don’t care.

As the old saying goes, “it takes a village to raise a child”, and we believe it takes a community to make the world a better and safer place for animals to live.

Having a pet can be a stressful and exhausting experience. No matter how trying your pet is, you will always love them. Our platform will provide many services to facilitate animal’s life.

“Animal caring” is a project undertaken as a mandatory requirement for the course “**PI-dev 4TWIN**” that is being conducted mutually by AYMEN Makhlouf, KHALED Ouertani, NACIRA Suidak, OUMAYMA Gader and SEJIR Ben Ali all of whom are members of 4TWIN2 class of 2017/2018 and the SEAHAWKS 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 4 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. -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 lines, 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: (Sprint 0: Main conception & Used methodology)

1. Introduction

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

The theme was proposed by the tutors as a third option after the team lost two drafts to pick other interesting themes and were tasked to expand the main feature of the application which can be resumed in facilitating “The animal caring” to pet owners primarily. 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
2. Issue

Today, we're living in a stressful world. Every minute counts and we're always busy around the clock. We tend to forget to value the important things. And one of those things is animals. They're a wonderful part of our lives. They bring us joy, make us laugh, give us unconditional love and bring us sorrow when they leave us, but do so with cherished memories of true friendship. But many animals in our society are suffering. They're either homeless or unwanted, hungry wandering the streets, put to death in shelters or suffer silently because of their owners, who either don't understand them or simply don't care. It takes doctors to save one life, several teams to construct a building, years to build a city and we believe it takes a community to make the world a better and safer place for our beloved friends.

That's where we come in play. We can improve the quality of life of animals, help strengthen the bond between them and their owners, end their suffering and eliminate the need for euthanasia of healthy, adoptable animals in our shelters just because they have no room.

With our project, we strive to address the suffering of animals and the epidemic of homelessness and euthanasia.

Together, we can make a difference in the lives of animals.

1. Target audience

The main actors as we will see in detail later would be the pet owners and animal care organizations. The services will range from reporting disturbing events regarding animals to providing paid and pre-paid services between users. Animal contests will be available for users to organize, host and enroll their pets in. 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

Depending on the main features of the application, we found many web platforms and animal-care service providers across the web such as Youcaring.com specialized in crowdfunding, fundraisers and donations by compassionate users.

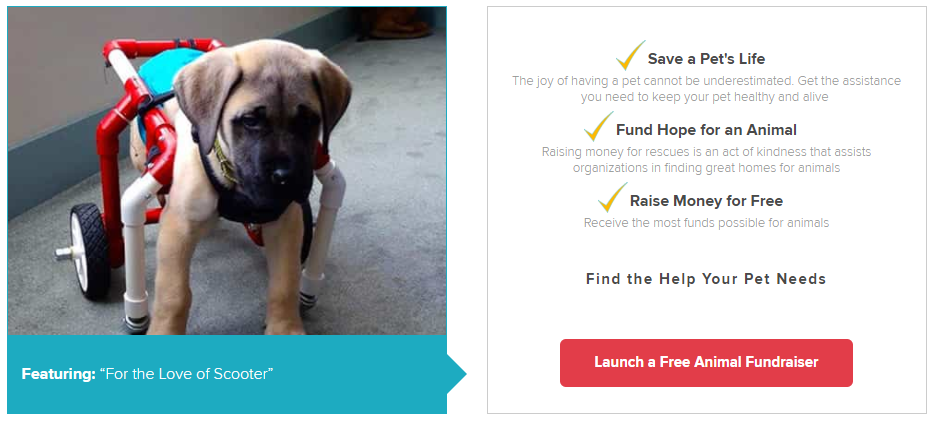


Figure 1 Launching a Fundraiser on Youcaring.com

Other similar features have been found in Petfinder.com such as organized adoption and Fostering.

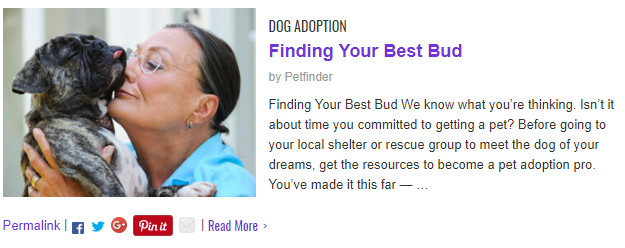


Figure 2 Dog adoption section on Petfinder.com

1. Analysis
2. Introduction

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

1. Functional requirements

***Adoption***

On one hand, organizations can place an eligible pet for adoption. On the other hand, users can request to adopt a pet from an organization’s shelter. Organizations can confirm or decline a pet adoption request. Once the adoption is confirmed, the user can pick up his pet.

***Lost & found***

Countless pets get lost on a daily basis, which is why our platform will provide a section dedicated to lost pets to help legitimate owners find their beloved ones. Therefore, either people enlist a found pet as lost or the rightful owner does.

***Fundraisers***

One of the crucial needs that our application will cover is the fundraisers aiming to help animals as end goal. Starting fundraisers will be quick and easy.

***Pet management***

Since our platform will be providing many animal-related services, we made sure to allow users and owners to manage their own pets.

***Pet sitting***

Another important need that our platform has to cover is fostering. Whether people find themselves unable to take their pet wherever they are going or not able to take care of it for whatever reason for a bit of time and need someone to look over their beloved animal, fostering comes in play and allows pet housing for a specific period.

***Donations***

Many animal lovers want to further assist with animal care and are willing to donate. The application will allow them to donate either money, food or equipment.

***Breeding***

For those who want to breed their animals for the sake of general good or to put the newborn for adoption or want to help others find a partner for their pet to be able to have their own puppy for instance, the platform will provide a breeding feature.

***Events***

This application will allow users to organize animal-related events to which the community will be able to participate.

***Accounts***

An obvious feature that our platform is to provide is account management as in registration, signing in, editing personal information and so forth.

***Stories***

Our animal caring application will gather a sheer number of animal lovers which means wide

communities will use it. Thus, implementing a stories tab for them to share their experiences,

stories, wishes and so forth is required.

1. Actors

### Simple Users

The simple user represents pet owners, farm owners and even zoos or anyone having a pet and wishing to find the tools and services necessary for him to take good care of his pets. This group of users contain also any user that does not own a furry friend and all share the same features such as Adopting but are mostly consumers.

### Advanced Users

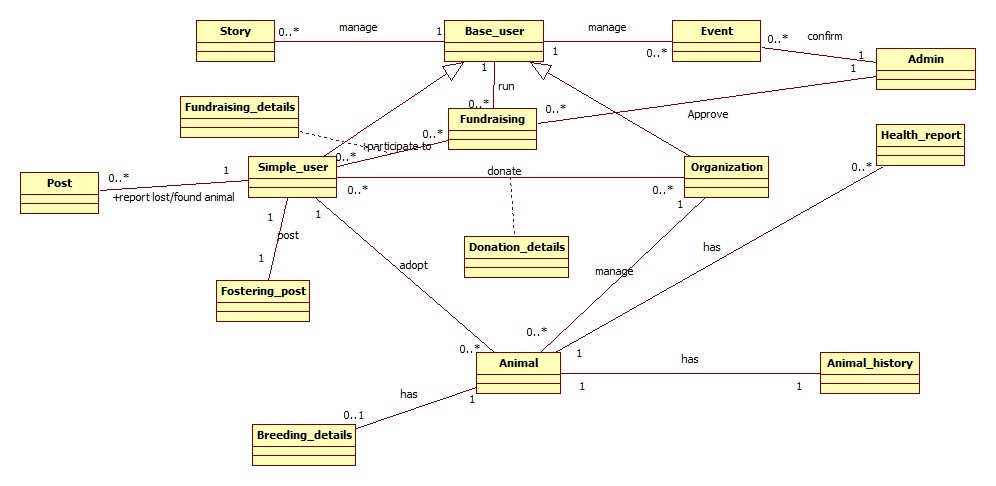
The advanced user either represents an organization that provide better and trusted services such as Adoptions and hosting big Fundraisers or veterinaries, the latter group will become the expert user giving Top posts and expert’s tag to identify them with ease.

### Administrator

The Administrator will have the key to manage false claims of pet ownership, control disputes, manage user groups and verify some organizations credibility. It is also expected from the administrator to moderate the story section as well as verify the Fundraisers if needed.

D. Use Case Diagram

E. Analysis Class Diagram



F. Non-functional requirements

Our application must assert the following non-functional requirements:

* Performance
* Security
* Maintainability
* Robustness
* Adaptability

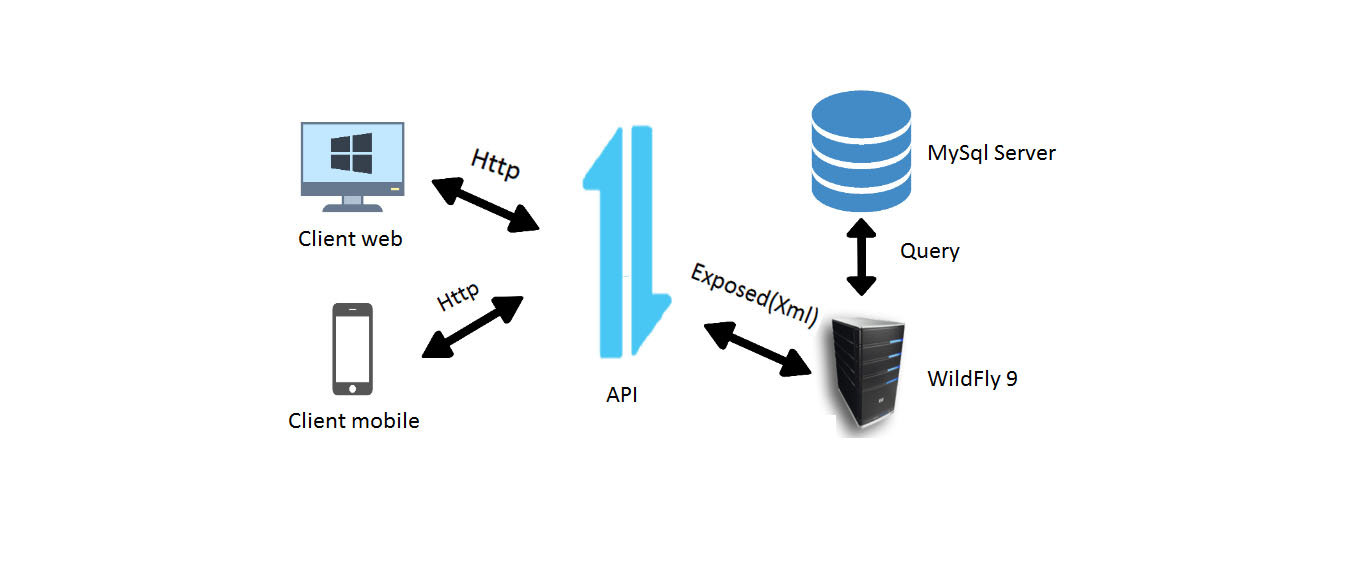
G. Conclusion

Throughout this project, we hope to partake in the animal caring cause and assist animal lovers with services and tools to better take care of animals and help them find their owners and homes. Our application manages to save time by gathering all the existing solutions and providing several additional features all related to animal caring that satisfy most of their important needs. In the upcoming chapter, we start the development process, beginning with Java Enterprise Edition and the web service.

1. Physical architecture
2. Introduction

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. Physical Architecture



C. 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 sprint, then exposing them as RESTful API that shall be consumed in an upcoming sprints.

* ***.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 Angular2 side of the app 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.

D. Deployment Diagram

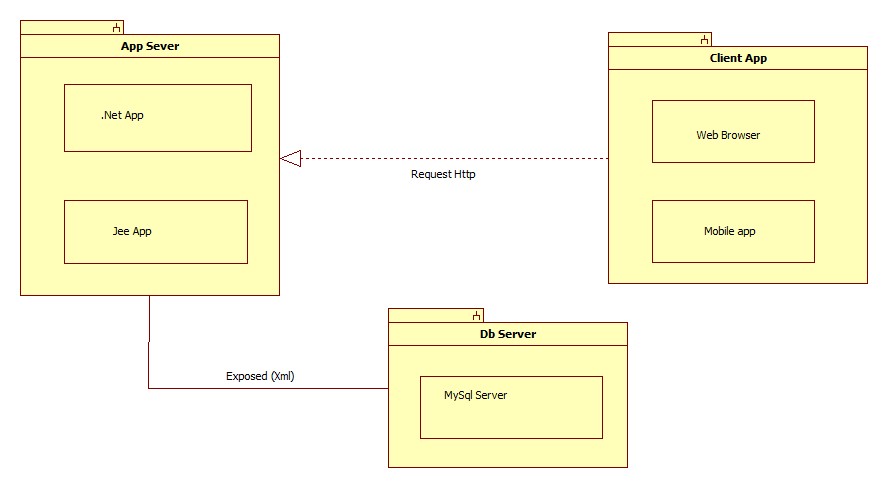


Figure 3 Initial view of the Deployment Diagram

E. Conclusion

This diagram resumes the global architecture of our application. During the sprint 1, we will develop the Java EE resources, deploying in the WildFly server, that shall be exposed via a REST API. In the sprint after, we will tackle the ASP.NET application, deployed on IIS express and provides the back-office system. In the final sprint, we will consume the previously exposed REST resources with Angular 2 as a Front-end application along with an Ionic mobile application.

1. Project Management

A. 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 scrum methodology. Scrum is an agile project management concept used for software development that revolves around the use of multiple small teams working in an intensive, interdependent manner to produce a functional product.

B. Methodology

Scrum prescribes four formal events for inspection and adaptation:

* Sprint Planning
* Daily Scrum
* Sprint Review
* Sprint Retrospective

The scrum team consists of:

* Product owner
* Scrum master
* The Development Team

Throughout this project, we will exchange roles during each sprint to guarantee our aptitude for managing a team and committing to an agile methodology.

C. Scrum

### Introduction

The sprints planning was pre-assigned by the tutors of the course depending on the use of certain technologies as we will see below.

### Sprints

|  |  |  |
| --- | --- | --- |
| SPRINTS | TECHNOLOGIES | NATURE |
| SPRINT 0 | DATA Gathering & Analysis | |
| SPRINT 1 | Java EE | Back-end |
| SPRINT 2 | .NET | Front-end |
| SPRINT 3 | ANGULAR2/IONIC | Front-end |

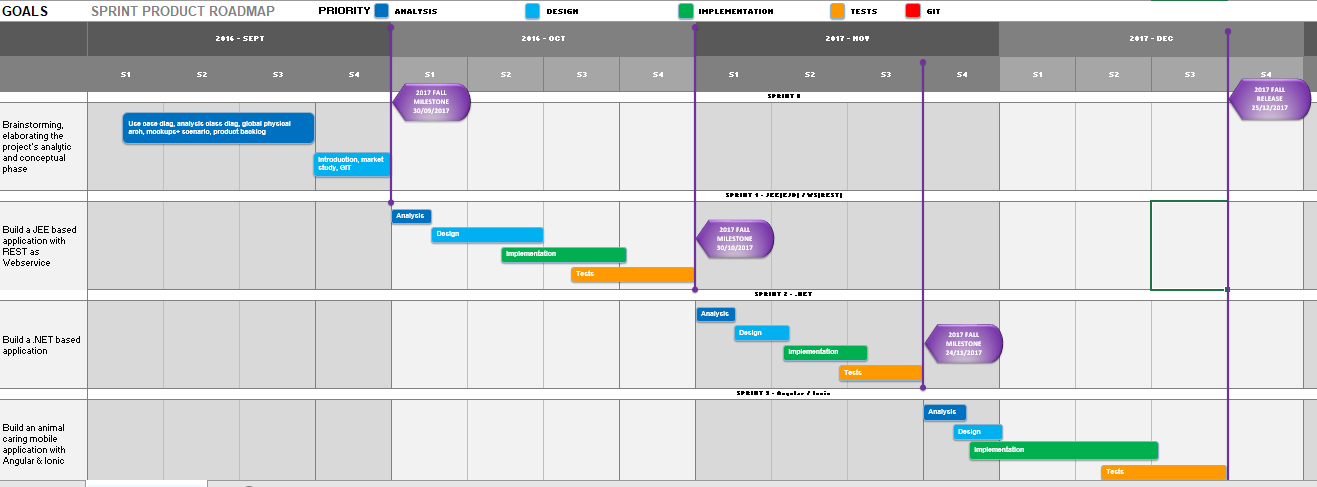
### Product Backlog

The agile product backlog in Scrum is a prioritized features list, containing short descriptions of all functionality desired in the product. Typically, a Scrum team and its product owner begin by writing down everything they can think of for agile backlog prioritization. This agile product backlog is almost always more than enough for a first sprint. The Scrum product backlog is then allowed to grow and change as more is learned about the product and its customers. Refer to the annex below to find our up-to-date Product Backlog.

### Conclusion

When practicing Scrum, we can make the sprint backlogs visible by putting it on a Scrum task board. We will update the task board continuously throughout every sprint. To make it more practical for us, using tools such as Trello.

1. Road Map



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’re 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 will be owned by Git Master Nacira Suidak as the rest of the group are assigned Developers.

1. 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.

# Mockups

The Mockups below resume the process of an adoption, in this case the Organization “Peta India “is managing its online shelter.

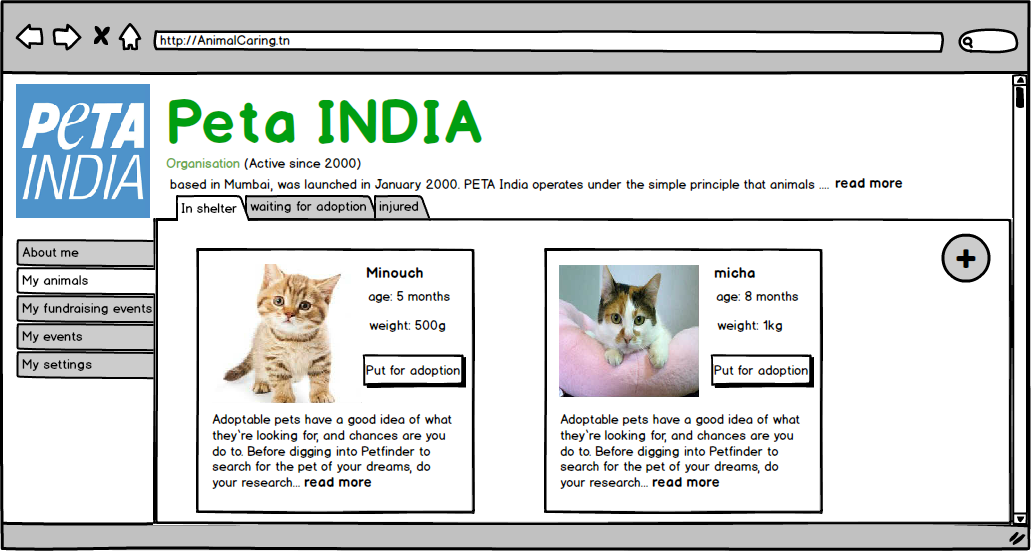


Figure 4 Managing shelter

The organization just put “Minouch” for adoption and “Foulena ben Foulena” sent a request to adopt him.

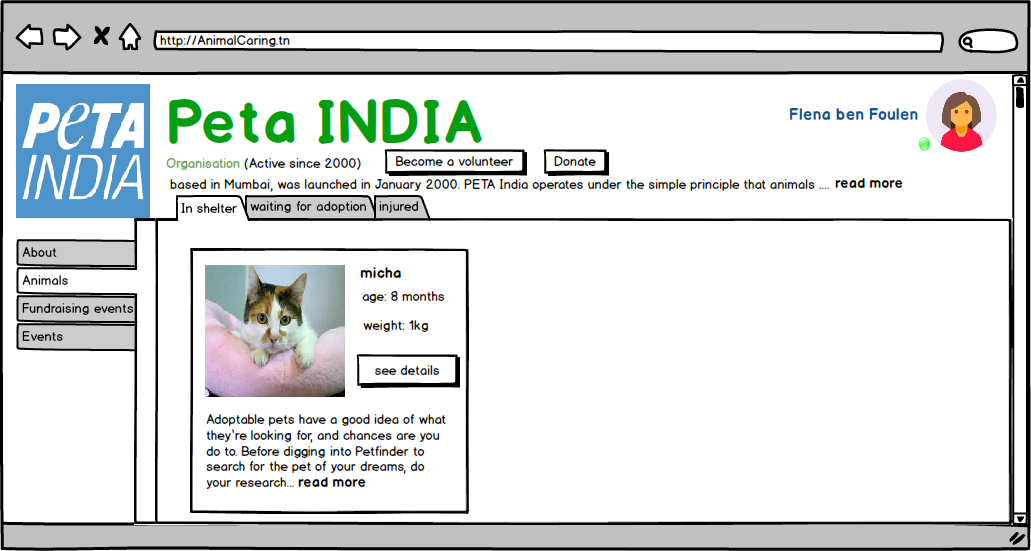


Figure 5 Adoption request

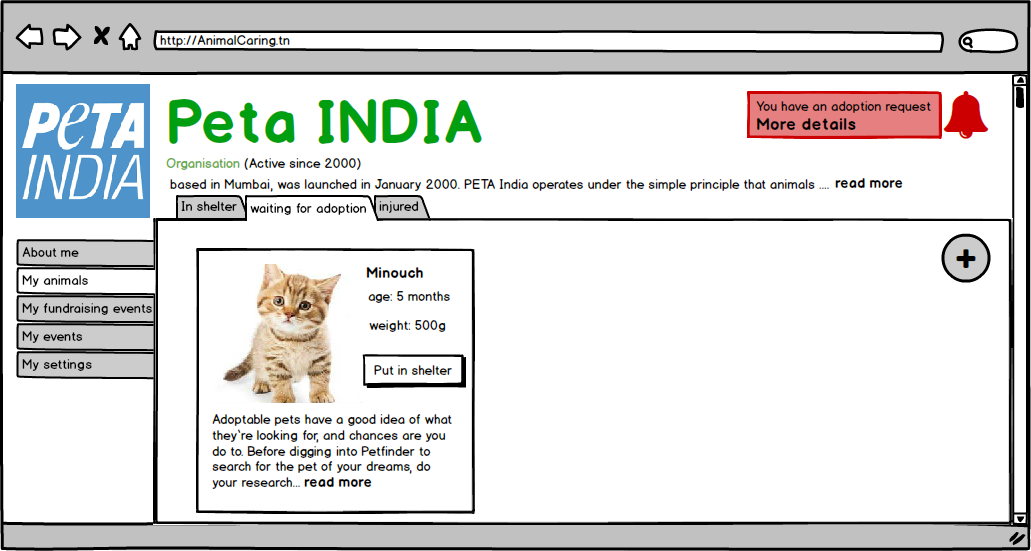


Figure 6 Notification for adoption

“Peta India” now has to confirm one of these requests she is receiving to finish the adoption process.



Figure 7 Confirm and end the adoption process

# Conclusion

Throughout our project, we will develop an integrated platform dedicated to a wide community willing to help animals in need the best way we can with whatever we can spare. Animals which either have owners or in shelters. We hope to allow this community to do so by providing several services and tools, thus they will be able to provide protection and improve the quality of life of their beloved friends. Our mission is to develop an integrated platform (web, desktop and mobile) throughout this semester. Our project will be divided into 3 sprints:

* Sprint 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 sprints.
* Sprint 2: Through which we will be using dotNet 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.
* Sprint 3: Finally, we will be implementing with Angular 2.0 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 for animal lovers. Together, we can make a difference. Together, we can make this world a better place.

CHAPTER2: (Sprint 1: JavaEE / REST API)

1. Introduction

During this sprint, 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 sprints. In this chapter, we will detail the process and steps of the realization of this sprint.

II. Sprint 1 Overview

* 1. Introduction

During this sprint, we aim to develop the back-end system of our application. Using Java Enterprise Edition 6 to build all necessary functionalities regarding this sprint, then exposing them

as RESTful API that shall be consumed in an upcoming sprint. In this chapter, we will detail the process of realization during this sprint.

* 1. Approach

As was shown by the team, on the product backlog previously, there was tremendous features that should be dealt with, from declaring a lost animal to breeding and even donating, so after a big deal of time minimizing all of the work to make it easier to produce, the scrum master and the team agreed on 3 main approaches to work on during the three sprint to come:

1st features were the “Offer/Service & Requests” system such as the Breeding Offers, Fostering, Lost Reports, Found Reports, Adoption Offers. All of these modules are to some degree similar in

structure, 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

animals lives directly so they hold the first priority among other features.

2nd are “Events” ranging from a Collective walk, Healthcare special events, a Zoo a circus ... 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 Animal Management. We chose to work on the Modules stated as third and first with respective priority during this sprint.

* 1. Sprint Backlog

The sprint Backlog will be available at the end of this report as an Annex due to visibility issues.

* 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 Oumayma Gader as the rest of the group are assigned Developers with master privileges. Currently 63 commits have been made by the 5 collaborators.

2. Overall Process

We went through designing our Managed Entities focusing on the User and Animal Classes, 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.

III. Project structure

1. 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:

* seahawks-web
* seahawks-ear
* seahawks-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
2. 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
2. 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.

IV. Design

1. 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
2. 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.

V. EE Implementation

1. 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.

VI. Web Services

1. 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. Exposed Resources

The Exposed resources will be in the Annex at the end of this document due to visibility issues.

(check the annex please)

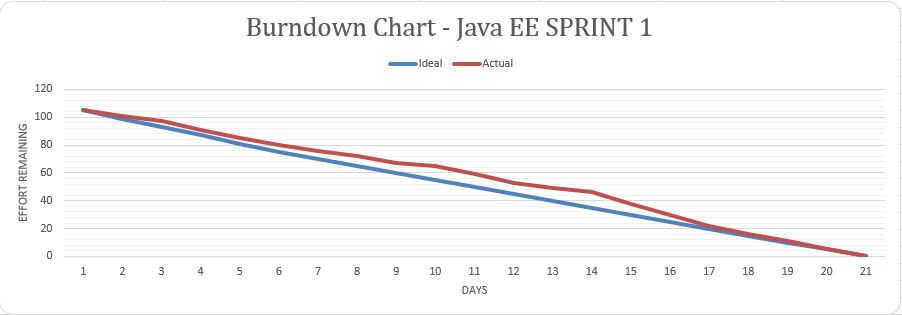
1. Conclusion

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 sprints especially when having to deal with RestAPI.

1. Burndown Chart



VIII. 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.

1. Scheduled Tasks

Implementing what EJB provides, we used the TimerService class to produce a Singleton Instance of the TimerService using the “@Scheduled () “annotation Interface to call methods depending on a timer or a trigger. Functionalities vary from finding the closest Organization to a found report situation to which no one replied, or no great answer was provided for 21 days, and assign this Organization to take over and find a solution, to finding the closes Free Localization for animal owners to breed their pets. The process is complex as it calls many EJB beans and depend on server stability.

1. 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 sprint, we split the features relevance-wise equally among the team members. We used the scrum 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 sprint. These resources, exposed as RESTful services, will be core parts of our future applications. In the upcoming sprint, we will tackle the ASP.NET application build that shall have extra as well as advanced features and have the back-office services.

CHAPTER3: (Sprint 2: ASP.NET /Consuming Web services)

1. Introduction

During the ASP.NET sprint, we developed a part of the front-end system of our application. We are consuming some resource from Java EE functionalities and implementing some others. The Model View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller as shown in the figure below.

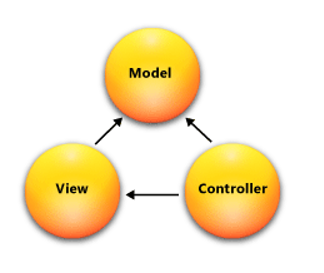


Figure 8 ASP.NET MVC

1. Sprint 2 Overview
2. Introduction

During this sprint, we aim to develop a part of the front-end system of our application. Using ASP.NET to build all necessary functionalities regarding this sprint. In this chapter, we will detail the process of realization during this sprint.

1. Approach

As was shown by the team, on the product backlog previously, there was tremendous

features that should be dealt with, from sharing pet stories, health care, walking, competitions and sitting so after a big deal of time minimizing all of the work to make it easier to produce, the scrum master and the team agreed on 3 main approaches to work on during the three upcoming sprints.

1. Sprint Backlog

The sprint Backlog will be available at the end of this report as an Annex due to visibility issues.

1. Workflow

As per usual, we used GIT in this sprint.

1. Conclusion

The workflow was smooth for all members as they aimed to do a good job. This will be further detailed in the upcoming chapter.

1. Project structure

Our solution consists of five mostly interlinked projects, that will be detailed in the upcoming section. It is important to note that we used the unit of work design pattern.

1. Data project

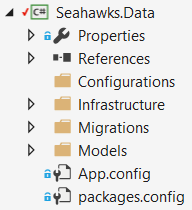


Figure 9 Data project structure

Contains the context, conventions, configurations, migrations history and infrastructure.

1. Domain project

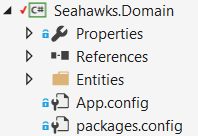


Figure 10 Domain project structure

Contains the entities. All the other projects refer to this project.

1. Pattern services project

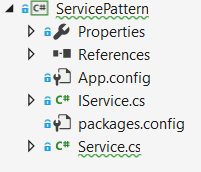


Figure 11 Service Pattern project structure

Contains the generic services like Add<T>, Delete<T> and so forth.

1. Services project



Figure 12 Service project structure

Contains the custom services that extend the generic services.

1. MVC project

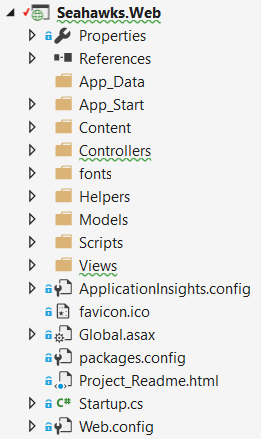


Figure 13 WEB project structure

Contains the views, models and controllers. Obviously, this is the presentation part of the ASP.NET project.

1. Entity framework



Figure 14 Entity framework

The Entity Framework is a set of technologies in ADO.NET that support the development of data-oriented software applications. This is the built-in Microsoft ORM.

1. Conclusion
2. Conception
3. Conceptual class diagram

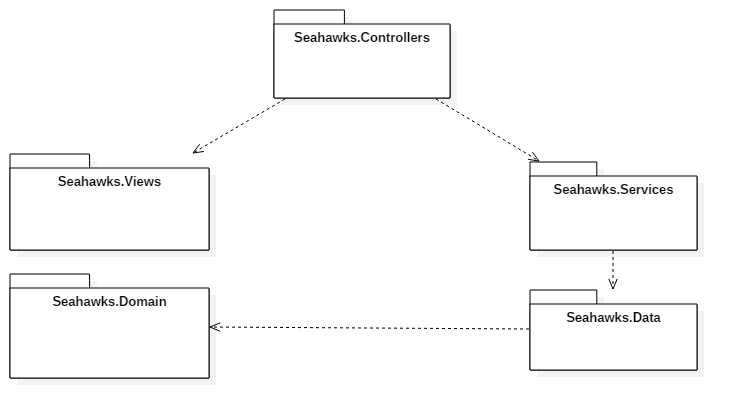


Figure 15 Global Architecture

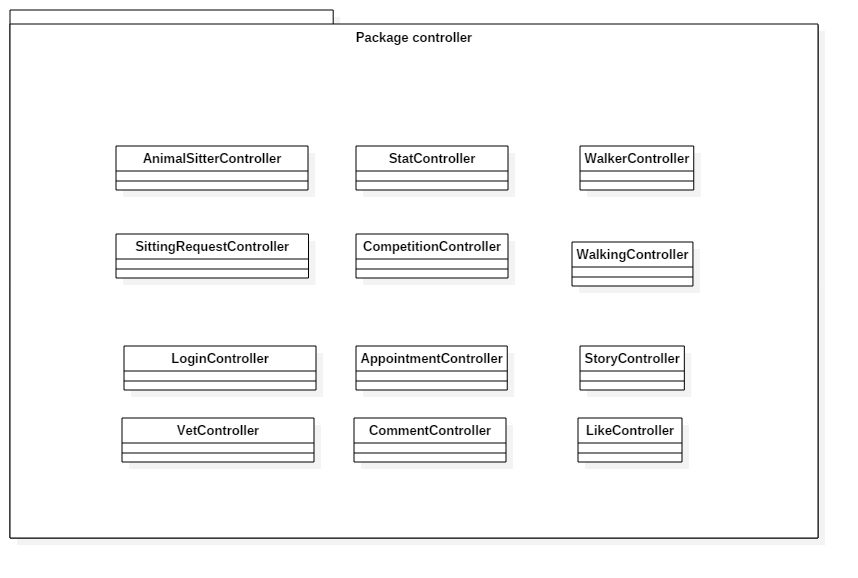


Figure 16 Application's controllers

This package contains the controllers we used in our application, which are responsible for responding to requests made against our ASP.NET MVC website. Each browser request is mapped to a particular controller. The next figure represents the view of each controller.

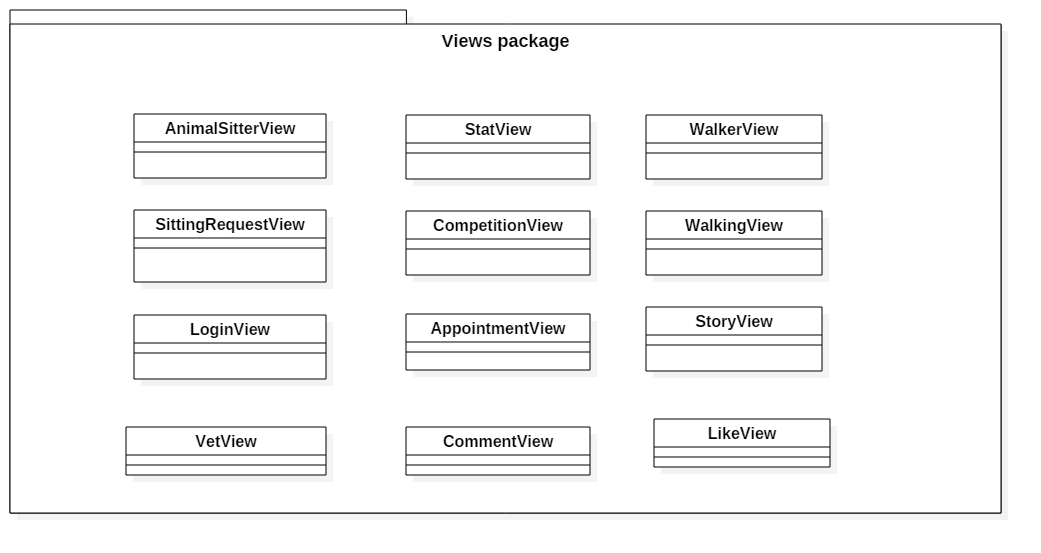


Figure 17 Views package

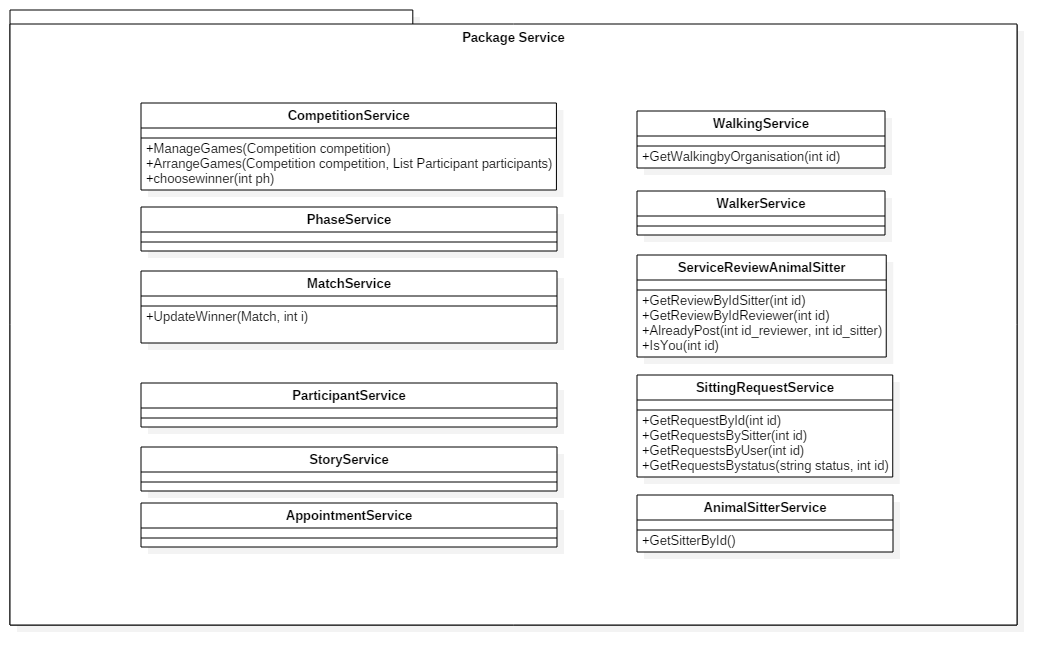


Figure 18 : Application Services

This figure represents all the services we used, including the generic services that are the result of the use of Unit of work, Database Factory and Repository Base design patterns that we shall talk about in the next section.

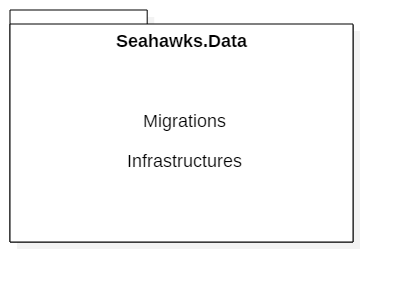


Figure 19 Application Data folder

The Data package contains the migrations after each database update and the infrastructure folder containing the used design patterns, the repository and unit of work patterns are intended to create an abstraction layer between the data access layer and the business logic layer of an application. We implemented these patterns to help insulate our application from changes in the data store.

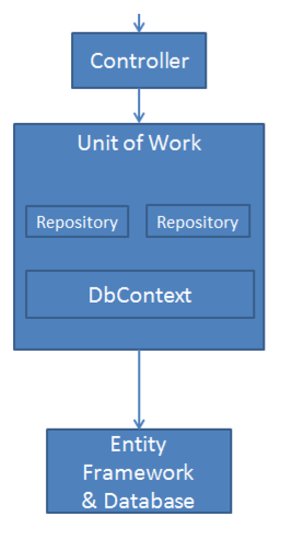


Figure 20 N-tiers architecture process

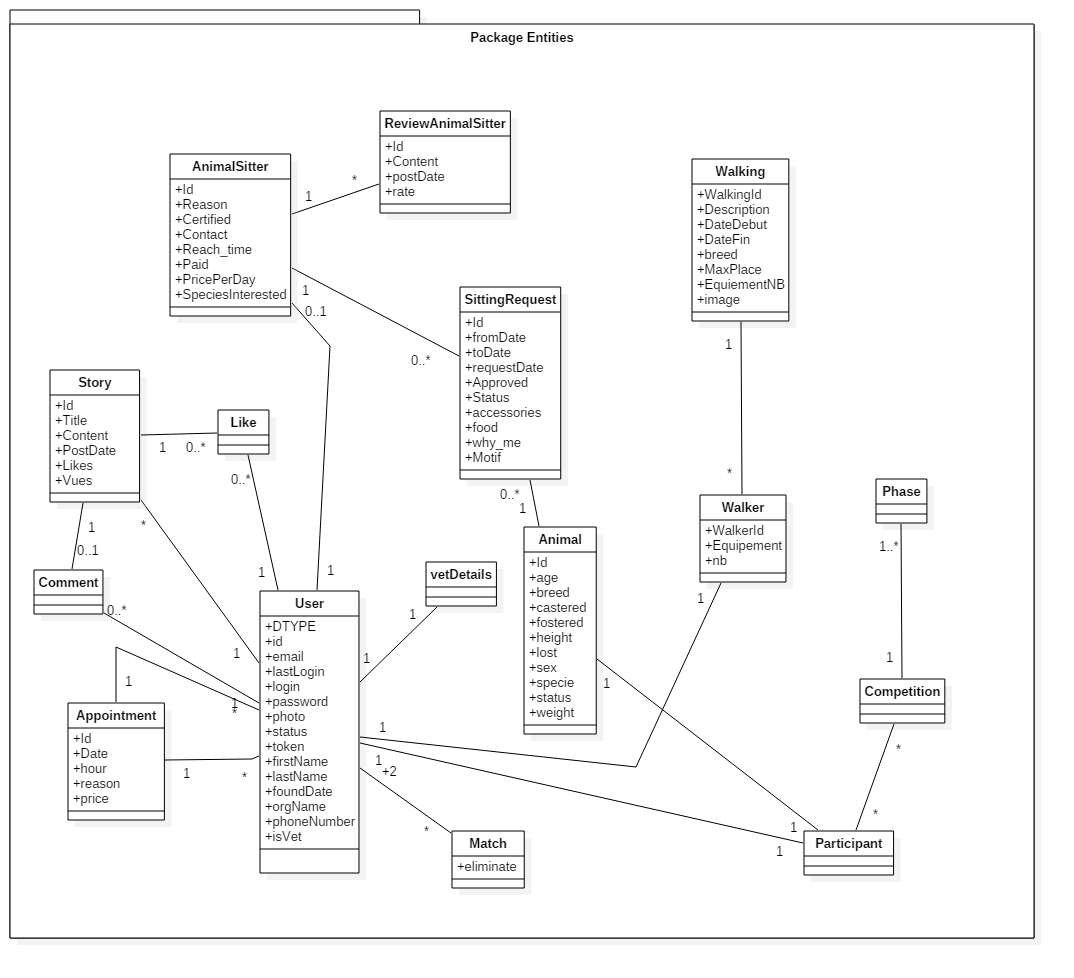


Figure 21 Entities package

The Domain package includes the entities generated from the reverse engineering of the database used in the previous sprints.

1. Sequence diagram

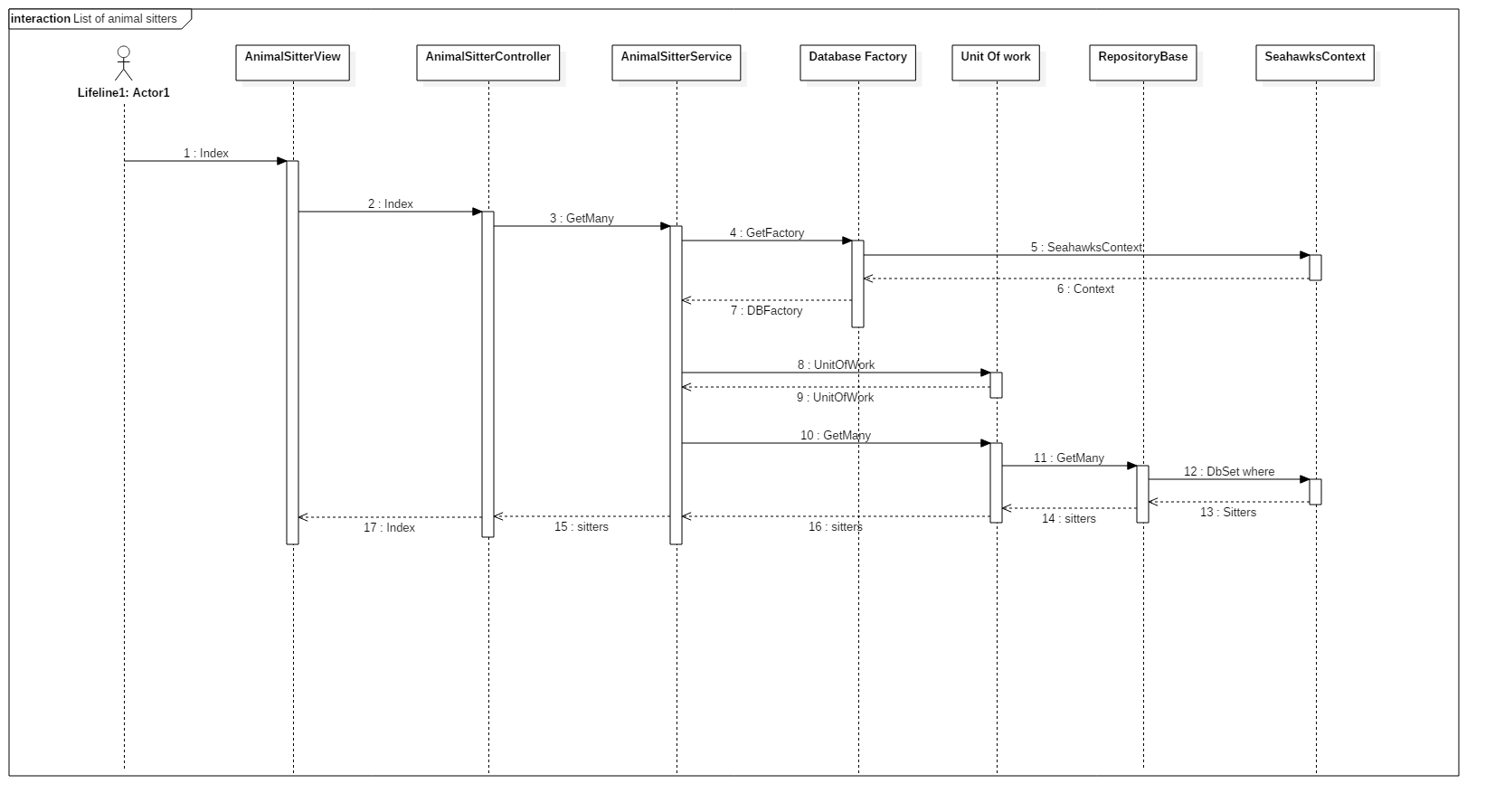


Figure 22 Sequence diagram of " list animal sitters"

1. Component diagram

The component diagram cannot represent the entire system, but a collection of diagrams is used to represent the whole. So, the purpose of the component diagram can be summarized as: Visualize the components of a system, construct executables by using forward and reverse engineering and describe the organization and relationships of the components.

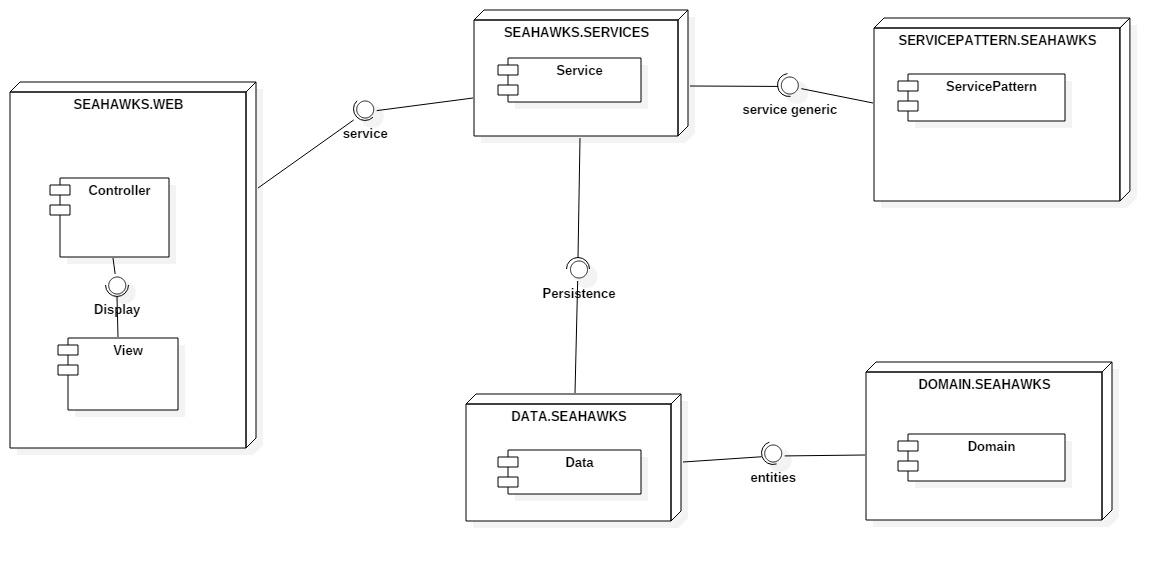


Figure 23 Component diagram

1. Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

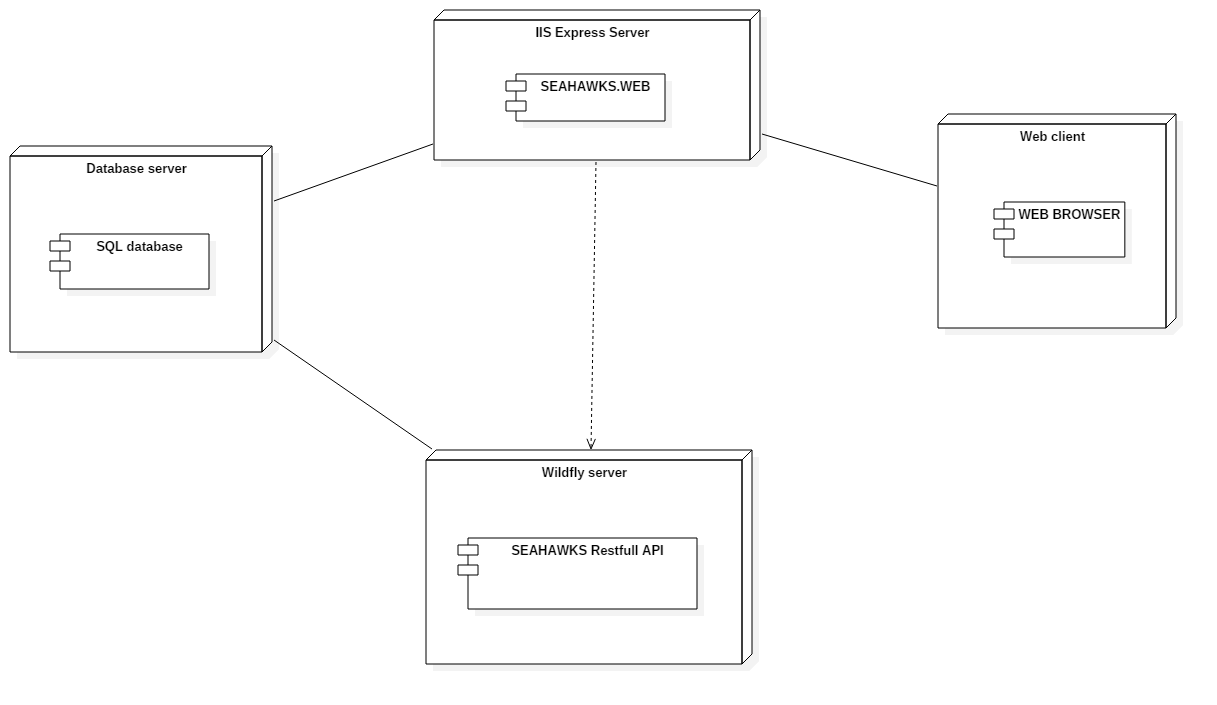


Figure 24 Deployment diagram

1. Conclusion

During this sprint, we developed a part of our application, the features already indicated in the sprint backlog, using ASP.NET. This framework with its project architecture facilitated the work for us.

CHAPTER4: (Sprint 3: Angular 2 & IONIC)

1. Introduction

During this sprint, we conceived the client side of our application in Angular 2. We consumed the REST API that we developed in the Java EE sprint. Angular2 is a structural framework for dynamic web apps classified as an SPA (Single Page Application). Along this sprint, we developed something special to consume the REST API more details should be provided soon.

1. Sprint 3: Overview
2. Approach

As was shown by the team, on the product backlog previously, there was tremendous

features that should be dealt with in this sprint, so after a big deal of time minimizing all of the work to make it easier to produce, the scrum master and the team agreed on 3 main approaches to work on during this sprint. All the user stories will be more detailed.

1. Sprint Backlog
2. Burndown chart
3. Application’s architecture
4. Angular 2 architecture

Angular 2 is using the standard bi-directional data flow model. Once an action is made, the data goes up to the service layer, then it goes back to the view.

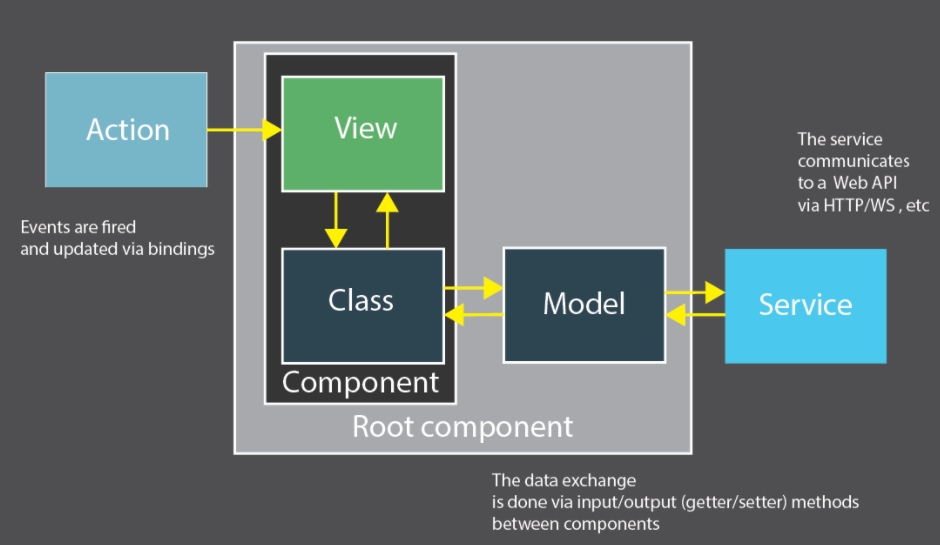


Figure 25 Angular 2 Application Architecture

1. Global Application Architecture

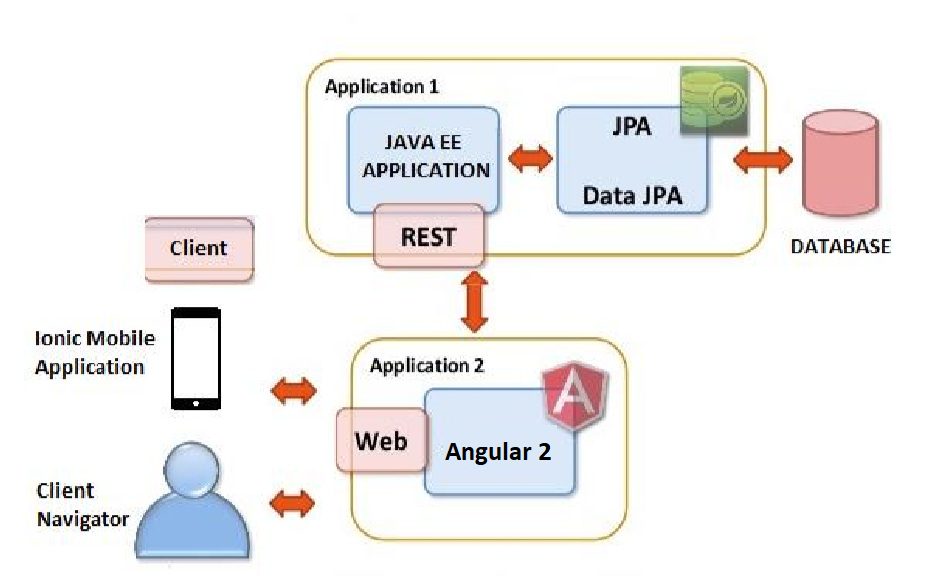


Figure 26 Global application architecture

In Sprint 1, we developed all the necessary services that were exposed as a restful API, in this context, and via the Angular 2 application, these resources are now being consumed.

1. Conception
2. Components diagram

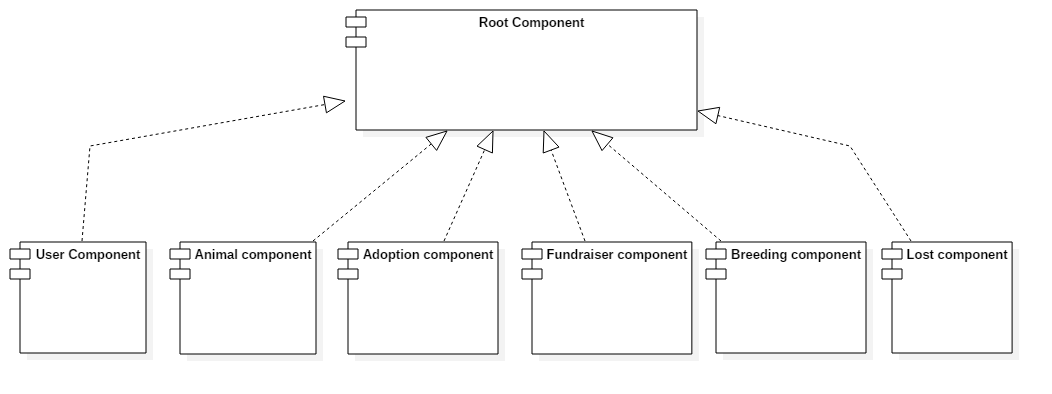


Figure 27 Components diagram

1. Deployment diagram

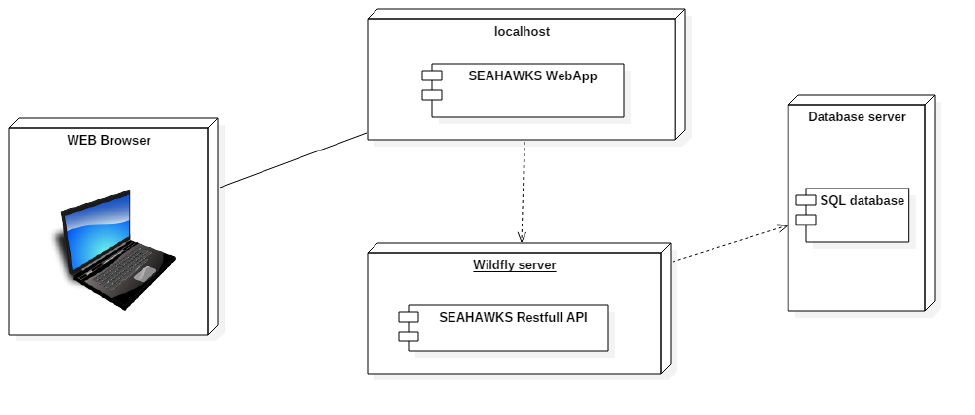


Figure 28 Deployment Diagram

1. Realization
2. Conclusion