Bachelor's Thesis

COMPUTER SCIENCE DEGREE

Faculty of Mathematics

University of Barcelona

SCRUM FRIEND – A WEB APPLICATION FOR AGILE PROJECT MANAGEMENT

Author: Laurențiu Mărcuț

Advisor: Oscar Amoros Huguet

Affiliation: Applied Mathematics and Analysis Department, UB

Barcelona, July 30 2016

Contents

THANKS TO

First, I would like to thank my family first and all the people who supported me during these four years of academic training. I am grateful to all the teachers because I had learned important things from each course.

For this project I want to thank to my supervisor and all my work colleagues who helped and guided me and all the person who participated on the real user test process.

1. INTRODUCTION

In recent years, the complexity of software applications has evolved a lot. As a consequence, building quality products requires agile methods.

Agile methods grew out of the real-life project experiences of leading software professionals who had experienced the challenges and limitations of traditional waterfall development on project after project. The approach promoted by agile development is in direct response to the issue associated with traditional software development, both in terms of overall philosophy as well as specific processes.

Agile development, in its simplest form, offers a lightweight framework for helping teams, given a constantly evolving functional and technical landscape, maintain a focus on the rapid delivery of business value. As a result of this focus, the benefits of agile software development are that organizations are capable of significantly reducing the overall risk associated with software development. [4]

To handle the management of agile development process, any team needs a proper tool. Nowadays, there are plenty of project management applications that help teams to easily follow an agile development process.

1.1. State of the art

Scrum is an iterative and incremental agile software development framework for managing product development.[7] It defines a flexible, holistic product development strategy where a development team works as a unit to reach a common goal, challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project.

Project management software gives any team the ability to organize, collaborate and track details and responsibilities of their projects. The project management software is now over $1 billion with hundreds of competitive solutions. Below is a look at the most popular options as measured by a combination of their total number of customers, users and social presence.

Table 1-1 Most popular list of product management software

Product	**Customers**	**Users**	**Facebook followers**	**LinkedIn followers**	**Twitter followers**
Microsoft Project	880,000	22,000,000	163,532	2,385074	13,810
Attlassian	23,000	30,000,000	35,325	33,411	12,872
Podia	500,000	2,500,000	8,880	4,974	4,682
Write	551,090	1,227,566	19,181	135,860	16,863
Basecamp	285,000	15,000,000	2,723	4,651	67

The previous list includes applications for which users must pay in order to have access to all options and features. There are applications that are free or open source. They have fewer functionalities and are targeted for smaller organizations with small or medium teams and less complex projects.

The most important free project management software programs are:

1. Easy Backlog.
    - Overview - Good user interface and API integration, very easy to manage, but there are no tasks linked to user stories. [7] Only online access.
    - Backlog administration [4] - Easy to create and reorder the product backlog item concerning item priority. Nice drag and drop feature and a good experience for adding and editing items due to the click-to-edit action on each section. The same screen that shows also lets you edit all the information.
    - Managing sprints [4], tasks, estimation and forecasting - Shows the user the story estimation and the cost of it concerning your team man/hour value. You can configure your team speed or get it after the first sprint. Does not have forecasting among sprints. You cannot create tasks linked to the user stories. All the estimation is done directly on the user story.
    - Development team integration - All team members access the same place, but with different permissions. You can manage it in the administration console. There is an API integration so you can manage all your project integration with another software, but you will need to develop it.

2. Visual Studio Online.
    - Overview - Very strong, simple to operate and manage. Good integration with Excel and Visual Studio that makes you get a lot of productivity. VS already comes with source control.
    - Backlog administration - Easy to create and order. Shows Kanban and list view. You can edit your backlog query so you can see it the way you need. Interoperability with Excel, what really makes you save time managing the backlog directly on Excel.
    - Managing sprints, tasks, estimation and forecasting - You can create tasks and bugs linked to your user story. Each task has its estimation and you can fill other fields like "Remaining Work" to update your progress during the sprint. Tasks have the list and Kanban view. A special feature is the forecasting, that virtually simulate where each item would be moved for the next sprint if the development team keeps that same velocity in the current sprint.
    - Development team integration - You can manage your project in Visual Studio or in the online portal. It means that developers can see the product backlog, sprints, tasks and bugs directly on the same software that they use. All the code can also be integrated in the same online repository.

3. Scrum Me.
    - Overview - Very simple and fast, but there are no special functions.
    - Backlog administration - Easy to create, but you cannot edit the item order in the product backlog. The backlog is shown just in Kanban View; you cannot see it in a simple list.
    - Managing sprints, tasks, estimation and forecasting - You can choose your effort measuring unit[1]. It is simple to edit the task status, just by dragging-and-dropping the task in Kanban parts.
    - Development team integration - Each team member accesses the same online environment. There is no integration with development tools.

[1] The measurement units of the effort in Agile methodologies are, in general, hours or points.

1.2. Motivation

As we have seen, neither of the free programs does fulfill all the necessities of the clients. My principal motivation is to build an application that combines all these functionalities and fills the missing gaps such as:

- A minimalist and user-friendly interface is needed so the user can focus only on the content of the managed projects without being distracted by useless elements. If the interface is not easy it may discourage some new users to use these methodologies.
- A clear hierarchy of user roles which offer the access only to the corresponding functionalities, avoiding the possible errors produced by the confusion of the responsibilities. This confusion may lead to problems like: wrong tasks states, incorrect estimation, unclear priority.
- A wide range of diagrams and measurements which offer a graphic visualization of the activity evolution. Not having them may make it more difficult to extract conclusions from the data gathered by the tool.
- An automatic and intelligent recommendation of the actions that can be taken in order to improve the errors which may occur during the project management process. For example, if sprints end always incomplete, the tool may recommend users to plan fewer tasks in the future sprints.
- Autocompleted templates which ease and motivate the users to correctly document the details of different tasks.

This projects can help all developers that want to learn Agile methods and need to develop small or medium projects using low resources. The universities that are teaching these methods may use this software to practice with their students.

1.3. Objectives

As I mentioned in the previous sections, the main existing non paid option are Easy Backlog, Visual Studio Online and Scrum Me. My goal is to do a minimum viable product in where I can iteratively add all the features I think this kind of tool should have. For this purpose, I'm going to use the MVP methodology that I explain next, and can be seen in Image 1.1.

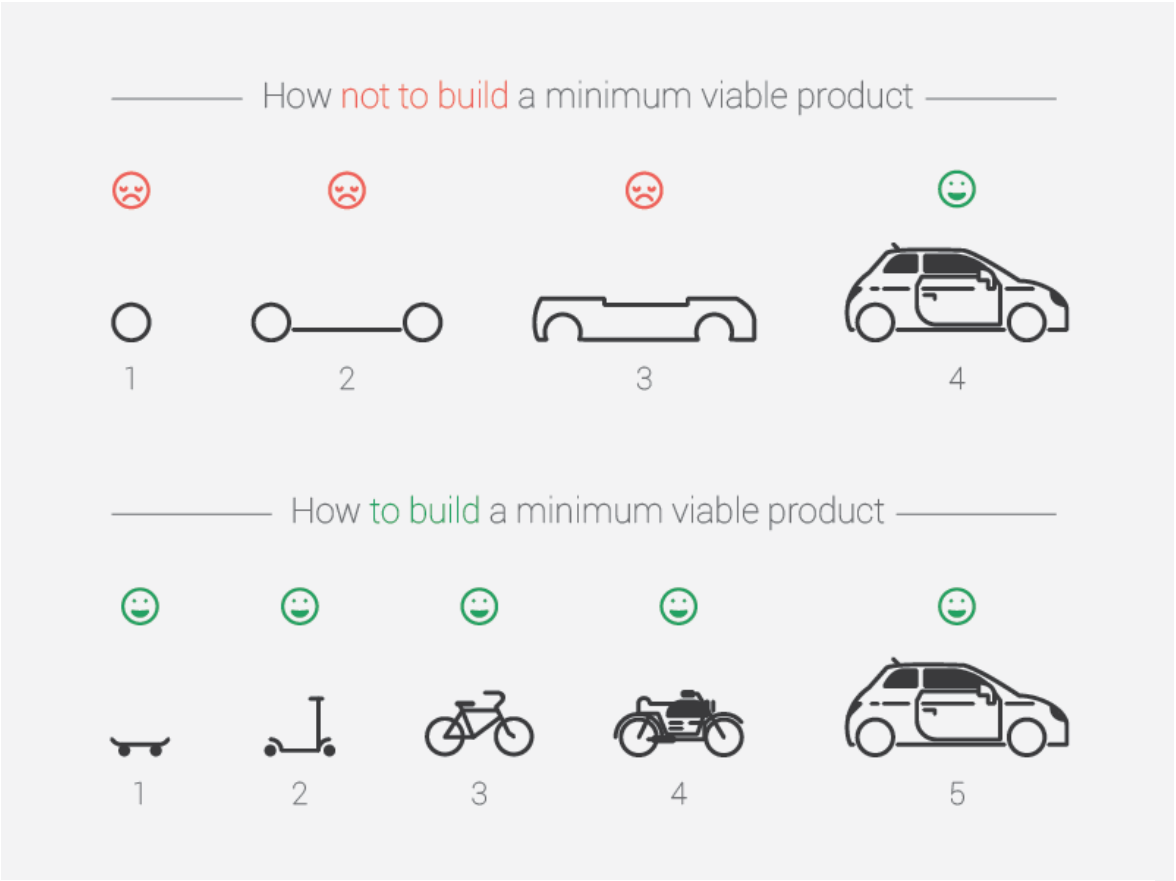


Image 1.1 Minimum viable product

A minimum viable product has only the core features that make it good enough to be deployed, and no more. [6] The MVP is the most pared down version of a product that can still be released. An MVP has three key characteristics:

- It has enough value that people are willing to use it or buy it initially
- It demonstrates enough future benefit to retain early adopters
- It provides a feedback loop to guide future development

I followed the MVP model, and the first iteration of the application, (that I named Scrum Friend) will be covered in this project and will have the following set of objectives:

- Scrum Friend will be a web application with a log in system, where the user can manage multiple projects. The website must have a Product Backlog section where the product owners can create tasks and establish a priority order. The tasks will have the possibility to be assigned to sprints. There must be a relation between the tasks and the members who create them or develop them, an estimation system and a burndown chart diagram for the completed sprints.

- The application must be available from any location with an Internet connection.
- The Scrum Friend’s interface has to be usable and user friendly.
- Zero cost. There are a lot of free tools and services on the market that can be used to create a product like Scrum Friend with all the features needed.
- Speed to market. This project needs to be ready for production on 30th of June.

2. REQUIREMENTS ANALYSIS

To meet the objectives set in the previous chapter it is necessary to obtain and analyze the requirements of the application.

Requirements analysis encompasses those tasks that go into determining the needs to satisfy for a new or altered product or project. This should take into account the possibly of conflicting requirements of the various stakeholders, as long as analyzing, documenting, validating and managing software or system requirements. [6]

Requirements analysis includes three types of tasks:

- Eliciting requirements: the task of communication with customers and users to determine what their requirements are. This is also called requirements gathering. The users involved are product owners, project managers, developers and testers. They are asked about what would need a good project management application and what's missing in the applications they regularly use. As a result, I received a different view on application requirements depending on the type of user interviewed.
- Analyzing requirements: determining whether the stated requirements are unclear, incomplete, ambiguous, or contradictory, and then solving these issues. In this step I have combined all the information to get a unique perspective of what it takes to get a functional application. I have established a boundary between mandatory and optional requirements. Requirements were arranged in ascending order according to the importance it.
- Recording requirements: requirements might be documented in various forms, such as natural-language documents, use cases, user, stories, or process specifications. The information was recorded on the use cases templates and they have been established the acceptance conditions used to declare a requirement accomplished.

To extract all these requirements, I used the FURPS+ model. FURPS+ is an acronym representing a model for classifying software quality attributes (functional and non-functional requirements):

- Functionality - What the customer wants! This includes security-related needs.

- Usability - How effective is the product from the standpoint of the person who must use it? Is it aesthetically acceptable? Is the documentation accurate and complete?
- Reliability - What is the maximum acceptable system downtime? Are failures predictable? Can we demonstrate the accuracy of results? How is the system recovered?
- Performance - How fast must it be? What's the maximum response time? What's the throughput? What's the memory consumption?
- Supportability - Is it testable, extensible, serviceable, installable, and configurable? Can it be monitored?

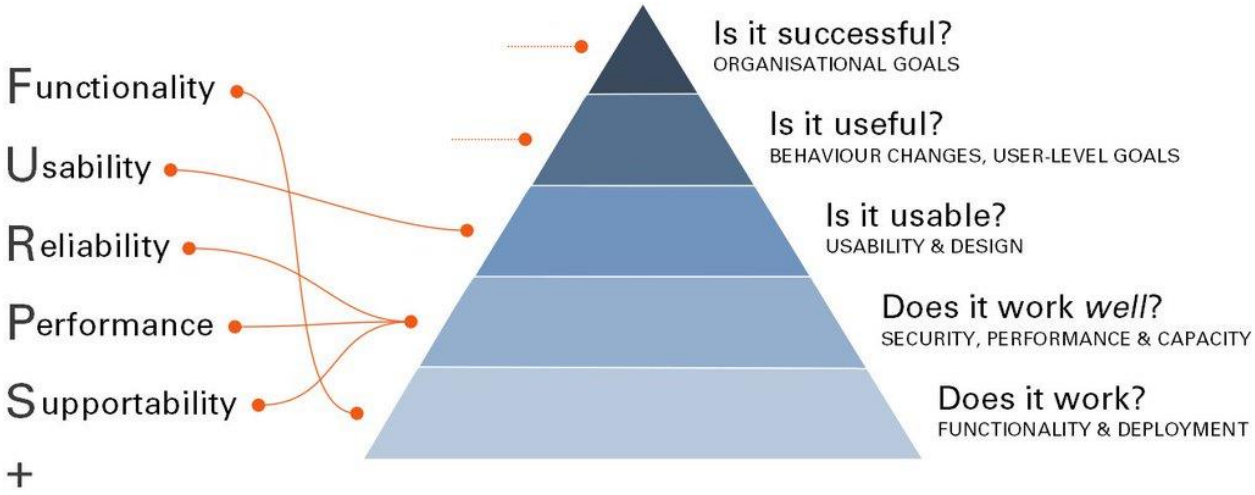


Image 2.1 FURPS+ pyramid of quality criteria

The + reminds us of a few additional needs that a customer could have:

- Design constraints - Do things like I/O devices or DBMS constrain how the software must be built?
- Implementation requirements - Do the programmers need to adhere to standards? Is the use of TDD required? Is statistically sound testing in the context of Cleanroom required?
- Interface requirements - What downstream feeds must be created? What other systems must this one interface with? How frequent are feeds produced?
- Physical requirements - What hardware must the system be deployable on? Must we be able to deploy to a machine no larger than 12" square, to be stationed in the Iraqi desert?

2.1. Functional requirements

The functional requirements define the functions of the system or its components. They are explored and described in the Use Cases Diagram (Annex 1).

A use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal. The actor can be a human or other external system.

This diagram contains two types of actors:

- The normal user - represents any person who accesses the application without authentication.
- The logged user - represents an authenticated person which was previously registered.

I will present the list of use case templates based on the Use Cases Diagram:

Case	**Register**
Description	Create an account to access and save all the user actions
Actor	Normal user
Basic flow	1. Access the main page. 2. Introduce the correct information. 3. The user sends the information to the server. 4. The server validates the information. 5. The server creates the account.
Pre conditions	The email is not registered
Post conditions	The user will have a personal account

Case	**Log in**
Description	Enter into users account
Actor	Normal user
Basic flow	1. Introduce the email and password. 2. The system validates the information. 3. The system shows the main page.
Pre conditions	The user is registered
Post conditions	The user will have access to all features

Case	Log out
Description	Sign out the account
Actor	Logged user
Basic flow	1. The user logs out. 2. The system destroys de session and shows the register page.
Pre conditions	The user is logged
Post conditions	The user will lose the access to the features, but the work will be saved

Case	Create project
Description	Create a new project
Actor	Logged user
Basic flow	1. The user inserts the required information. 2. The system validates the information. 3. The system creates the project and automatically assigns the user as a member of the project. 4. The system shows the first project first in the list.
Pre conditions	The user is logged
Post conditions	The project it's created with the user assigned as member

Case	Remove project
Description	Remove an existing project
Actor	Logged user
Basic flow	1. The user sends the request. 2. The system deletes the project from the system and from the list.
Pre conditions	The project exists
Post conditions	The project is removed from the system and from the list

Case	Unfollow project
Description	Revoke user membership from the project
Actor	Logged user
Basic flow	1. The user unfollows the project.

	2. The system removes the member from the project and the project will disappear from his project list.
Pre conditions	The user is assigned to the project
Post conditions	The project will not be accessed anymore by this user

Case	**Assign member to project**
Description	Assign a new member to the current project
Actor	Logged user
Basic flow	1. The user inserts the email of the new user. 2. The system checks if the email exists and if it is not assigned to the project. 3. The system adds the member to the project and the project will appear in the user list.
Pre conditions	The user exists
Post conditions	The project will be accessed by the new member

Case	**Load project**
Description	Select a project
Actor	Logged user
Basic flow	1. The user selects the project. 2. The system shows the project information.
Pre conditions	The project exists and the user is its member
Post conditions	The user will access the project information

Case	**Create task**
Description	Create a new task to the project
Actor	Logged user
Basic flow	1. The user inserts the required information and sends it to the server. 2. The system validates the information. 3. The system creates and adds the new task at the end of the list.
Pre conditions	A project is selected
Post conditions	The task is created and it can be used

Case	Edit task
Description	Edit the extra information of a task
Actor	Logged user
Basic flow	1. The user can modify the assigned member, cost and status. 2. The system updates the information.
Pre conditions	The task is created
Post conditions	The task will have the information updated

Case	**Remove task**
Description	Delete a task from the project
Actor	Logged user
Basic flow	1. The user sends the delete request. 2. The system removes the task from the list and all the associated information.
Pre conditions	The task exists
Post conditions	The task cannot be accessed

Case	**Create sprint**
Description	Create a new sprint
Actor	Logged user
Basic flow	1. The user sends the request. 2. The system automatically creates a new sprint at the end of the list.
Pre conditions	A project is selected
Post conditions	The sprint can be used with its features

Case	**Remove sprint**
Description	Remove sprint
Actor	Logged user
Basic flow	1. The user sends the request. 2. The system removes the sprint from the list with all its associated information.
Pre conditions	The sprint exists

Post conditions	The sprint cannot be accessed

Case	**Edit sprint**
Description	Edit sprint information
Actor	Logged user
Basic flow	1. The user edits the sprint information: members, capacity, add or remove tasks. 2. The system updates the information.
Pre conditions	A project is selected
Post conditions	The sprint will update the information

Case	**Consult burndown diagram**
Description	Visualize the sprints statistics
Actor	Logged user
Basic flow	1. The user selects a sprint. 2. The system will build a burndown diagram and will show the statistic information.
Pre conditions	The sprint is completed
Post conditions	The burndown diagram will be shown

The security related needs are included as functional requirements:

- On the registration page, the user must introduce his password twice.
- The password must have at least 8 characters.
- Does not allow sensitive data exposure (like passwords).
- A normal user can only access the registration page.
- The database has a daily automated backup system.
- Does not allow code injection on the input fields.
- Does not allow Cross Site Scripting.

2.2. Non-functional requirements

Non-functional requirements specify criteria that can be used to judge the operation

of a system, rather than specific behavior. They are contrasted with functional requirements that define specific behavior or functions. The principal non-requirements are detailed on the following. [6]

Usability (UX) - Human Factors, Aesthetics, Consistency, Documentation, Responsiveness.

To build the interface requirements I used an iterative user design process. Starting from the functional requirements I designed wireframes for each page of the application (Annex 1, 2, 3). They were used to test the aesthetics, consistency and reaction time of the real users. I used three iterations to reach a favorable outcome. These wireframes are the starting point for the style and appearance of the application and may change during development due to user's feedback.

Regarding the usage of colors and effects, I obtained the following requirements:

- The selected project must be colored green to differentiate itself from other projects.
- Unfollow project button must be red in order to avoid erroneous actions.
- When the user adds an element, the system must show the input form in a centered modal.
- The blocked tasks must have a different color.
- The sprint will contain a progress bar that will show the percentages of all task statuses.
- The system must communicate with the user in case of an error or an important action state through messages. They will appear at the top center of the page.
- The tabs must have a color change effect on hover.

Reliability - Availability (Robustness, Durability, Resilience, Recoverability, Survivability), Predictability (Stability), Accuracy (Frequency, Severity of Error).

The database must have a daily automated backup system. When the CPU frequency reaches a high level of usage, a new server instance will be created in order to avoid slow navigation. This may mean an extra cost but the system must be prepared for this scenario. The RAM and HDD must have enough capacity and an alert us in case of an over 70% usage.

Performance - Speed, Efficiency, Resource Consumption (power, ram, cache, etc.), Throughput, Capacity, Scalability.

The application uses a decent amount of resources on the client side in order to work properly with any configuration. On the server side, the system must respond at a less than a second rate.

Hardware limitation will be imposed by the hosting service free tier. It's recommended that the application should not require more than 1 GB RAM, 1 CPU and 20 GB HDD storage. Users from any part of the world can access the website and the system must be auto scalable to offer optimal speed.

Supportability - Testability (Serviceability, Maintainability, Sustainability, Repair Speed), Flexibility (Modifiability, Configurability, Adaptability, Extensibility, Modularity), Installability, Localizability.

The users around the world will need an internet connection to access this application or they can deploy it on their private server. This later option one would increase the security level.

The application will be available for all browsers on desktop environment. The mobile and tablet version can be implemented optionally in the future. The code must be modular and with the capacity to be extended in order to implement other functionalities. It must support Unit test and will include Javadoc in order to be easy to understand by other programmers.

Additional requirements

The database type must be SQL and it must avoid the design constraint. This will help us in the future if we want to change the database. The code implementation will be using agile methodologies. The deploy system will be available from any location.

3. PROJECT PLANNING AND MANAGEMENT

Project planning and management refers to the whole methodology and approach to build this project, including its documentation and demo. This planning is done after analyzing the requirements that were made since only after that, I can gather the information about what needs to be implemented. However, the plan must include the dedicated time towards the requirements analysis in order to have a good vision of the cost and time spent at the end of the project.

The planning was done keeping in mind the deadline set. Thus, according to the priority requirements established in the previous chapter, there were planned only those tasks whose implementation cost were included in the available time and those who were required to provide a complete product.

For a better view of the process flow I will use a Gant chart (Annex 6). A Gantt chart is a type of bar chart, adapted by Karol Adamiecki in 1896 and independently by Henry Gantt in the 1910s, that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project.

I divided the project planning into 6 modules:

1. Requirements – define the project requirements and create the UX interface prototype.
2. Design and architecture – define the project structure, external dependencies, server's architecture, design patterns, database model and all necessary design to mount the basic structure of the project.
3. Development – contains all the components or features necessary for the project implementation.
4. Testing – the test mechanism used to ensure the quality of the final product.
5. Documentation – elaborate the project documentation with a minimum of 60 pages.
6. Demo presentation – create a Power point presentation and an application live demo.

Each module requires an implementation of at least 50% of the previous module to be started. Because the project is developed by a single person, I chose to finish each module to be able to start on the next.

Module 4 was elaborated in parallel with Module 3 because I used a continuous test during the development process. Some diagrams were elaborated in the Requirements or Design and architecture modules but were used later in the documentation.

For better management of the project I relied on Agile methods, like Scrum. I used Sprint planning every two weeks. Each task is estimated in hours and has four states:

- Ready – the task is defined and all the requirements are described. It contains the user stories and the acceptance conditions.
- Ongoing – the task is on the implementation process. This process includes the previous analysis.
- Test – each task has its own unit or integration test and must meet all his acceptance conditions.
- Done – a task is finished after a successful test, master branch integration and deployment.

For each encountered bug, I created a new task and I followed the same procedure.

Image 3.1 Scrum Friend's Kanban table

4. DESIGN AND ARCHITECTURE

The Design and architecture chapter will fit the base for the development step. Here are included the software and hardware infrastructures, necessary to build a robust software which can fulfill all the requirements set out in Chapter 2.

4.1. Application logical architecture

Application logical architecture refers to a detailed design which includes all major components and entities, and their relationship. The objective of this design step is to ensure that all components and functionality is accounted and well understood. It does not refer to any physical server names or addresses, but they include any business services, application names and details, and another relevant information for development purposes.

This web application will use the Java EE programming language, which is a widely used enterprise computing platform developed under the Java Community Process. The version used for Java EE is 1.8.

4.1.1. Model View Controller pattern

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

The MVC components are the following:

- **Model**: The Model component corresponds to all the data related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic related data. For example, a Task object will retrieve the task's information from the database, manipulate it and update it data back to the database or use it to render data.
- **View**: The View component is used for all the UI logic of the application. For example, the Task view would include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

- **Controller**: Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Task controller would handle all the interactions and inputs from the Task View and update the database using the Task Model. The same controller would be used to view the Task data.

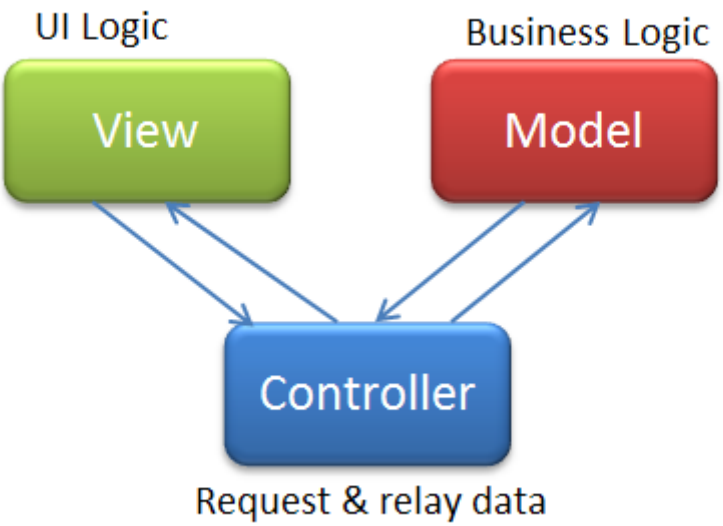


Image 4.1 MVC diagram

For the MVC implementation I used the Spring MVC framework, specially designed for this type of web applications. The Spring MVC framework provides model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing an independency between these elements:

- The Model encapsulates the application data and in general they will consist of POJO.
- The View is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
- The Controller is responsible for processing user requests and building appropriate model and passes it to the view for rendering.

Spring MVC framework is designed around a DispatcherServlet that handles all the HTTP requests and responses. [1] The request processing workflow of the Spring Web MVC DispatcherServlet is illustrated in the following diagram:

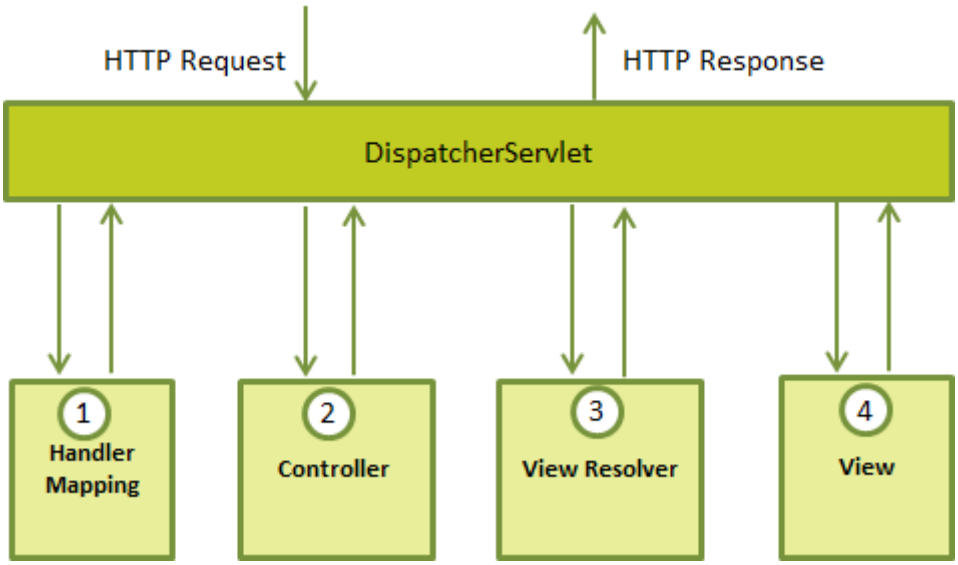


Image 4.2 Spring MVC workflow

Following is the sequence of events corresponding to an incoming HTTP request to DispatcherServlet:

1. After receiving an HTTP request, DispatcherServlet consults the HandlerMapping to call the appropriate Controller.
2. The Controller takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the DispatcherServlet.
3. The DispatcherServlet will take help from ViewResolver to pick up the defined view for the request.
4. Once view is finalized, The DispatcherServlet passes the model data to the view which is finally rendered on the browser.

Spring's web module includes many unique web support features [9]:

- Clear separation of roles. Each role (controller, validator, command object, form object, model object, Dispatcher Servlet, handler mapping, view resolver etc.) can be fulfilled by a specialized object.

- Powerful and straightforward configuration of both framework and application classes as JavaBeans. This configuration capability includes easy referencing across contexts, such as from web controllers to business objects and validators.
- Adaptability, non-intrusiveness, and flexibility. Define any controller method signature you need, possibly using one of the parameter annotations (such as @RequestParam, @RequestHeader, @PathVariable etc.) for a given scenario.
- Reusable business code, no need for duplication. Use existing business objects as command or form objects instead of mirroring them to extend a particular framework base class.
- Customizable binding and validation. Type mismatches as application-level validation errors that keep the offending value, localized date and number binding, and so on instead of String-only form objects with manual parsing and conversion to business objects.
- Customizable handler mapping and view resolution. Handler mapping and view resolution strategies range from simple URL-based configuration, to sophisticated, purpose-built resolution strategies. Spring is more flexible than web MVC frameworks that mandate a particular technique.
- Flexible model transfer. Model transfer with a name/value Map supports easy integration with any view technology.
- Customizable locale and theme resolution, support for JSPs with or without Spring tag library, support for JSTL, support for Velocity without the need for extra bridges etc.
- A simple yet powerful JSP tag library known as the Spring tag library that provides support for features such as data binding and themes. The custom tags allow for maximum flexibility in terms of markup code.
- A JSP form tag library, that makes writing forms in JSP pages much easier.
- Beans whose lifecycle is scoped to the current HTTP request or HTTP Session. This is not a specific feature of Spring MVC itself, but rather of the WebApplicationContext container's that Spring MVC uses.

4.1.2. Logical layering design

Irrespective of the type of application that you are designing, and whether it has a user interface or it is a services application that only exposes services you can decompose the design into logical groupings of software components. These logical groupings are called layers.

Layers help to differentiate between the different kinds of tasks performed by the components, making it easier to create a design that supports reusability of components. Each logical layer contains a number of discrete component types grouped into sub layers, with each sub layer performing a specific type of task. [13]

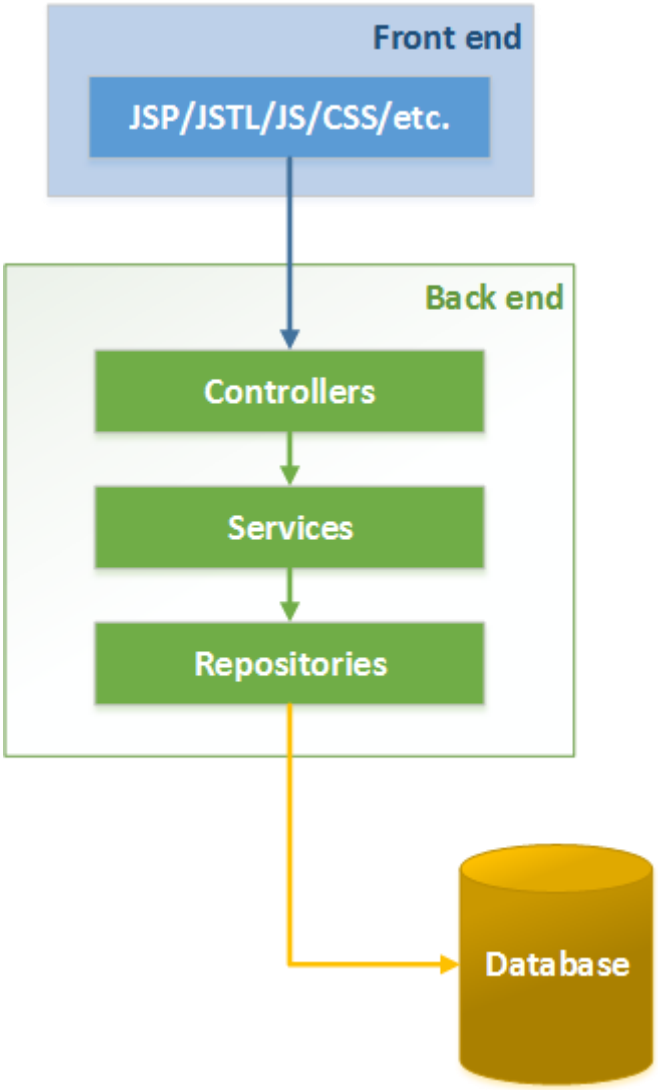


Image 4.3 Application layering

By identifying the generic types of components that exist in most solutions, you can construct a meaningful map of an application or service, and then use this map as a blueprint for your design.

Dividing an application into separate layers that have distinct roles and functionalities helps you to maximize maintainability of the code, optimize the way that the application works when deployed in different ways, and provides a clear delineation between locations where certain technology or design decisions must be made.

The names of the Scrum Friend application layers are inspired by the Spring core annotations and are related with the classic layer's names like: Data Access Object, Business Object, Managers etc.

UI layer (Front end)

Represents the user interface layer. It should not contain any business logic, as well as should not contain any Java code. In my case, I implemented this layer using JSP pages with JSTL and Spring forms (Spring Tag Library). JavaScript is used to implement the events for all elements and CSS and Bootstrap to apply the style and appearance of the application. [13]

In general, JSTL is always used to represent data in JSP (lists, tables, etc.) and avoid using Java code in the pages. Spring forms can help in building HTML forms and handling forms submit in a more convenient way.

Server side (Back end)

- Controllers - are Java classes which handle all input requests from UI layer, perform input parameters validation, call necessary services, handle service responses, wrap response data in a convenient way and return back to the UI layer.
- Services - are Java classes which implement all application business logic. Ideally the service layer does not know anything about web requests or the database layer. Services accept model classes, perform some application business logic actions, call other services, call repositories layer and produce other model classes as a result. Initially services are called from the controller layer. They might use repository classes to load or safe any information in the data layer.
- Repositories - these Java classes provide access (read and write data) to the data layer. Usually relational databases are used as a data layer, but it easily might be something else – for instance, a NoSQL database or a file share.

Besides such application design provides a good loose coupling, such as code separation allowing to unit test the web application in a convenient way. Each back end layer can be easily covered with unit tests independently from each other.

4.1.3. Data model

To analyze the database infrastructure, I created the Database Entity Relationship (ER) model from the Annex 5. The ER model defines the conceptual view of the database. It works around real-word entities and the associated among them. At view level, the ER model is considered a good option for designing databases.

I chose MySQL database because it is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS, being used for developing web-based software applications.

So nowadays, I use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys. [5]

A Relational Database Management System (RDBMS) is a software that:

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is becoming so popular because of many good reasons:

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.

- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

This type of database meets with the additional requirements established on the Chapter 2.

4.1.4. Project and dependency management

For a good management of external libraries and internal structure of the project I used Apache Maven.

Apache Maven is a build automation tool used primarily for Java projects. The word maven means "accumulator of knowledge" in Yiddish. Maven addresses two aspects of building software: first, it describes how software is built, and second, it describes its dependencies.

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:

- Making the build process easy.
- Providing a uniform build system.
- Providing quality project information.
- Providing guidelines for best practices development.
- Allowing transparent migration to new features.

Maven allows a project to build using its project object model (POM) and a set of plugins that are shared by all projects using Maven, providing a uniform build system.

One of the strongest points of Maven is that it automatically manages project dependencies. The developer just needs to specify which dependencies and in which version are needed and Maven takes care of the rest, including downloading and storing them at the right location and additionally packaging them into a final artifact (WAR).

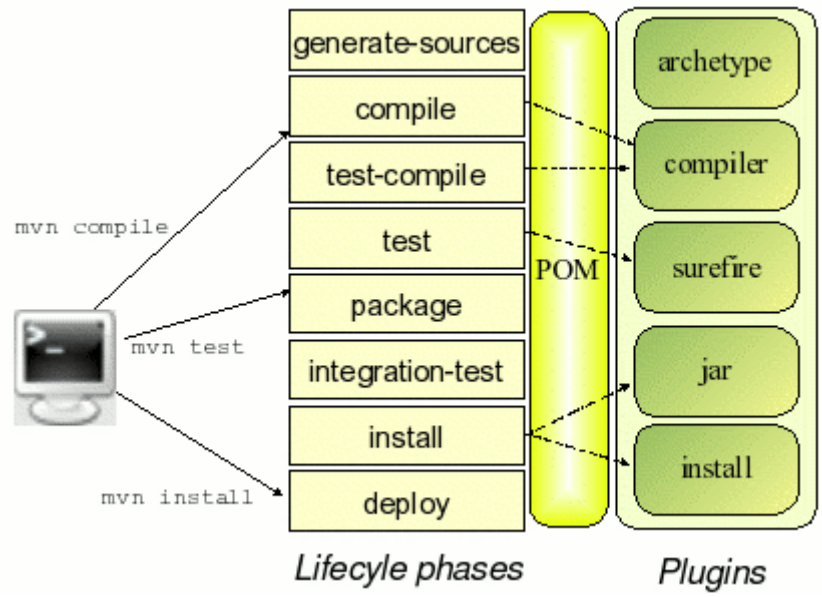


Image 4.4 Maven workflow

This application is organized in a unique artifact and the principal use of Maven is for the dependency management, project building and unit tests launching.

The POM file includes the ultimate version of all external libraries used: Spring, Apache Tiles, JDBC, Validation API, Hibernate validator, Jackson, Log4j, MySQL connector etc.

4.2. Application physical architecture

A physical design has all major components and entities identified within specific physical servers and locations or specific software services, objects, or solutions. It includes all known details such as operating systems, version numbers, and even patches that are relevant. Any physical constraints or limitations should also be identified within the server components, data flows, or connections.

This subchapter contains all the hardware infrastructure necessary to run the application.

4.2.1. Three tier architecture

As the need for enterprise scalability is increasing, it challenged the traditional two tier architecture. In 1995, a new variant of two tier architecture came into existence where there was reduction in client-side headache. [12]

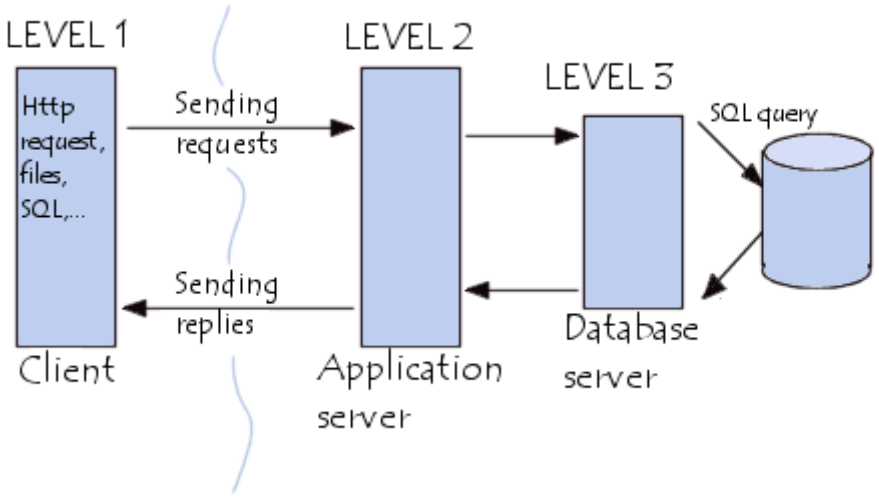


Image 4.5 Three tier architecture

Such systems comprised of following three levels/tiers [12]:

1. **User Interface Layer**: This layer/tier comprised of functionality for simple GUI required for communication with database server and it runs on end user's computer. It sometimes performs a role of validating the input at user's side.
2. **Business Logic & Data Processing Layer**: It acts as a middleware which runs on separate machine and whose responsibility is to handle business and application logic. Sometimes it is also known as "Application Server".
3. **Database Server**: It stores the data required by middle layer. This layer may run on separate server known as database server whose primary job is to handle all database related requests from client side or through middle layer.

As there is clear separation between business logic and user interface, client will require very few resources. So it is known as "Thin Client".

Advantages of 3-Tier Architecture:

- It is less expensive as client requires very limited amount of resources like hardware, memory etc.
- Application maintenance is centralized with transfer of business logic for many clients into a single application server.
- The added modularity makes it convenient for modification or replacement of any of the tiers without affecting others.

- Load balancing is easier with the separation of core business logic from database functions.

This architecture easily maps to web environment with a web browser acting as "Thin Client' and web server acting as application server. This architecture can be easily extended to N tiers which will provide more flexibility and modularity.

To implement this type of architecture, the most recommended provider is Amazon Web Services.

4.2.2. Amazon Web Services

Amazon Web Services (AWS), is a subsidiary of Amazon.com which offers a suite of cloud computing services that make up an on-demand computing platform. These services operate from 12 geographical regions across the world. I am using for this application the Frankfurt region due to the short distance which offers the highest speed.

Amazon offers a free tier for students which gives us everything we need to build hardware for our three tier application architecture.

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. [8]

Amazon EC2 contains the Application server and it can be accessed by all IP's through ports 80 and 443. It has a 64bit Linux operating system installed with Tomcat 8 and Java 8. All these features can be configured when the EC2 instance is created.

The instance type used on this application is an "t2.micro" from the free tier with one CPU and 1 GB of RAM. This hardware configuration is enough for a demo presentation but it is not recommended for a massive production environment.

The IP of the machine is 52.29.208.111 and it can be used to access the log files or other server information or to configure the security access groups.

Benefits of using EC2:

- Elastic Web-Scale Computing - Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds or even thousands of server instances simultaneously. Of course, because this is all controlled with web service APIs, your application can automatically scale itself up and down depending on its needs.

- Completely Controlled - You have complete control of your instances. You have root access to each one, and you can interact with them as you would with any machine. You can stop your instance while retaining the data on your boot partition and then subsequently restart the same instance using web service APIs. Instances can be rebooted remotely using web service APIs. You also have access to console output of your instances.
- Flexible Cloud Hosting Services - You have the choice of multiple instance types, operating systems, and software packages. Amazon EC2 allows you to select a configuration of memory, CPU, instance storage, and the boot partition size that is optimal for your choice of operating system and application. For example, your choice of operating systems includes numerous Linux distributions, and Microsoft Windows Server.
- Designed for use with other Amazon Web Services - Amazon EC2 works in conjunction with Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon SimpleDB and Amazon Simple Queue Service (Amazon SQS) to provide a complete solution for computing, query processing and storage across a wide range of applications.
- Reliable - Amazon EC2 offers a highly reliable environment where replacement instances can be rapidly and predictably commissioned. The service runs within Amazon's proven network infrastructure and data centers. The Amazon EC2 Service Level Agreement commitment is 99.95% availability for each Amazon EC2 Region.
- Inexpensive - Amazon EC2 passes on to you the financial benefits of Amazon's scale. You pay a very low rate for the compute capacity you actually consume. See Amazon EC2 Instance Purchasing Options for a more detailed description.
- Easy to Start - Quickly get started with Amazon EC2 by visiting AWS Marketplace to choose preconfigured software on Amazon Machine Images (AMIs). You can quickly deploy this software to EC2 via 1-Click launch or with the EC2 console.
- Secure.

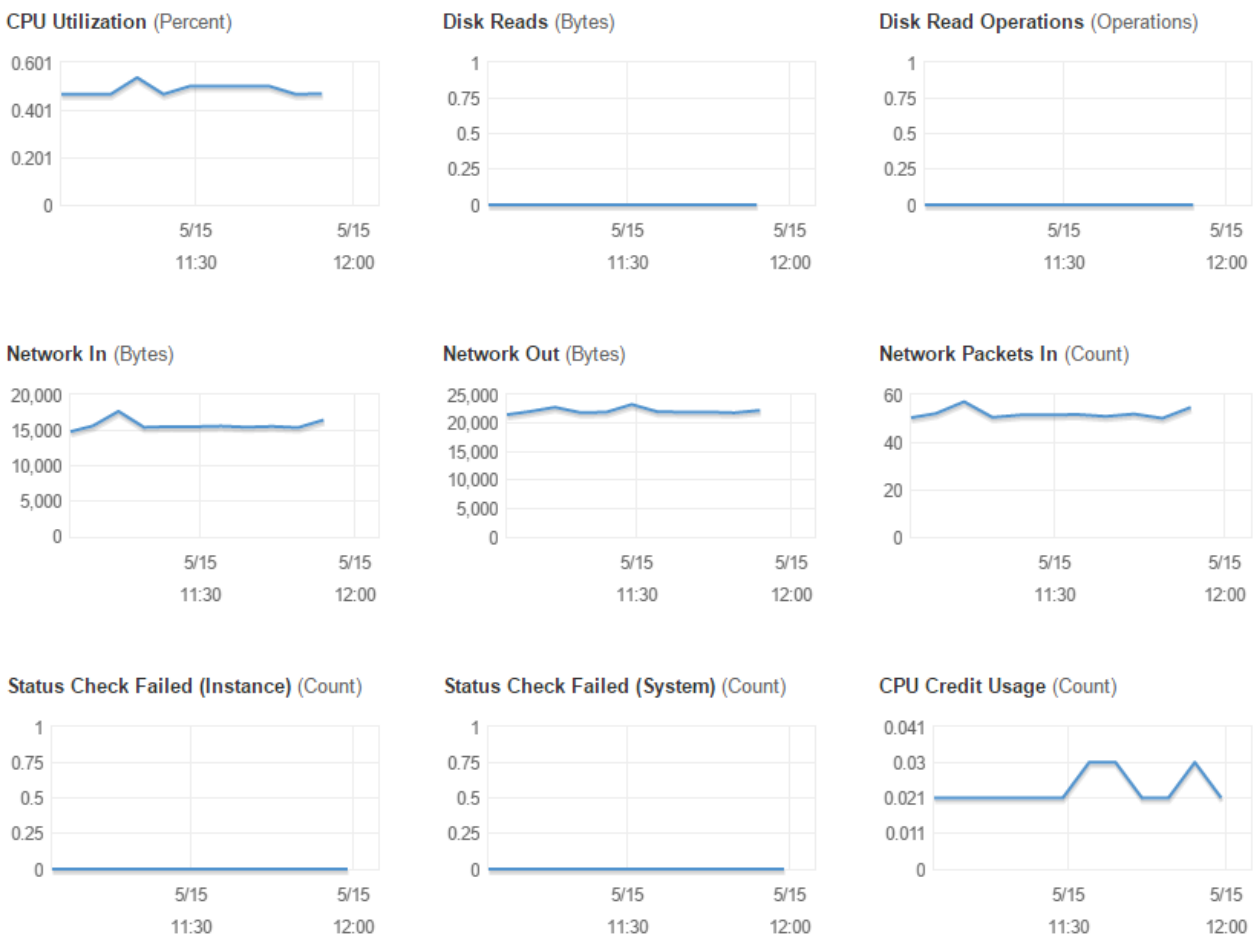


Image 4.6 Amazon EC2 statistics

With the management console I can read a list of metrics for different resources and I configured an alarm with Cloud Watch which sends me an email when the usage of the CPU surpasses 70%. This feature is related with the non-functional requirements from the Chapter 2.

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you up to focus on your applications and business. Amazon RDS provides you six familiar database engines to choose from, including Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL and MariaDB. [8]

The Scrum Friend application uses a MySQL database because of its free license and it is hosted by the Amazon RDS. The server hardware configuration contains a “db.t2.micro” instance with one CPU, 1 GB RAM and 20 GB magnetic storage.

For a better security level, I configured the RDS IP's to be accessed only from the EC2 instance and my home location. The automatic backup is enabled and it is refreshing every day.

Benefits of using RDS:

- Easy to Administer - Amazon RDS makes it easy to go from project conception to deployment. Use the AWS Management Console, the AWS RDS Command-Line Interface, or simple API calls to access the capabilities of a production-ready relational database in minutes. No need for infrastructure provisioning, and no need for installing and maintaining database software.
- Scalable - You can scale your database's compute and storage resources with only a few mouse clicks or an API call, often with no downtime. Many Amazon RDS engine types allow you to launch one or more Read Replicas to offload read traffic from your primary database instance.
- Available and Durable - Amazon RDS runs on the same highly reliable infrastructure used by other Amazon Web Services. When you provision a Multi-AZ DB Instance, Amazon RDS synchronously replicates the data to a standby instance in a different Availability Zone (AZ). Amazon RDS has many other features that enhance reliability for critical production databases, including automated backups, database snapshots, and automatic host replacement.
- Fast - Amazon RDS offers database server sizing choices up to 32 vCPUs and 244 GB, as well as storage choices for a wide range of application performance requirements. You can choose SSD-backed storage optimized for high-performance OLTP applications or for cost-effective general-purpose use. You can also choose magnetic storage for workloads in which data is accessed less frequently.
- Inexpensive - You pay very low rates and only for the resources you actually consume. In addition, you benefit from the option of On-Demand pricing with no up-front or long-term commitments, or even lower hourly rates via our reserved pricing option.
- Secure.

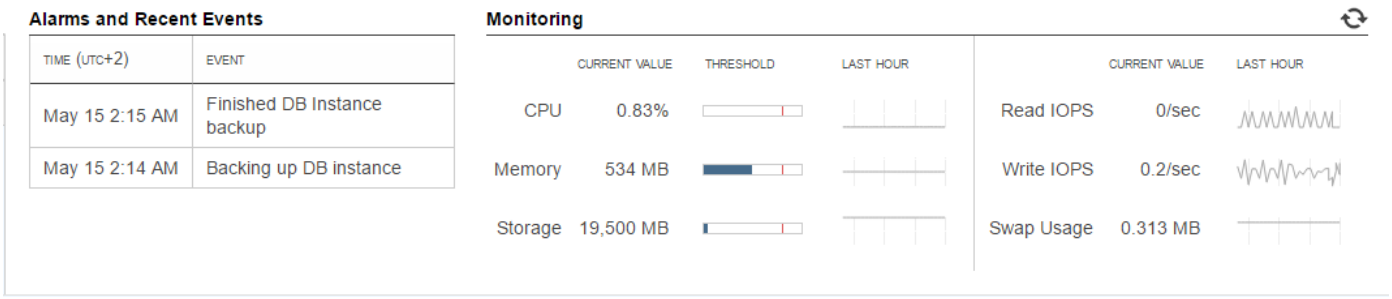


Image 4.7 RDS monitoring

The RDS instance can be monitored by the Amazon management console. I can see the most recent events, the CPU or memory using. This instance is not recommended for a massive production environment because of its low hardware configuration.

4.2.3. Security

These security features are related to the functional requirements. I will present what Amazon gives us to maintain the security of the application.

Amazon EC2 works in conjunction with Amazon VPC to provide security and robust networking functionality for your computer resources. [2]

- Compute instances are located in a Virtual Private Cloud (VPC) with an IP range that you specify. You decide which instances are exposed to the Internet and which remain private.
- Security Groups and networks ACLs allow you to control inbound and outbound network access to and from your instances.
- You can connect your existing IT infrastructure to resources in your VPC using industry-standard encrypted IPsec VPN connections.
- For additional isolation, you can provision your EC2 resources on Dedicated Hosts or as Dedicated Instances. Both allow you to use EC2 instances in a VPC on hardware dedicated to a single customer.

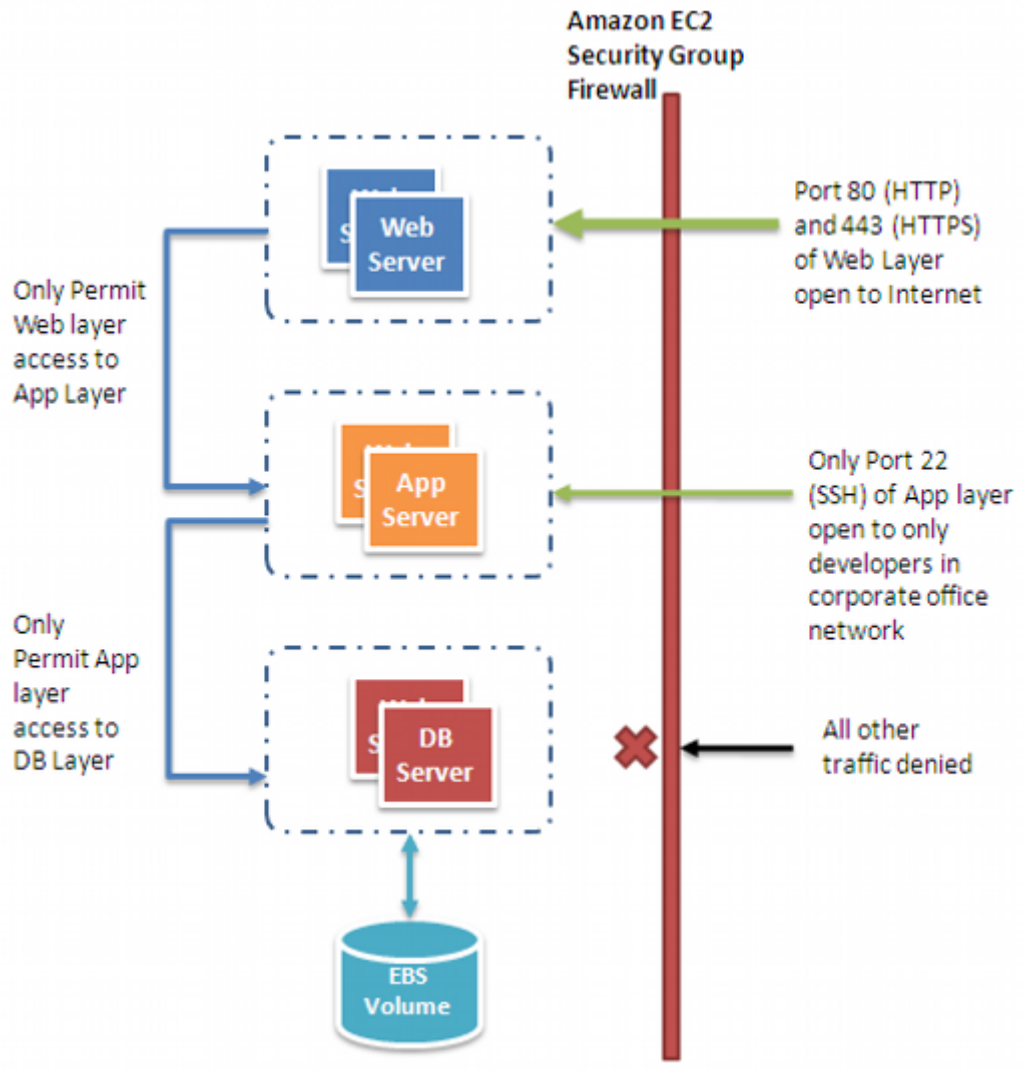


Image 4.8 Amazon security workflow

Source: https://www.amazon.com

My RDS security it's based on security groups and automatic backups. The security groups allow the access only for the Application Server and my home location IP.

5. DEVELOPMENT

In this chapter I will present the main features implemented in the application. This is the next step after the architecture and design. It is more oriented to coding and offers a certain vision of the software built and technologies used.

5.1. Development methodology

Describes the necessary tools for the development process, things like where the code will be stored and how it will pass from application to hardware infrastructure.

5.1.1. Version control

GitHub is a web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.

Unlike Git, which is strictly a command tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, requests, task, and wikis for every project.

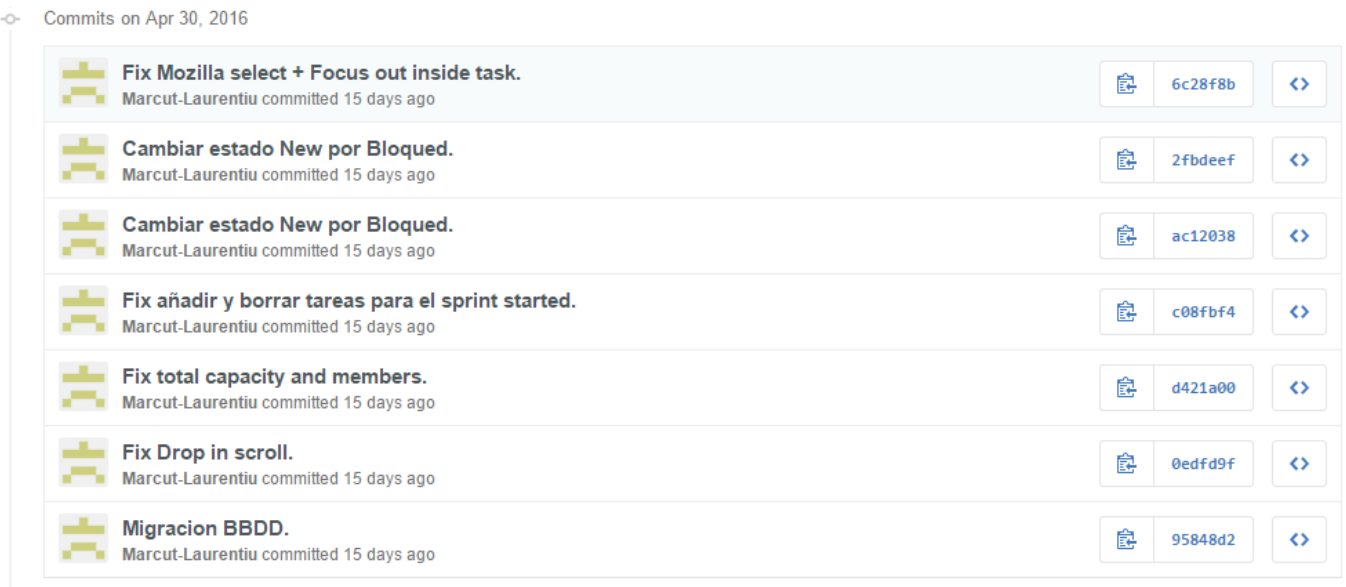


Image 5.1 GitHub commits list

In addition to source code, GitHub supports the following formats and features:

- Putting the code on the cloud in order to accessed from any location.
- Documentation, including automatically-rendered README files in a variety of Markdown-like file formats.
- Issue tracking including feature requests with labels, milestones, assignees and a search engine.
- Commits history.
- Unified and split diffs.
- Reverting to an older version.
- Graphs: pulse, contributors, commits, code frequency, punch card, network, members.

5.1.2. Deployment methodology

With AWS Elastic Beanstalk, you can quickly deploy and manage applications in the AWS cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and AWS Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

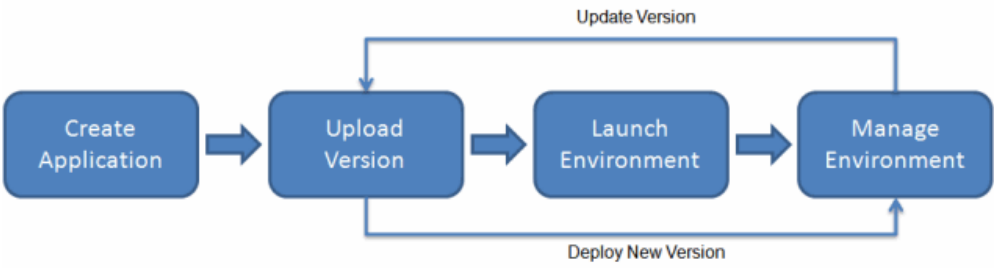


Image 5.2 Deployment methodology with Elastic Beanstalk

Source: https://www.amazon.com

You can also perform most deployment tasks, such as changing the size of your fleet of Amazon EC2 instances or monitoring your application, directly from the Elastic Beanstalk web interface.

To use Elastic Beanstalk, you create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically

launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions.

After you create and deploy your application, information about the application including metrics, events, and environment status is available through the AWS Management Console, APIs, or Command Line Interfaces.

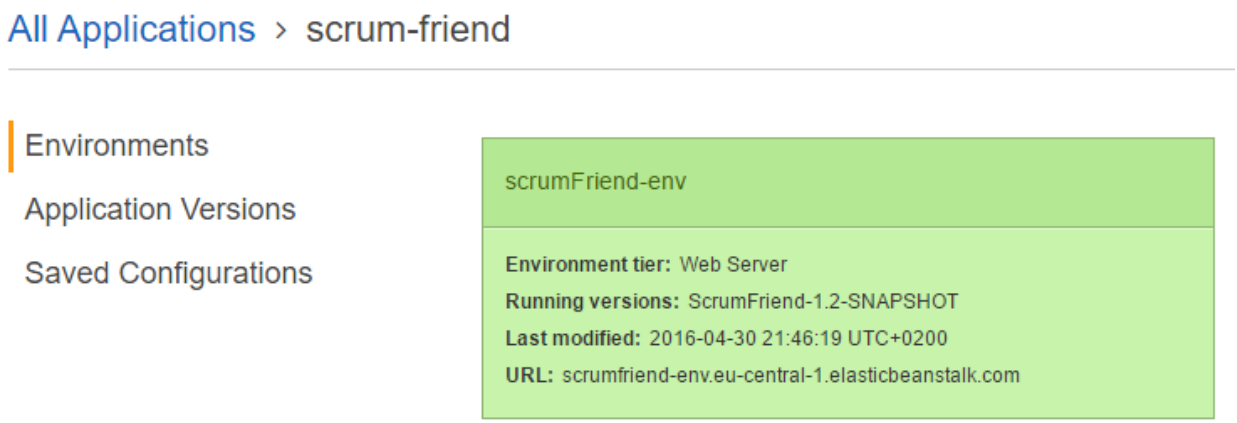


Image 5.3 Manage applications with Elastic beanstalk

Source: https://www.amazon.com

5.1.3. Integrated development environments

The following list of tools are necessary for building the code of the application.

IntelliJ IDEA is a Java integrated development environment (IDE) for developing computer software. The principal benefits of this IDE are:

- Deep intelligence - After IntelliJ IDEA's indexed your source code, it offers blazing fast and intelligent experience by giving relevant suggestions in every context: instant and clever code completion, on-the-fly code analysis and reliable refactoring tools.
- Out-of-the-box experience - Mission-critical tools such as integrated version controls systems and a wide variety of supported languages and frameworks are at hand — no plugin hustle included.
- Smart code completion - While the basic completion suggests names of classes, methods, fields, and keywords within the visibility scope, the smart completion suggests only those types that are expected in the current context.

- Framework-specific assistance - While IntelliJ IDEA is an IDE for Java, it also understands and provides intelligent coding assistance for a large variety of other languages such as SQL, JPQL, HTML, JavaScript, etc., even when the language expression is injected into a String literal in your Java code.
- Productivity boosters - The IDE predicts your needs and automates the tedious and repetitive development tasks so you can stay focused on the big picture.
- Unobtrusive intelligence - The coding assistance in IntelliJ IDEA is not only about the editor: it helps you stay productive when dealing with its other parts as well: e.g. filling a field, searching over a list of elements; accessing a tool window; or toggling for a setting, etc.

DataGrip is a database IDE developed by JetBrains that is tailored to suit specific needs of professional SQL developers and DBAs. DataGrip has built in support for many relational database platforms. I used this database client to construct the entire database infrastructure and to insert SQL statements in order to create virtual data for testing.

- Intelligent query console - Allows you to execute queries in different modes and provides local history that keeps track of all your activity and protects you from losing your work.
- Efficient schema navigation - Lets you jump to any table, view, or procedure by its name via corresponding action, or directly from its usages in the SQL code.
- Explain plan - Gives you an extended insight into how your queries work and into database engine behavior so you can make your queries more efficient.
- Smart code completion - DataGrip provides context-sensitive code completion, helping you to write SQL code faster. Completion is aware of the tables structure, foreign keys, and even database objects created in code you're editing.
- On-the-fly analysis and quick-fixes - DataGrip detects probable bugs in your code and suggests the best options to fix them on the fly. It will immediately let you know about unresolved objects, using keywords as identifiers and always offers the way to fix problems.
- Refactoring’s that work in SQL files and schemas - DataGrip correctly resolves all references in your SQL code and helps you refactor them. When you rename a variable or an alias, it will update their usages throughout the entire file. Even

the actual table names in the database are updated when you rename references to them from your queries.

- Version control integration - They provide unified support for all major version control systems: Git, SVN, Mercurial and many others.

SourceTree is a powerful Git and Mercurial desktop client for developers on Mac or Windows. SourceTree simplifies how you interact with your Git and Mercurial repositories so you can focus on coding. Visualize and manage your repositories through SourceTree's simple interface.

AWS Management Console is a simple and intuitive web-based user interface. You can also use the AWS Console mobile app to quickly view resources on the go. I used this console to manage all the server configuration. [8]

The Console provides cloud management for all aspects of your AWS account, including monitoring your monthly spending by service, managing security credentials, or even setting up new IAM Users.

With the AWS Console mobile app, you can quickly and easily view your existing resources, including Cloud Watch alarms, and perform operational tasks from your mobile device.

5.2. Front End features implementation

The Front End includes all the features of the presentation layer which is the interface between the user and the back end. This part is related with the view layer of the MVC pattern (Subchapter 4.1.1. Model View Controller pattern) and most of it runs in the browser, which is the client tier (Subchapter 4.2.1. Three tier architecture).

5.2.1. Dynamic web pages with JSP

JavaServer Pages (JSP) is a technology that lets you add dynamic content to web pages. In absence of JSP, to update the appearance or the content of plain static HTML pages, you always have to do it by hand. Even if all you want to do is change a date or a picture, you must edit the HTML file and type in your modifications. Nobody is going to do it for you, whereas with JSP, you can make the content dependent on many factors, including the

time of the day, the information provided by the user, the user's history of interaction with your web site, and even the user's browser type.

This capability is essential to provide online services in which you can tailor each response to the viewer who made the request, depending on the viewer's preferences and requirements.

A crucial aspect of providing meaningful online services is for the system to be able to remember data associated with the service and its users. That's why databases play an essential role in dynamic web pages. But let's take it one step at a time.

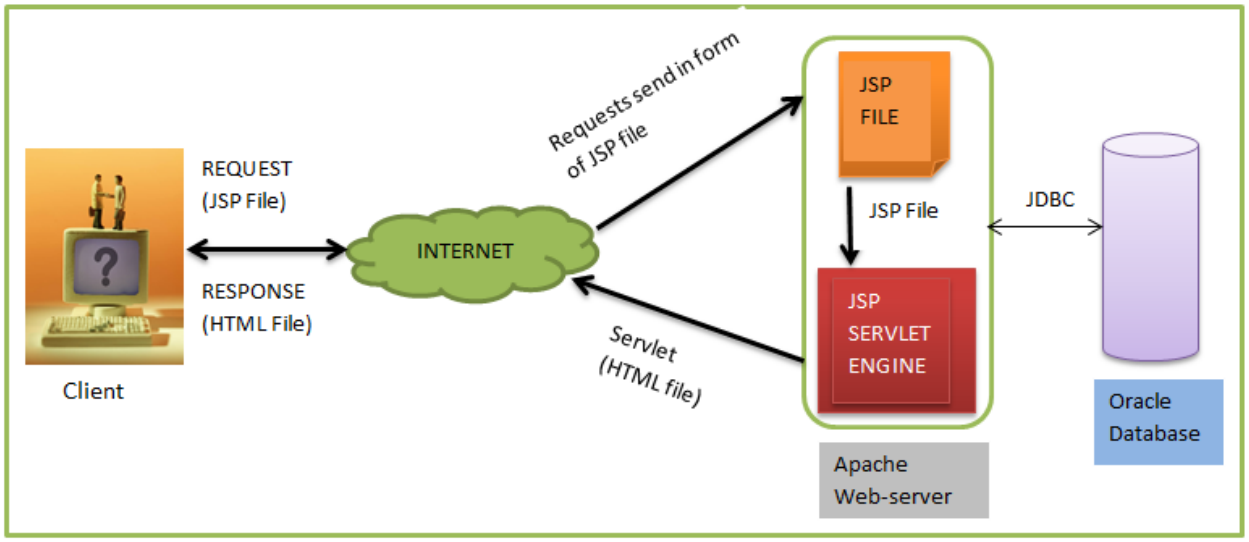


Image 5.4 Dynamic JSP workflow

The following steps explain how the web server creates the web page:

1. As with a normal page, your browser sends an HTTP request to the web server. This doesn't change with JSP, although the URL probably ends in .jsp instead of .html or .htm.
2. The web server is not a normal server, but rather a Java server, with the extensions necessary to identify and handle Java servlets. The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine.
3. The JSP engine loads the JSP page from disk and converts it into a Java servlet. From this point on, this servlet is indistinguishable from any other servlet developed directly in Java rather than JSP, although the automatically generated Java code of a JSP servlet is not always easy to read, and you should never modify it by hand.

4. The JSP engine compiles the servlet into an executable class and forwards the original request to another part of the web server called the servlet engine. Note that the JSP engine only converts the JSP page to Java and recompiles the servlet if it finds that the JSP page has changed since the last request. This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster.
5. The servlet engine loads the servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
6. The web server forwards the HTTP response to your browser. 7. Your web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page. In fact, static and dynamic web pages are in the same format.

The JavaServer Pages API allows you to define custom JSP tags that look like HTML or XML tags and a tag library. A tag library is a set of user-defined tags that implements custom behavior. The taglib directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in your JSP page.

The list of taglibs used in this application is:

- http://www.springframework.org/tags
- http://java.sun.com/jsp/jstl/core
- http://www.springframework.org/tags/form
- http://java.sun.com/jsp/jstl/functions
- http://tiles.apache.org/tags-tiles

The principal advantages of using JSP in my application are:

- the construction of dynamic lists of elements (projects, tasks, sprints, members etc.).
- variables for conditional cases (show the state of a task, sprint etc.).
- inclusion of other JSP files in the main pages (we avoid repeated code and I gain modularity).

5.2.2. Asynchronous communication using Ajax and Restful web services

Ajax is a set of web development techniques using many web technologies on the client-side to create asynchronous Web applications. With Ajax, web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

By decoupling the data interchange layer from the presentation layer, Ajax allows for web pages, and by extension web applications, to change content dynamically without the need to reload the entire page. Despite the name, the use of XML is not required (JSON is often used in the AJAJ variant), and the requests do not need to be asynchronous.

The term Ajax has come to represent a broad group of Web technologies that can be used to implement a Web application that communicates with a server in the background, without interfering with the current state of the page. The following technologies are incorporated:

- HTML (or XHTML) and CSS for presentation.
- The Document Object Model (DOM) for dynamic display of elements and interaction with data.
- JSON or XML for the interchange of data.
- The XMLHttpRequest object for asynchronous communication.
- JavaScript to bring these technologies together.

RESTful web services are built to work best on the Web. Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web.

In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.

The following principles encourage RESTful applications to be simple, lightweight, and fast:

- Resource identification through URI: A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery.
- Uniform interface: Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.
- Self-descriptive messages: Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.
- Stateful interactions through hyperlinks: Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction.

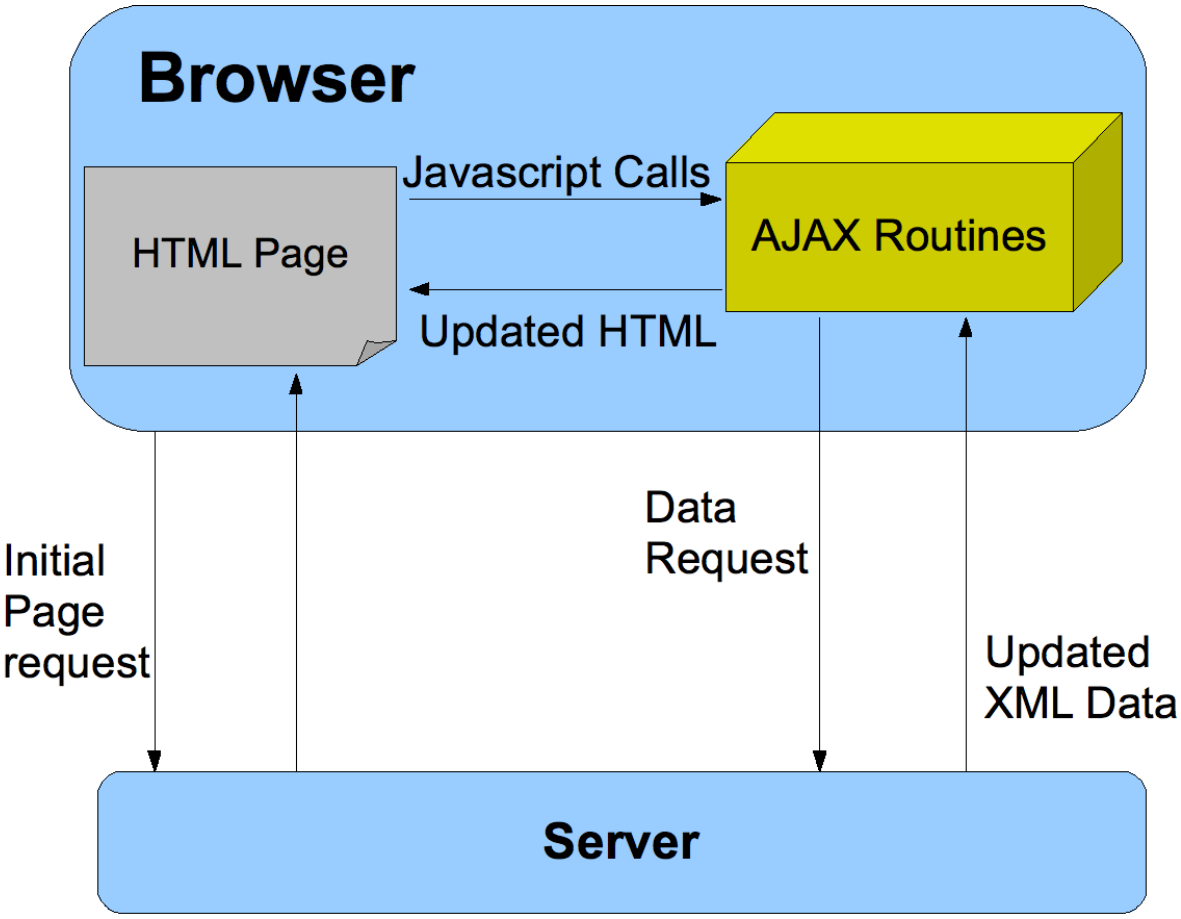


Image 5.5 AJAX workflow

The Ajax technology is very useful in this application when the user adds, updates or removes elements on the page. For example, when the user adds a new project, the web page launches an event and then JavaScript calls an Ajax routine to a specify URL. The Web Server receives and processes the JSON request and the data persists in the database. It responds with a JSON and a state message for the user.

5.2.3. Page fragmentation management with Apache Tiles

Apache Tiles is a free open-sourced templating framework for modern Java applications. Based upon the Composite pattern it is built to simplify the development of user interfaces. For complex web sites it remains the easiest and most elegant way to work alongside any MVC technology.

Tiles allows authors to define page fragments which can be assembled into a complete page at runtime. These fragments, or tiles, can be used as simple includes in order

to reduce the duplication of common page elements or embedded within other tiles to develop a series of reusable templates. These templates streamline the development of a consistent look and feel across an entire application.

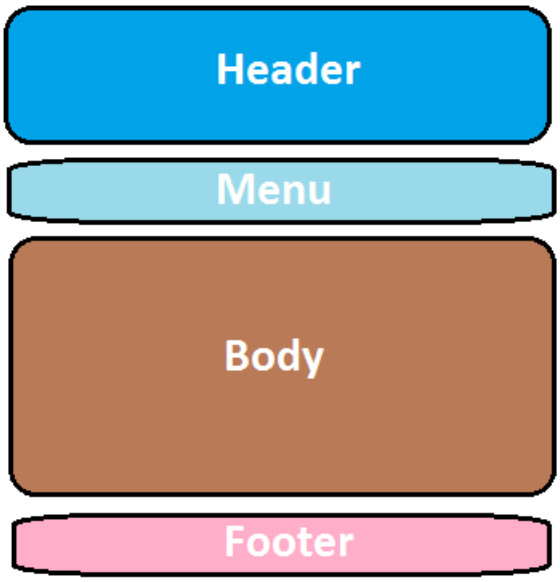


Image 5.6 HTML page organization

The configuration of the Apache Tiles is contained in a XML file. Here I defined each template and specified a path and a name for them. I have a unique template for the header, menu and footer so I don't need to duplicate any code. When I want to make a change in the code I need to apply the modifications only on the specified template.

5.2.4. Styling and appearance

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts. This

separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

This separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to display the web page differently depending on the screen size or device on which it is being viewed. Readers can also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author has specified.

Changes to the graphic design of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing markup in the documents.

The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C).

Bootstrap is a free and open-source front-end library for creating websites and web applications. It contains HTML and CSS based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. It aims to ease the development of dynamic websites and web applications. [3]

Bootstrap is a front end web framework, that is, an interface for the user, unlike the server-side code which resides on the "back end" or server.

Bootstrap is modular and consists essentially of a series of Less stylesheets that implement the various components of the toolkit. A stylesheet called bootstrap less includes the components stylesheets. Developers can adapt the Bootstrap file itself, selecting the components they wish to use in their project. [3]

Bootstrap features:

- Stylesheets - Bootstrap provides a set of stylesheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements.

- Re-usable components - In addition to the regular HTML elements, Bootstrap contains other commonly used interface elements. These include buttons with advanced features, labels, advanced typographic capabilities, thumbnails, warning messages and a progress bar. The components are implemented as CSS classes, which must be applied to certain HTML elements in a page.
- JavaScript components - Bootstrap comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields.

This project has a separated .css file for each page. Some elements can change their style by assigning another class that contains jQuery or JavaScript. Bootstrap it's very useful for elements like buttons, panels, badges, menus, messages, tables, progress bar or dialogs.

5.2.5. Drag and drop elements with jQuery UI

jQuery UI is a collection of GUI widgets, animated visual effects, and themes implemented with jQuery, Cascading Style Sheets, and HTML. Both jQuery and jQuery UI are free and open-source software distributed by the jQuery Foundation under the MIT License.

This library facilitates the drag and drop effect of the tasks by defining the lists involved. This effect can be used on the same list or between different lists. We can control the actions using a series of events from jQuery UI like:

- activate - This event is triggered when using connected lists, every connected list on drag start receives it.
- beforeStop - This event is triggered when sorting stops, but when the placeholder/helper is still available.
- change - This event is triggered during sorting, but only when the DOM position has changed.
- create - Triggered when the sortable is created.
- deactivate - This event is triggered when sorting was stopped, it is propagated to all possible connected lists.

- out - This event is triggered when a sortable item is moved away from a sortable list.
- over - This event is triggered when a sortable item is moved into a sortable list.
- receive - This event is triggered when an item from a connected sortable list has been dropped into another list. The latter is the event target.
- remove - This event is triggered when a sortable item from the list has been dropped into another. The former is the event target.
- sort - This event is triggered during sorting.
- start - This event is triggered when sorting starts.
- stop - This event is triggered when sorting has stopped.
- update - This event is triggered when the user stopped sorting and the DOM position has changed.

5.2.6. Integration of Summernote for advanced text edition

Summernote is a library that provides a user interface for editing the text area HTML element.

With this library the user can enter a brief description of the task using:

- text styles like bold, italic, underline, strikethrough.
- Superscript and subscript.
- Unordered and ordered lists.
- Paragraphs.

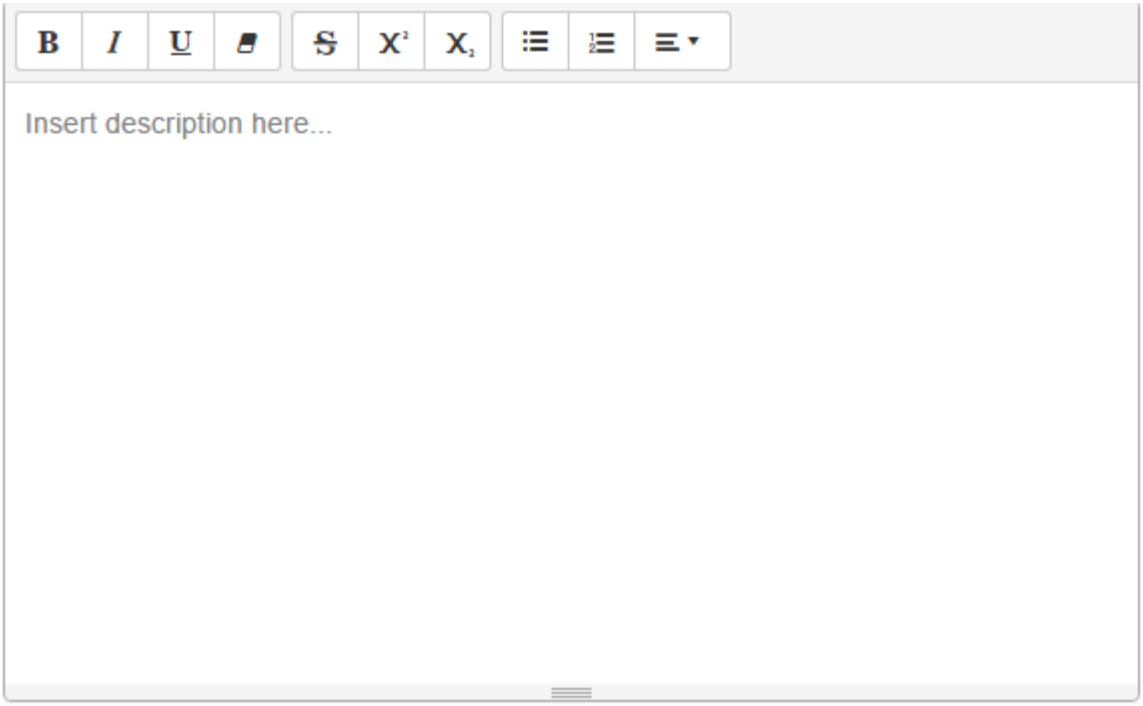


Image 5.7 Summernote edit textbox

The formatted text inserted by the user can be stored in the database like an HTML code and shown to the user in the same format where they access the tasks description.

5.2.7. Create diagrams with Cloud Flare library

Cloud Flare offers a free library for the burn down diagram construction. It is a library based on jQuery, Css and HTML technology. I have to populate the information of the diagram by providing:

- The list with Sprint days.
- The list with the ideal points.
- The list with burned points.

All this information is calculated on the server side.

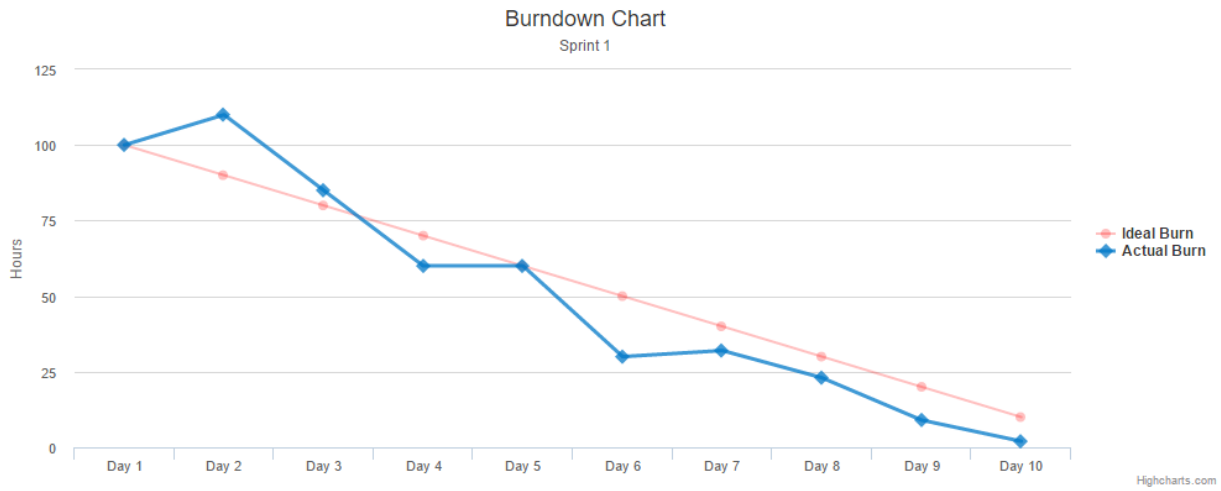


Image 5.8 Cloud Flare burndown chart

5.3. Back End features implementation

The Back End represent the data access layer and the business logic from the server side. It is related with the Controller and the Model of the MVC pattern (Subchapter 4.1.1. Model View Controller pattern) and it is hosted on the Amazon EC server, which is the application server tier (Subchapter 4.2.1. Three tier architecture).

5.3.1. Dependency injection using Spring Core

Central to the Spring Framework is its inversion of control (IoC) container, which provides a consistent means of configuring and managing Java objects using reflection. The container is responsible for managing object lifecycles of specific objects: creating these objects, calling their initialization methods, and configuring these objects by wiring them together. [1]

Image 5.9 Spring inversion of control

Objects created by the container are also called managed objects or beans. The container can be configured by loading XML files or detecting specific Java annotations on configuration classes. These data sources contain the bean definitions that provide the information required to create the beans.

Objects can be obtained by means of either dependency lookup or dependency injection. Dependency lookup is a pattern where a caller asks the container object for an object with a specific name or of a specific type. Dependency injection is a pattern where the container passes objects by name to other objects, via either constructors, properties, or factory methods.

In many cases it is not needed to use the container when using other parts of the Spring Framework, although using it will likely make an application easier to configure and customize. The Spring container provides a consistent mechanism to configure applications and integrates with almost all Java environments, from small-scale applications to large enterprise applications.

Dependency Injection offers at least the following advantages

- Loosely couple code.
- Separation of responsibility.
- Configuration and code is separate.

5.3.2. Create the Sign up and Sign in system with Spring Security

Spring Security is a customizable authentication and access service framework for server side Java-based enterprise software applications. The Spring security OAuth provides a method for making authenticated HTTP requests using a token - an identifier used to denote an access grant with specific scope, duration, and other attributes. [1]

Tokens are issued to third-party clients by an authorization server with the approval of the resource owner. Instead of sharing their credentials with the client, resource owners grant access by authenticating directly with the authorization server which in turn issues a token to the client. The client uses the token (and optional secret) to authenticate with the resource server and gain access.

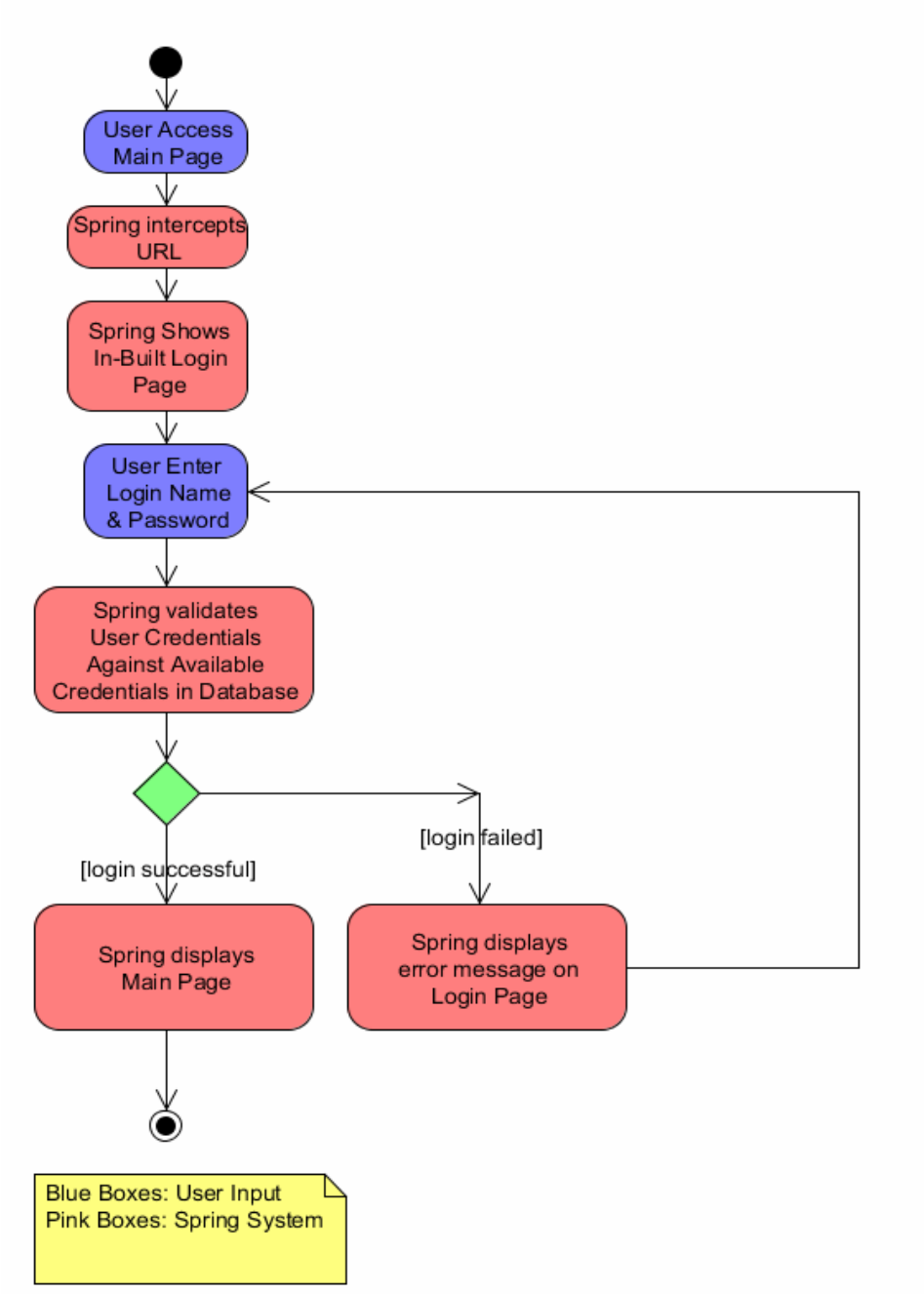


Image 5.10 Spring security log in system

Two major areas of application security are "authentication" and "authorization" (or "access-control"). These are the two main areas that Spring Security targets:

1. Authentication - is the process of establishing a principal[2] is who they claim to be.

[2] *A "principal" generally means a user, device or some other system which can perform an action in your application.*

2. Authorization - refers to the process of deciding whether a principal is allowed to perform an action within your application. To arrive at the point where an authorization decision is needed, the identity of the principal has already been established by the authentication process. These concepts are common, and not at all specific to Spring Security.

Authentication is the assurance that the user is actually the user he is claiming to be, for example, when the user logs into any application and gives his credentials, he authenticates himself. At the authentication level, spring supports various authentication models such as Http Basic authentication, Form Based authentication.

Authorization is the assurance that the user is allowed to access only those resources that he is authorized to use. For example, in a corporate application, there are some parts of an application where only admin has access and to some parts all the employees have access. These access rules are determined by the access rights given to each user of the system. At the authorization level, spring targets three main areas: authorizing web request, authorizing whether methods can be invoked and authorizing access to individual domain object instances.

Following are the some of the important facilities that Spring Security Framework provides to its users:

- User authentication and authorization.
- Role based authorization control.
- Easy to configure with database based authentication and authorization.
- Encrypted password.
- Form authentication.
- File bases user authentication and authorization

5.3.3. Hitting the database with Spring and JDBC

A database transaction is a sequence of actions that are treated as a single unit of work. These actions should either complete entirely or take no effect at all. Transaction management is an important part of and RDBMS oriented enterprise applications to ensure data integrity and consistency. The concept of transactions can be described with following four key properties described as ACID [1]:

- Atomicity: A transaction should be treated as a single unit of operation which means either the entire sequence of operations is successful or unsuccessful.

- Consistency: This represents the consistency of the referential integrity of the database, unique primary keys in tables etc.
- Isolation: There may be many transactions processing with the same data set at the same time, each transaction should be isolated from others to prevent data corruption.
- Durability: Once a transaction has completed, the results of this transaction have to be made permanent and cannot be erased from the database due to system failure.

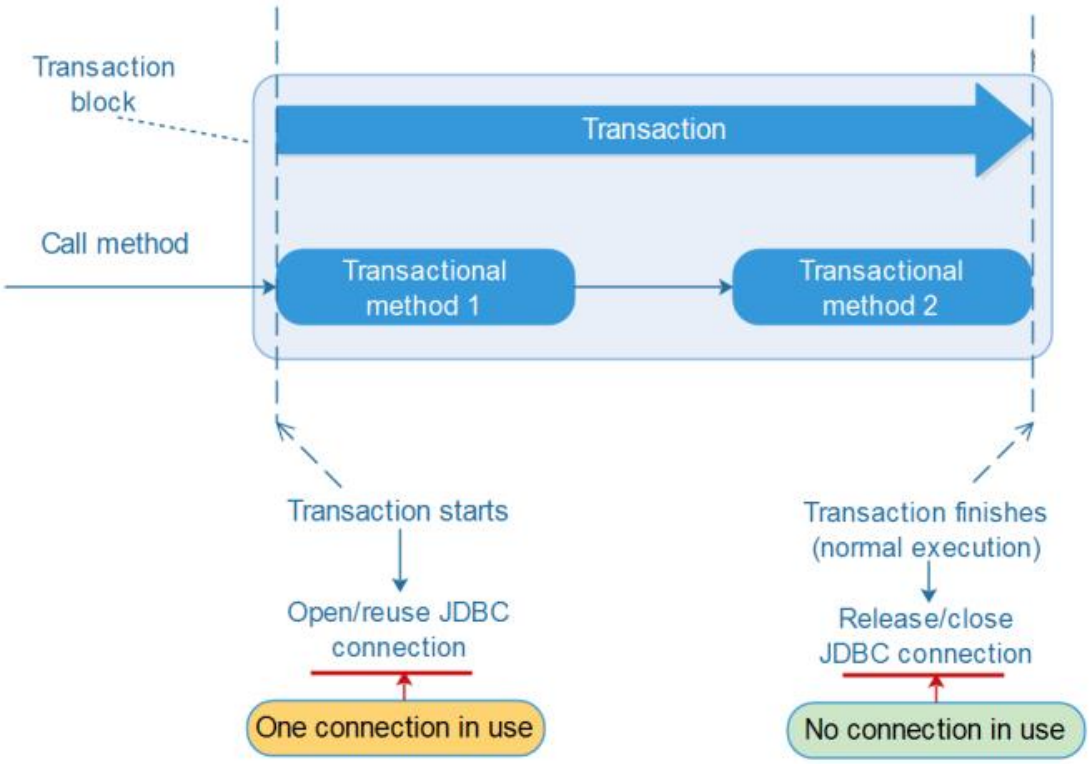


Image 5.11 JDBC transactions with Spring

The transactional annotation from Spring Framework is responsible to manage all database transactions. I can put the annotation in the Repository or in the Service layer. The advantage of managing the transactions in the Service layer is to not create a lot of performance overhead. The Service can use multiple Repositories and will create a unique transaction for the entire operation.

5.3.4. Monitoring using logs

Log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License.

Log4j has been ported to the C, C++, C#, Perl, Python, Ruby, and Eiffel languages. Log4j is highly configurable through external configuration files at runtime. It views the logging process in terms of levels of priorities and offers mechanisms to direct logging information to a great variety of destinations, such as a database, file, console, UNIX Syslog, etc.

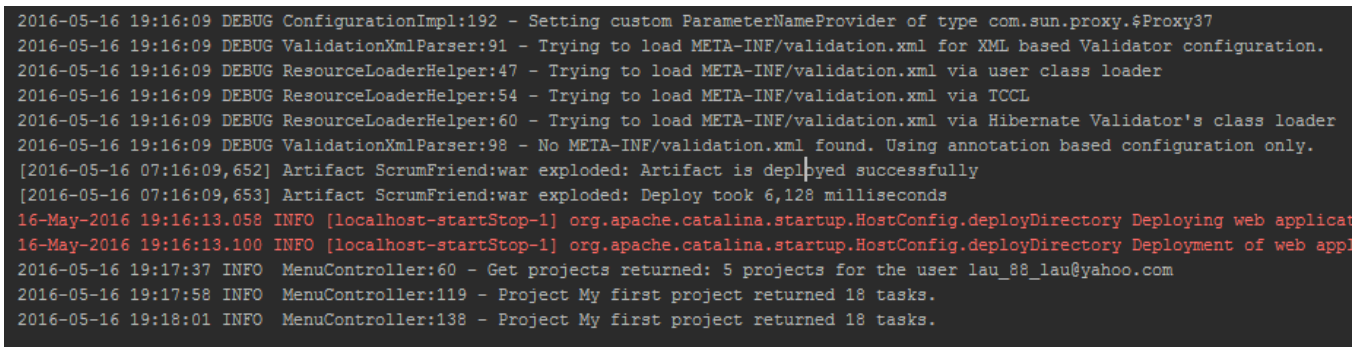


Image 5.12 Log example

Log4j has three main components:

- loggers: Responsible for capturing logging information.
- appenders: Responsible for publishing logging information to various preferred destinations.
- layouts: Responsible for formatting logging information in different styles.

The basic features of the Log4j are:

- It is thread-safe.
- It is optimized for speed.
- It is based on a named logger hierarchy.
- It supports multiple output appenders per logger.
- It supports internationalization.
- It is not restricted to a predefined set of facilities.
- Logging behavior can be set at runtime using a configuration file.
- It is designed to handle Java Exceptions from the start.

- It uses multiple levels, namely ALL, TRACE, DEBUG, INFO, WARN, ERROR and FATAL.
- The format of the log output can be easily changed by extending the Layout class.
- The target of the log output as well as the writing strategy can be altered by implementations of the Appender interface.
- It is fail-stop. However, although it certainly strives to ensure delivery, log4j does not guarantee that each log statement will be delivered to its destination.

5.4. Database implementation

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

The database is hosted on the database server (Subchapter 4.2.1. Three tier architecture) and follows the Entity Relationship model (Subchapter 4.1.3. Data model).

Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.

5.4.1. Infrastructure

Following the Entity Relationship model (Subchapter 4.1.3. Data model) I built the database infrastructure. Each entity represents a table with a primary key. I used foreign keys to reference other tables and an “on delete” statement which deletes the row once its reference is removed. With features like this I do not need to make these operations on the server side.

The default “current timestamp” is very useful to automatically fill fields like “created date” when a row is inserted.

5.4.2. Optimization policies

Looping queries - The most basic performance issues often will not be the fault of the database itself. One of the most common mistakes is to query in a loop without need. Most likely looped SELECT queries can be rewritten as a JOIN.

Picking only needed columns - It is common to see a wildcard used to pick all columns but is not efficient. Depending on the number of participating columns and their type (especially large types such as the TEXT variants), we could be selecting much more data from the database than we actually need. The query will take longer to return since it needs to transfer more data (from the hard-disk if it doesn't hit the cache) and it will take up more memory doing so.

Picking only the needed columns is a good general practice to use, and avoids those problems.

Filtering rows correctly and using indexes - My main goal is to select the smallest number of rows we need in the fastest way possible. I want to filter rows using indexes, and in general I want to avoid full table scans unless it is absolutely necessary (aside from special cases where it actually improves the performance). The MySQL manual provides great information regarding the optimization of the WHERE clause.

Filtering conditions include the WHERE, ON (for joins) and HAVING clauses. I want those clauses to hit indexes as much as possible; unless I select a very large number of rows index lookup is much faster than a full table scan. Those clauses should be used alongside with the LIMIT clause in case it is relevant to filter the amount of rows / data returned by the query. The LIMIT clause itself can lead to important optimizations for queries if used correctly.

Since my goal is to hit indexes with the WHERE clause, an important rule would be to avoid using calculations. When the filtering condition has to be calculated for each row, the WHERE clause cannot use an index.

Indexing correctly - MySQL can use one index per table alias in a query, so I need to plan our indexes to maximize their effectiveness. Using more indexes than necessary can have negative effects because it could slow down the operation of INSERT and UPDATE queries, while taking up more memory. Some indexes can even slow down performance depending on their selectivity.

Caching - MySQL has an internal query cache that caches results from frequently running queries if it meets certain requirements. If our queries are cached by MySQL (this can be verified by running the queries several times), there is no need to cache it - we just need to be aware that a MySQL service restart could cause a noticeable slow down while the cache is being primed again.

Common options include caching to disk (files) or caching to memory (using solutions such as mem cache or APC). Another form of caching is to the database - by de-normalizing the schema to store data that is the result of expensive to run queries.

5.4.3. Data integrity

Use database constraints whenever possible. There are two main reasons why using database constraints is a preferred way of enforcing data integrity. [10]

- First, constraints are inherent to the database engine and so use less system resources to perform their dedicated tasks. We resort to external user-defined integrity enforcement only if constraints are not sufficient to do the job properly.
- Second, database constraints are always checked by the database engine before insert, update, or delete operations. Invalid operations are cancelled before the operation is undertaken. So they are more reliable and robust for enforcing data integrity.

Data integrity is enforced by database constraints. Database Constraints are declarative integrity rules of defining table structures. They include the following 7 constraint types [10]:

1. Data type constrain - This defines the type of data, data length, and a few other attributes which are specifically associated with the type of data in a column.
2. Default constraint - This defines what value the column should use when no value has been supplied explicitly when inserting a record in the table.
3. Nullability constraint - This defines if a column is NOT NULL or it allows NULL values to be stored in it.
4. Primary key constraint - This is the unique identifier of the table. Each row must have a distinct value. The primary key can be either a sequentially incremented integer number or a natural selection of data that represents what is happening in the real world (e.g. Social Security Number). NULL values are not allowed in primary key values.
5. Unique constraint - This defines that the values in a column must be unique and no duplicates should be stored. Sometimes the data in a column must be unique even though the column does not act as Primary Key of the table.

6. Foreign key constraint - This defines how referential integrity is enforced between two tables.
7. Check constraint - This defines a validation rule for the data values in a column so it is a user-defined data integrity constraint. This rule is defined by the user when designing the column in a table. Not every database engine supports check constraints.

6. TEST

The software is tested at different levels. Initially, the individual units are tested arid once they are tested, they are integrated and checked for interfaces established between them. After this, the entire software is tested to ensure that the output produced is according to user requirements. There are four levels of software testing, namely, unit testing, integration testing, system testing, and acceptance testing.

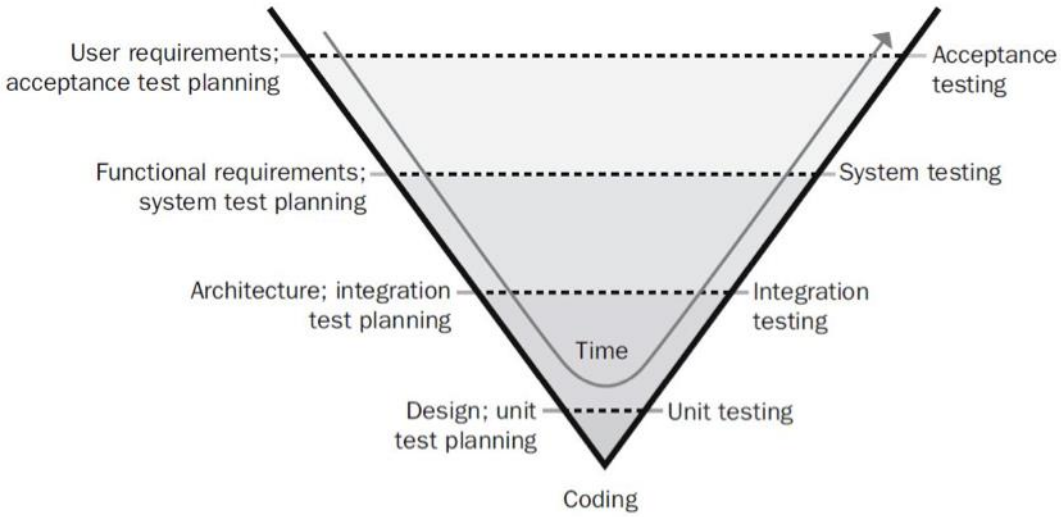


Image 6.1 Levels of software testing

Source: http://ecomputernotes.com/

6.1. Unit testing

Unit testing is performed to test the individual units of software. Since the software comprises various units/modules, detecting errors in these units is simple and consumes less time, as they are small in size. However, it is possible that the outputs produced by one unit become input for another unit. Hence, if incorrect output produced by one unit is provided as input to the second unit then it also produces wrong output. If this process is not corrected, the entire software may produce unexpected outputs. To avoid this, all the units in the software are tested independently using unit testing. [11]

Unit testing is not just performed once during the software development, but repeated whenever the software is modified or used in a new environment. Some other points noted about unit testing are listed below.

- Each unit is tested separately regardless of other units of software.

- The developers themselves perform this testing.
- The methods of white box testing are used in this testing.

Unit testing is used to verify the code produced during software coding and is responsible for assessing the correctness of a particular unit of source code. In addition, unit testing performs the following functions:

- It tests all control paths to uncover maximum errors that occur during the execution of conditions present in the unit being tested.
- It ensures that all statements in the unit have been executed at least once.
- It tests data structures (like stacks, queues) that represent relationships among individual data elements.
- It checks the range of inputs given to units. This is because every input range has a maximum and minimum value and the input given should be within the range of these values.
- It ensures that the data entered in variables is of the same data type as defined in the unit.
- It checks all arithmetic calculations present in the unit with all possible combinations of input values.

Scrum Friend application has implemented unit test using the JUnit framework. The unit test was implemented during the development process. JUnit is included in the Maven's POM file because it is an external dependency. Every time the project is build, Maven automatically launches all unit tests. If a test fails, the build process is interrupted. All the test classes are located in the test/main/java folder and have the same package name like the tested classes.

6.2. Integration testing

Once unit testing is complete, integration testing begins. In integration testing, the units validated during unit testing are combined to form a subsystem. The integration testing is aimed at ensuring that all the modules work properly as per the user requirements when they are put together.

The objective of integration testing is to take all the tested individual modules, integrate them, test them again, and develop the software, which is according to design specifications. Some other points that are noted about integration testing are listed below.[11]

- It ensures all modules work together properly and transfer accurate data across their interfaces.
- It is performed with the intention to uncover errors that lie in the interfaces among the integrated components.
- It tests those components that are new or have been modified or affected due to any change.

In the development process after each task was implemented I did an integration test to ensure the correct behavior of the application and to avoid conflicts with other features. The test was done manually using the deployed version from Amazon Web Service.

6.3. System testing

Software is integrated with other elements such as hardware, people, and database to form a computer-based system. This system is then checked for errors using system testing. IEEE defines system testing as "a testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirement." [11]

In system testing, the system is tested against non-functional requirements such as accuracy, reliability, and speed. The main purpose is to validate and verify the functional design specifications and to check how integrated modules work together. The system testing also evaluates the system's interfaces to other applications and utilities as well as the operating environment.

During system testing, associations between objects, control and infrastructure, and the compatibility of the earlier released software versions with new versions are tested. System testing also tests some properties of the developed software, which are essential for users. These properties are listed below.

- Usable: Verifies that the developed software is easy to use and is understandable. This test was made with real users like: product owners, project managers, testers, developers and it is related to the UX prototype. This test was several times, after each page implementation.
- Secure: Verifies that access to important or sensitive data is restricted even for those individuals who have authority to use the software. I tested the security of

the application by trying to access the private data from another user (Cross site scripting).

- Compatible: Verifies that the developed software works correctly in conjunction with the existing data, software and procedures. I made the compatibility test using the most popular browsers like: Chrome, Safari, Opera, Mozilla. Internet Explorer.
- Documented: Verifies that manuals that give information about the developed software are complete, accurate and understandable. The documentation of my application includes Javadoc and it is available in English for all classes and public methods.
- Recoverable: Verifies that there are adequate methods for recovery in case of failure.

6.4. Acceptance testing

Acceptance testing is performed to ensure that the functional, behavioral, and performance requirements of the software are met. IEEE defines acceptance testing as a "formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system." [11]

During acceptance testing, the software is tested and evaluated by a group of users either at the developer's site or user's site. This enables the users to test the software themselves and analyze whether it is meeting their requirements. To perform acceptance testing, a predetermined set of data is given to the software as input. It is important to know the expected output before performing acceptance testing so that outputs produced by the software as a result of testing can be compared with them. Based on the results of tests, users decide whether to accept or reject the software. That is, if both outputs (expected and produced) match, the software is considered to be correct and is accepted; otherwise, it is rejected.

The acceptance test was the final test for the Scrum Friend application. With this type of test, I verified that the project meets all the previous functional requirements from the Use cases diagram. Also, the acceptance test was made after each task to check all the acceptance conditions set out initially.

7. RESULTS

This chapter includes information about various results and statistics, after the development process was completed and the application was ready for the production environment.

7.1. Final product review and user experience

The final product is a professional application with minimalist design, great response time and modular code, using the latest technologies. The application can be easily extended with new functionalities and the database can be replaced with a NoSQL.

The product has been tested by real users. They had a very good opinion about it and were impressed by these features:

- Very easy to use.
- All the elements are visible and easy to find.
- The sprint progress bar is very useful.
- The burndown charts are clear and very explicit.
- You can manage multiple projects.

Things that can be improved in the future:

- Selecting the project anywhere on the web.
- Editing options for all elements.
- Confirmation messages for important actions.

7.2. Software metrics and measures

In terms of size metrics, I calculated the lines of code using the Statistic plugin for the IntelliJ IDE. This metrics do not include the external libraries I used. As we can see, Java represents almost 50% of the total code and is running on the server side. The other 50% resulted from the CSS, JS and HTML (JSP) files that are downloaded in the client side and executed with the browser.

Table 7-1 Files length statistic

FILE TYPE	EXTENSION	SIZE (kb)	LINES OF CODE
Cascade Style Sheet	.css	40	690
Java classes	.java	152	5601
JavaScript	.js	307	2209
Java Server Faces	.jsp	54	1140
Java properties	.properties	1	25
XML configuration	.xml	97	1521
Database script	.sql	4	100
TOTAL		657	11293

With an advanced analysis of Java code, we can see more clearly the difference between code formatting, Javadoc and programmed lines.

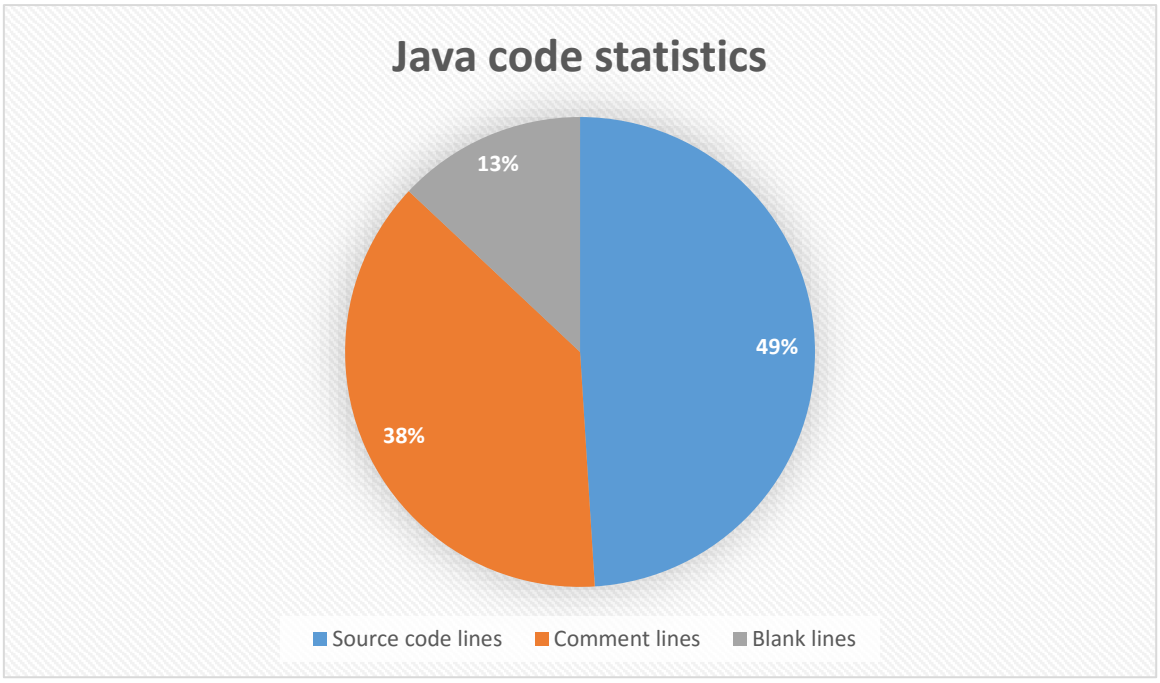


Image 7.1 Java code statistic chart

GitHub offers us a statistics of the commit history. As we can see, most commits were carried out in March and April when the Product Backlog and Sprints tab were constructed.

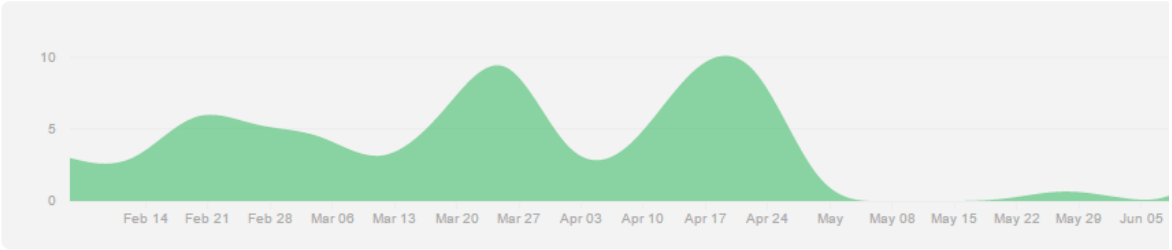


Image 7.2 GitHub commits history

GitHub also provides us the number of commits per day. In general, the commits were made during Saturdays and Sundays.

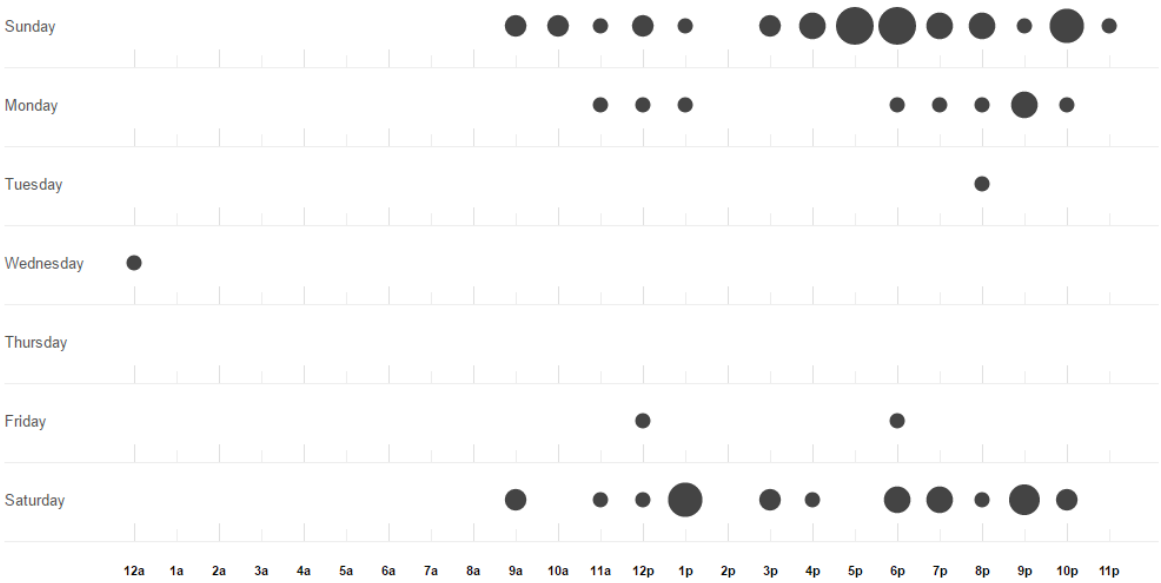


Image 7.3 GitHub distributed daily commits

All development tools used were free because I used student licenses. The MySQL database is open source and all Amazon Web Services servers and applications belong to the free tier and I can host the application there for 1 year without any charges. The external libraries were also free to use.

The books and the online material used for the documentation were free and are mentioned in the References chapter.

The time spent is detailed on the following table:

Table 7-2 Time spent statistic

Module	**Hours**	**Percentage %**
Requirements	24	6.19
Analysis	32	8.24
Development and testing	200	51.54
Documentation	96	24.75
Presentation	36	9.28
Total time spent	388	100

As we can see for this project were required 388 hours or 49 working days if we count that I worked 8 hours per day. To reach these values, a good planning was needed together with previous knowledge and experience with complex projects.

8. CONCLUSIONS

After three and a half months of work I have fulfilled the initial objectives. Scrum Friend is a complex application and ready to be published in production environment. To achieve this, a good planning, previous requirements analysis, modern tools and technologies, and a qualitative test system were needed.

During the development process I was impressed by several technologies. With the Spring frameworks I got a modular code and well structured, Bootstrap offers a lot of elements that can be used to obtain a professional interface, JSP makes your work very easy when you have to deal with list of elements and Amazon Web Services facilitate a complete console where you can configure and monitor all the server's activity.

The user feedback was good and that promises a good evolution in the production environment. They proposed some improvements that could be implemented in the second iteration of the development process.

Next, I will present a list of new features that were proposed for future versions:

- Registration with email confirmation. This feature would increase the security level.
- History of actions. Users could visualize the history of actions and who made the changes.
- Rollback features and edit options. It could avoid errors due to user actions.
- Kanban view of the projects.
- Implementation of Google Analytics to make statistics about the user's actions.

9. REFERENCES

Books:

[1] Walls, C. (2015). *Spring in action Fourth Edition*. Manning: New York.

[2] Wittig, M.; Wittig A. (2015). *Amazon Web Services in Action.* Manning: New York.

[3] Bhaumik S. (2015). *Bootstrap Essentials.* Packt Publishing: Birmingham.

[4] Stellman A.; Greene J. (2014). *Learning Agile.* O'Reilly Media: California.

[5] Schwartz B.; Zaitsev P.; Tkachenko V. (2012). *High Performance MySQL 3rd Edition.* O'Reilly Media: California.

[6] Khorana R. (2011). *Software Engineering (WBUT) 2nd Edition.* Vikus Publishing House: US.

Websites:

[7] *Scrum Alliance. R*etrieved from: https://www.scrumalliance.org/why-scrum/scrum-guide

[8] *AWS Documentation.* Retrieved from: https://aws.amazon.com/documentation/

[9] *Spring – MVC Framework.* Retrieved from: http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

[10] *Data integrity.* Retrieved from: http://www.geeksengine.com/

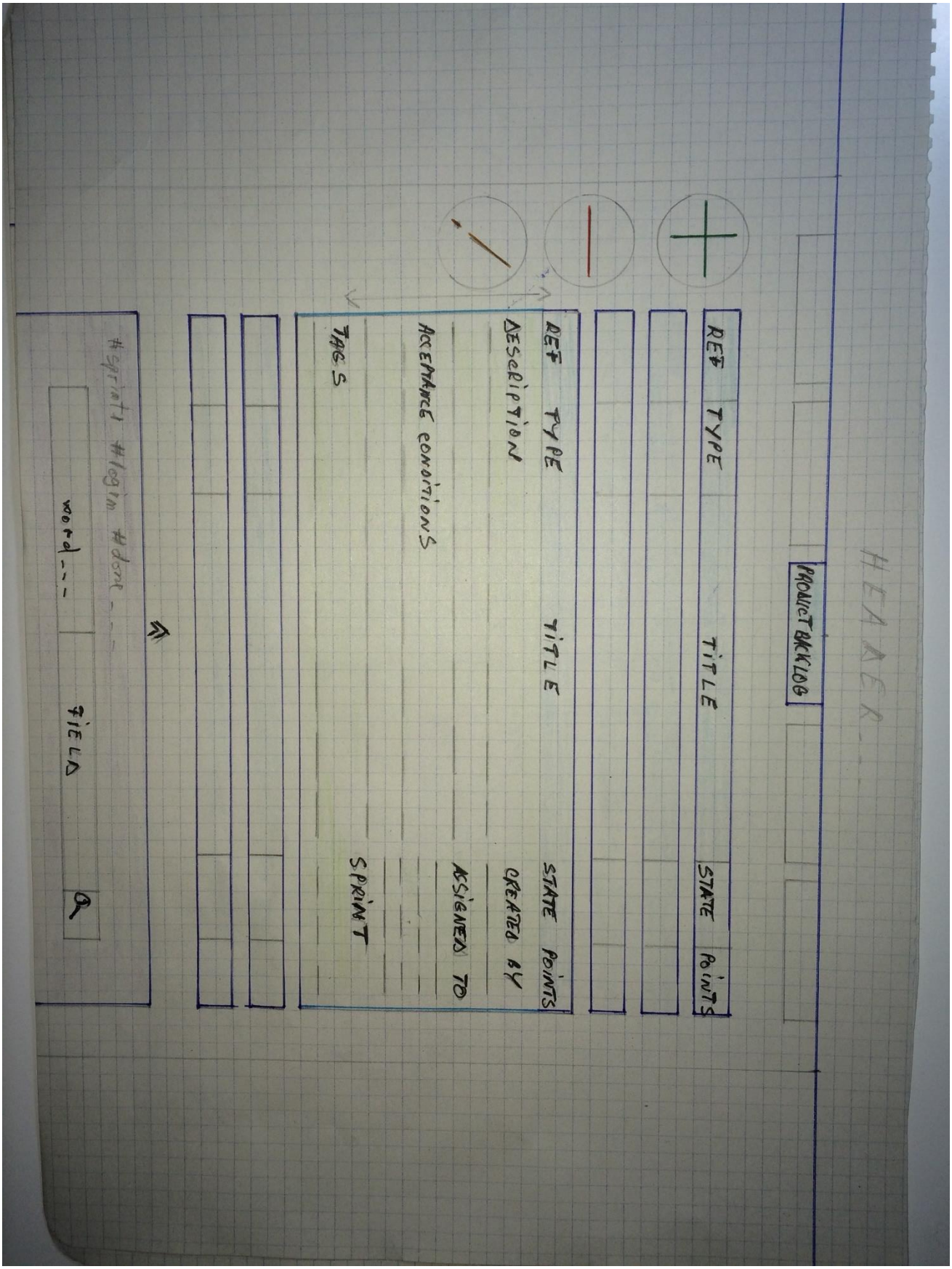
[11] *Levels of software testing*. Retrieved from: http://ecomputernotes.com/

[12] *Using a Three-Tier Architecture Model.* Retrieved from: https://msdn.microsoft.com/en-s/library/windows/desktop/ms685068(v=vs.85).aspx

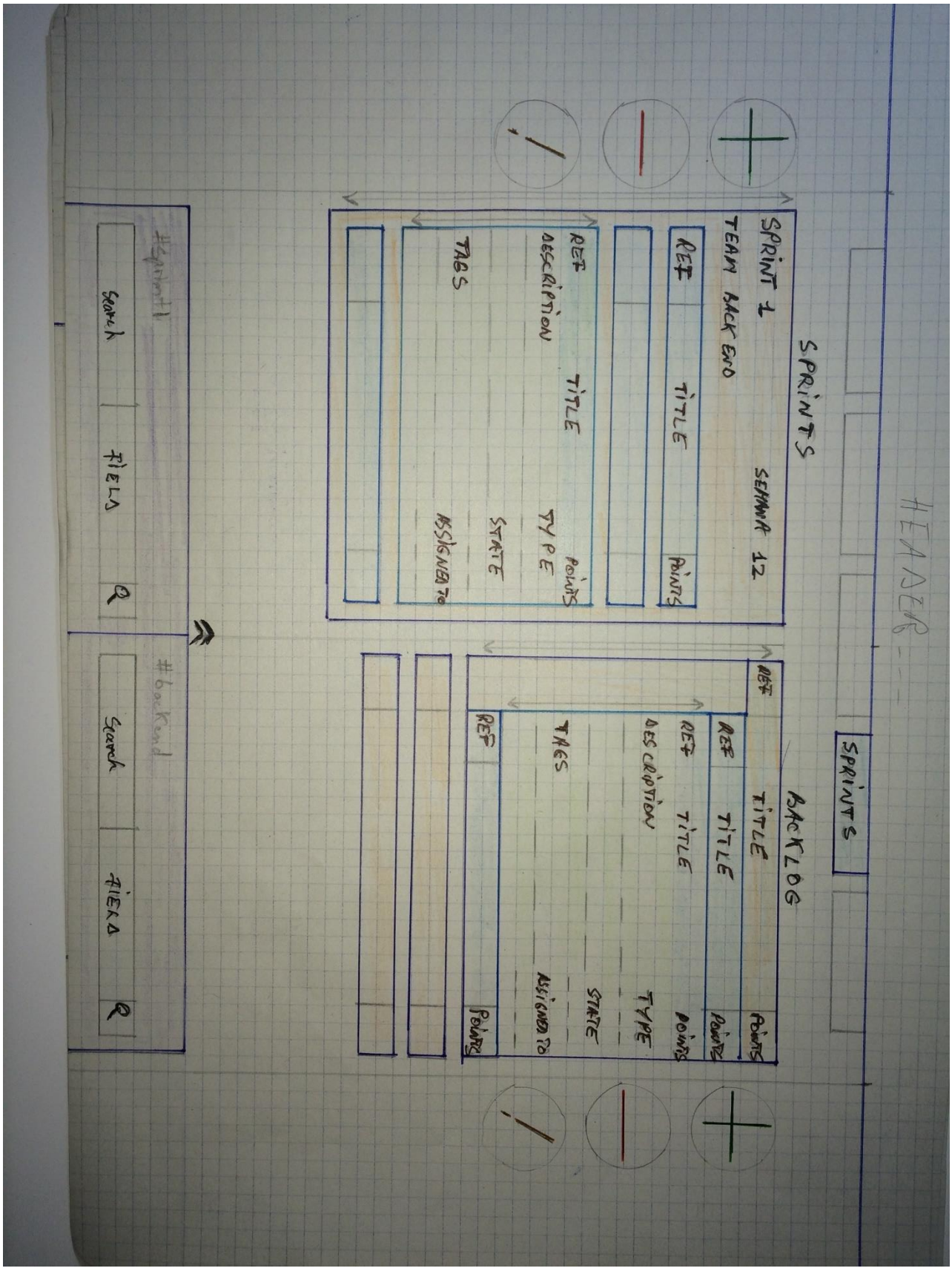
[13] *Layered Application Guidelines.* Retrieved from: https://msdn.microsoft.com/en-us/library/ee658109.aspx

10.ANNEXES

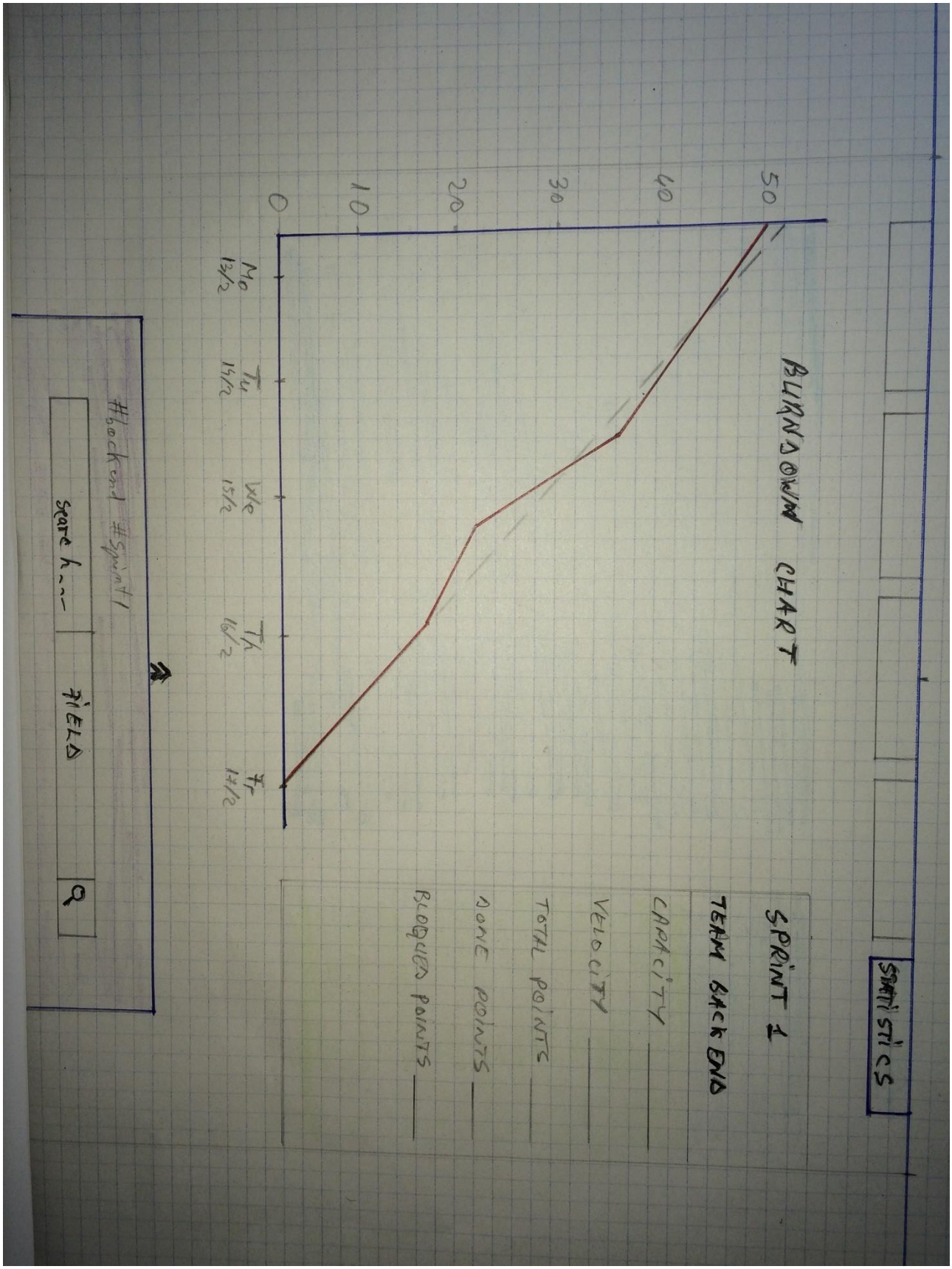
Annex 1: Backlog wireframe



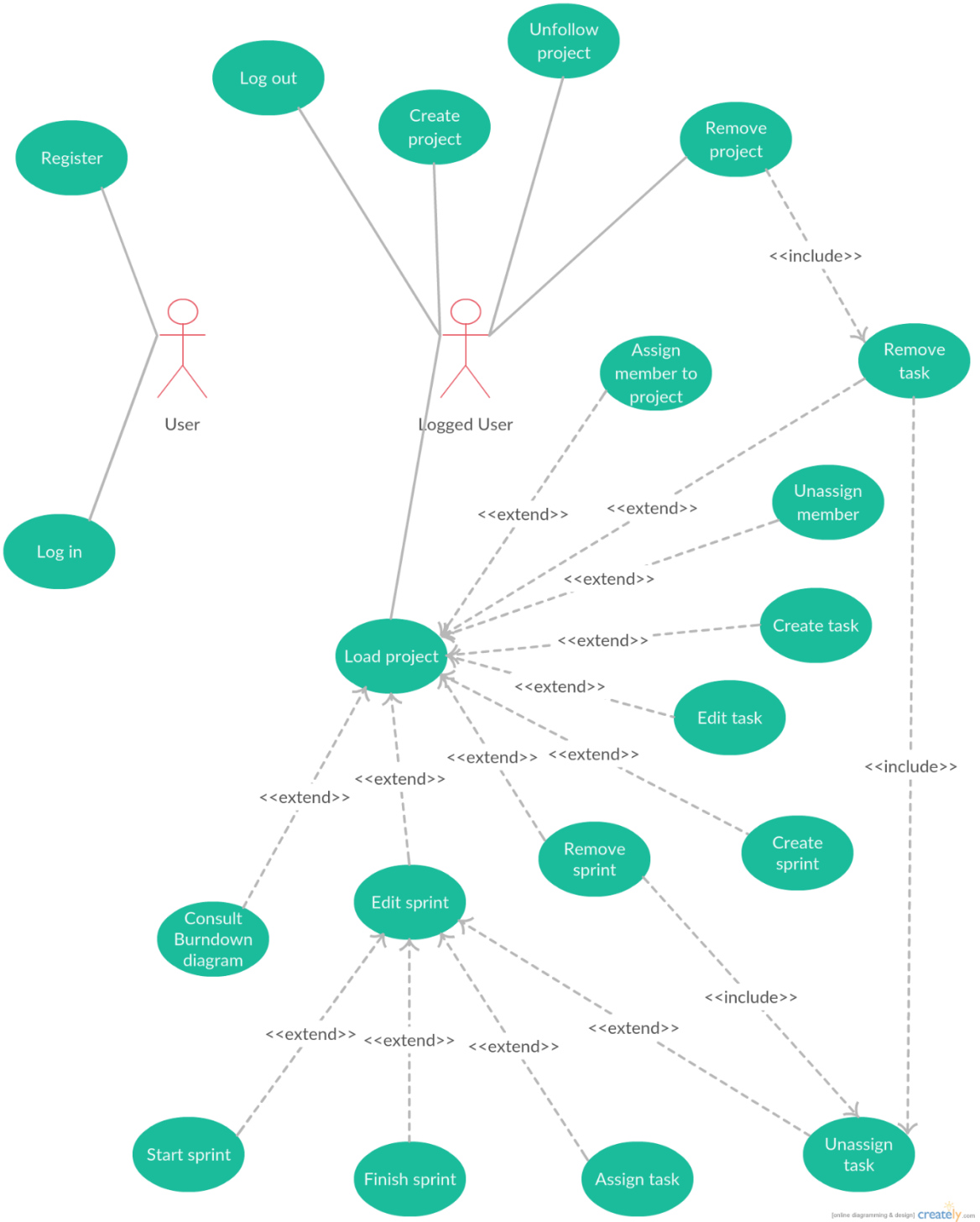
Annex 2: Sprints wireframe



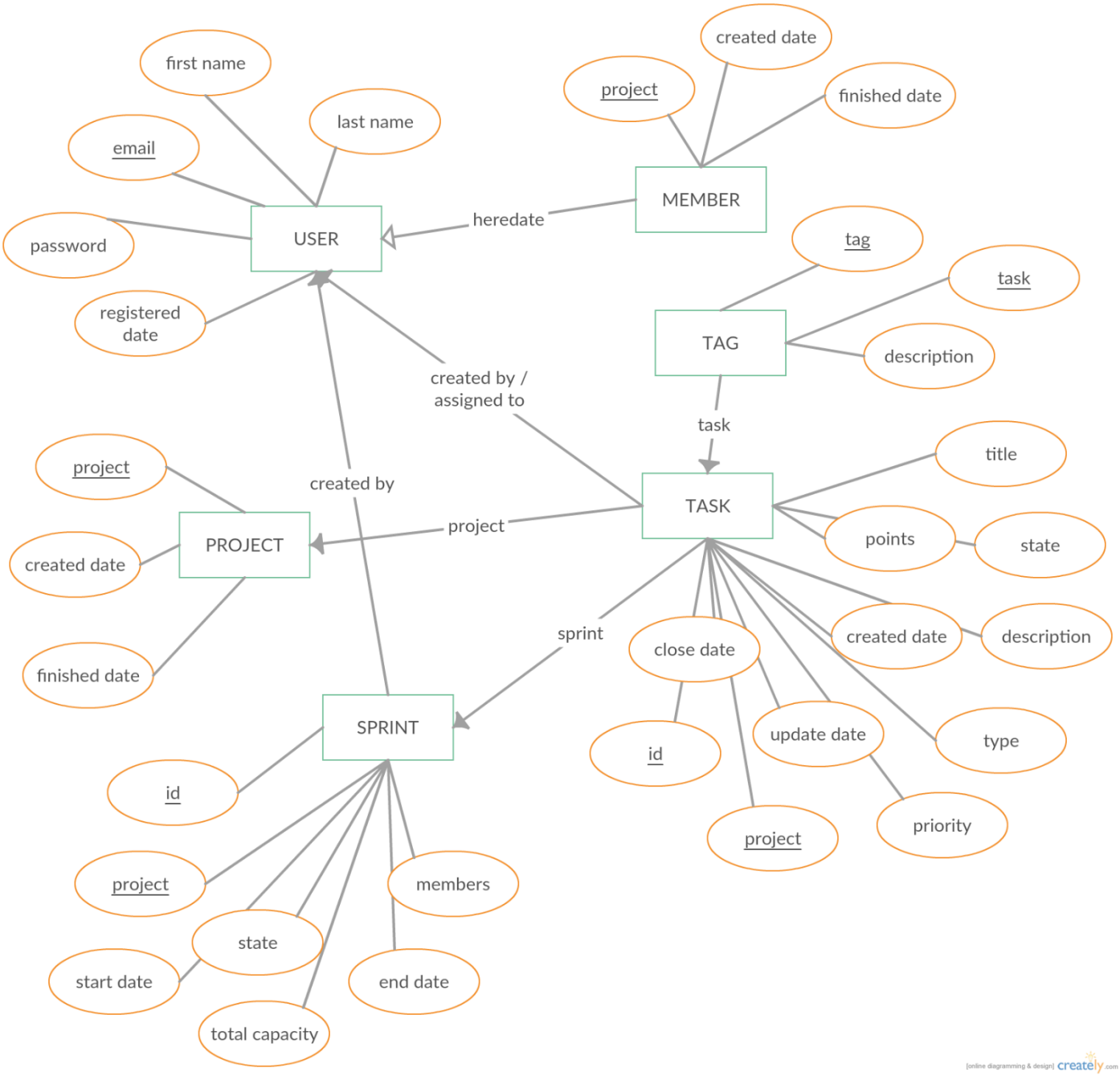
Annex 3: Statistics wireframe



Annex 4: Use cases diagram (UML)



Annex 5: Database Entity relationship (UML)



Annex 6: Gant diagram