ACME-SF

G1.007

Previous knowledge about architecture

14 February, 2024

Cover

Repository: https://github.com/Pablo-Caballero-Maria/Acme-One-24.1.0-C1.07

Student #1

ID: 31878881F
UVUS: pabcabmar3
Name: Caballero María, Pablo
Roles: manager, developer

Student #2

ID Number:49034820Q
UVUS: mararnmon
Name: Arnáiz Montero, Marco Antonio
Roles: developer, operator

Student #3

ID Number: 77865211E
UVUS: alfalolan
Name: Alonso Lanzarán, Alfonso Luis
Roles: developer, tester

Student #4

ID Number: 53932912M
UVUS: albsanmim
Name: Sánchez Mimbrero, Alberto
Roles: developer

Student #5

ID Number: 48123111G
UVUS: juagarcar4
Name: Garcia Carballo, Juan
Roles: developer

Table of contents

Executive summary

Intentionally blank.

Revision table

Number	Date(dd/mm/yyyy)	Description
1.0	14/02/2024	Document done in its entirety, reviewed by peers. No major errors were found.

Introduction

The purpose of this document is to explain the previous knowledge of the group members about the architecture of a web information system. This knowledge has been learned through previous subjects such as AISS, ISSI I and ISSI II, DP1 and IR, and involves a wide range of both theoretical and practical knowledge about web information systems architecture, patterns, and technologies.

Contents

We know different architectural patterns such as: layers, pipelines, model-view-controller, central repository and their main characteristics and operation. As an example we will use a diagram of a web system with a layered architecture.

In the case of a layered architecture such as the one in our diagram, we know that it is divided into 3 layers: presentation, application and database. These layers can only communicate between adjacent ones using methods that we will delve into later. Each layer is divided into components with specific functions that will be detailed in the following paragraphs. This allows the simultaneous development of layers, since each one should be

independent of the others, so that each one offers methods to the layer above it, without the need for it to know the technical details of its implementation.
In the case of a layered architecture such as the one in our diagram, we know that it is divided into 3 layers: presentation, application and database. These layers can only communicate between adjacent ones using methods that we will delve into later; each layer is divided into components with specific functions that will be detailed in the following paragraphs. This allows the simultaneous development of layers, since each one should be independent of the others, so that each one offers methods to the layer above it, without needing to know the technical details of its implementation.

The presentation layer is where the application is shown to the user and through which he interacts with it, it is rendered in the user's browser and communicates with the application layer through HTTP requests. In previous subjects we have been taught how engines such as AJAX (Asynchronous Javascript And XHR) allow the presentation layer, when rendered in the browser, to dynamically update its components through Single Page Application or SPA technology. In addition, we know how browsers create a Document Object Model or DOM through which they are able to map them hierarchically into HTML components with which the user interacts.

At the application layer we distinguish between the following components: controller, service and repository. The controller is responsible for receiving HTTP requests and managing them (each method has an endpoint, a method, and, optionally, query parameters), the service contains the business logic, for which it uses entities that are created with the information from the database, the service accesses this data through the repository not directly from the database. Furthermore, this is where web information systems must implement the exceptions that are responsible for applying said business logic.

The application must be deployed on a server, in the diagram it is tomcat.

Finally, the application's persistent information is stored in the database layer. We mainly know SQL databases and how to use them using tools like mariaDB or Dbeaver. Likewise, we know how the Java Persistence API or JPA interacts with the database, and how tools like Hibernate or Sequelize can convert database rows to programming objects. Additionally, in some subject we were introduced theoretically that there are other database schemes, such as those based on “graphs” or “documents”.

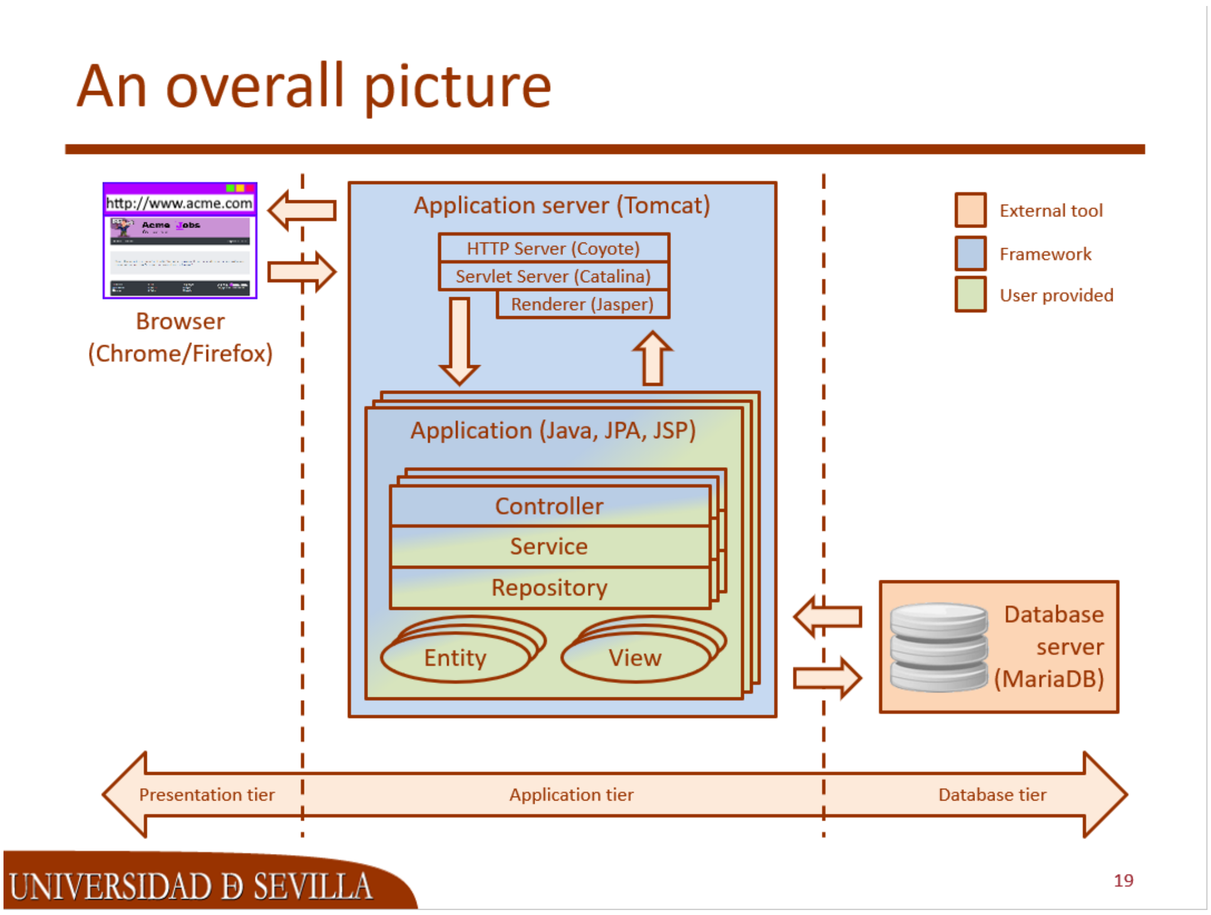

Illustration 1Software architecture diagram

Conclusions

Bibliography

Intentionally blank