**Project Order**

This document describes the Project order for the *Zoo Manager (ZM).* It is based on the [Project Proposal](https://reset.inso.tuwien.ac.at/repo/2020ss-SEPM-group/ss20_sepm_qse_07/-/wikis/Project-Proposal).

It describes the requirements, as well as the project management artefacts, which together, form the basis of the contract with the client.

**Project Name & Development team**

See Project Name & Development team in the [Project Proposal](https://reset.inso.tuwien.ac.at/repo/2020ss-SEPM-group/ss20_sepm_qse_07/-/wikis/Project-Proposal)

**Initial situation & Existing Similar Products**

See Initial situation in the [Project Proposal](https://reset.inso.tuwien.ac.at/repo/2020ss-SEPM-group/ss20_sepm_qse_07/-/wikis/Project-Proposal)

**Project Description**

See Project Description in the [Project Proposal](https://reset.inso.tuwien.ac.at/repo/2020ss-SEPM-group/ss20_sepm_qse_07/-/wikis/Project-Proposal)

**Target Audience**

The application is intended to facilitate and speed up the organization of a zoo.

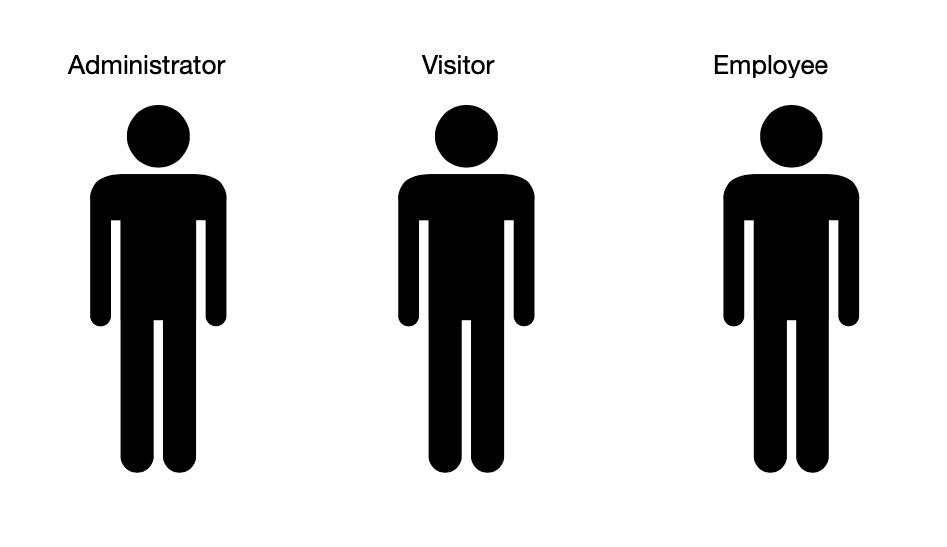
The target group are the people who run the zoo, as well as the employees of the zoo.

On the other hand, the application allows everyone, both employees and visitors, to view general zoo information.

Administrator - The administrator is in charge of creating and managing the user accounts, animals and enclosures. His job is to assign animals to employees. The administrator has access to the whole systems functionality. He has an overview of all employees, animals/enclosures and tasks bounded to them.

Employee - Employee is responsible for creating, assigning and managing tasks bound to the animal/enclosure he was assigned by the administrator. An employee has an overview of the tasks he has been assigned and the tasks he created.

Visitor - Visitor can get general information about the zoo and all current and upcoming events in the zoo.



**Functional Requirements**

To see a description of the features see Feature-list in the [Project Proposal](https://reset.inso.tuwien.ac.at/repo/2020ss-SEPM-group/ss20_sepm_qse_07/-/wikis/Project-Proposal)

**Use Case Overview**

1. Task Management
2. Animal Management
3. Event Management
4. Enclosure Management
5. General Information
6. Login/Authentication

***Iceberg List***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Feature, Actuator | Use case | Priority | Cost | Version |
| 1.1 | Task, Admin/Worker | Create tasks for assigned animals and their enclosures | H | 9 | V1 |
| 1.2 | Task, Worker | Create tasks of enclosures by assigned animals | H | 6 | V1 |
| 1.3 | Task, Worker | Assigning tasks as worker | H | 7 | V1 |
| 1.4 | Task, Admin/Worker | Edit tasks | H | 2 | V2 |
| 1.5 | Task, Worker | Mark tasks as done | H | 3 | V1 |
| 1.6 | Task, Worker, Admin | Comment on Tasks | L | 7 | V3 |
| 1.7 | Task, Worker | View of assigned tasks | H | 6 | V1 |
| 1.8 | Task, Admin, Worker | Assign task automatically | H | 8 | V2 |
| 1.9 | Worker, Admin | Delete tasks | H | 3 | V1 |
| 1.10 | Task, Worker | Tasks order by priority(high first) | M | 6 | V1 |
| 1.11 | Task, Worker | See unfinished personal tasks in calendar | L | 8 | V3 |
| 2.1 | Animal, Admin | Create animals | H | 8 | V1 |
| 2.2 | Animal, Admin | Edit animals | H | 6 | V2 |
| 2.3 | Admin, Admin | Assigning animal to worker | H | 4 | V1 |
| 2.4 | Animal, Admin/Worker | See detailed information about animals | M | 3 | V1 |
| 2.5 | Animal, Admin/Worker | Search for the animals | L | 3 | V2 |
| 2.7 | Animal, Admin | Remove animals | H | 5 | V1 |
| 2.8 | Animal, Worker | Look up unfinished tasks of animals | L | 6 | V3 |
| 2.9 | Animal, Admin | Look up all existing Animals | H | 3 | V1 |
| 3.1 | Event, Worker | Create event | L | 7 | V3 |
| 3.2 | Event, Worker | Assign event to worker | L | 6 | V3 |
| 3.3 | Event, Worker | See detailed information on events | L | 3 | V3 |
| 3.4 | Event, Visitor | See public information on events | L | 3 | V3 |
| 3.5 | Event, Visitor | Search for Events | L | 5 | V3 |
| 4.1 | Enclosure, Admin | Create enclosure | H | 8 | V1 |
| 4.2 | Enclosure, Admin | Edit enclosures | H | 6 | V2 |
| 4.3 | Enclosure, Admin | Assign animals to enclosures | H | 5 | V1 |
| 4.4 | Enclosure, Admin | Delete enclosures | H | 2 | V1 |
| 4.5 | Enclosure, Worker/Visitor | Access information about enclosures | L | 3 | V3 |
| 4.6 | Enclosure, Admin | Look up all existing Enclosures | H | 3 | V1 |
| 5.1 | Zoo information, Admin/Worker/Visitor | See information about the zoo | L | 2 | V3 |
| 5.2 | Zoo information, Admin | Edit zoo information | L | 3 | V3 |
| 6.1 | Registration/Login, Admin | Create admin accounts | M | 5 | V3 |
| 6.2 | Registration/Login, Admin/Worker | Create users | H | 10 | V1 |
| 6.3 | Registration/Login, Admin | Edit existing accounts | H | 6 | V2 |
| 6.4 | Registration/Login, Admin | Delete existing accounts | H | 7 | V1 |
| 6.5 | Registration/Login, Worker | Able to change password | H | 2 | V2 |
| 6.6 | Registration/Login, Admin/Worker | Logging in/out | H | 7 | V1 |
| 6.7 | User, Admin | Look up all existing Users | H | 3 | V1 |

**Domain Model**

See the Domain Model of the [Project Proposal](https://reset.inso.tuwien.ac.at/repo/2020ss-SEPM-group/ss20_sepm_qse_07/-/wikis/Project-Proposal).

**Work Structure & Rough Plan (WBS)**

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Assignment | Start-Date | End-Date |
| 1.1 | Kick-Off | 16.04.2020 | 16.04.2020 |
| 1.1.1 | Develop project idea | 16.04.2020 | 20.04.2020 |
| 1.1.2 | Jour-Fixe | 20.04.2020 | 20.04.2020 |
| 1.1.3 | Project Proposal | 20.04.2020 | 23.04.2020 |
| 1.1.4 | Jour Fixe | 27.04.2020 | 27.04.2020 |
| MR-1 | Management Review 1 - Requirement Review (Project Order) | 04.05.2020 | 04.05.2020 |
| 1.2 | Jour Fixe | 04.05.2020 | 04.05.2020 |
| 1.2.1 | Finish Requirement Analysis |  |  |
| 1.2.2 | All use-cases with User Stories |  |  |
| 1.2.3 | Develop implementation structure |  |  |
| 1.2.4 | Assign User Stories |  |  |
| MS-2 | First System Check |  |  |
| 2.1 | Implementation Sprint 1 | 07.05.2020 | 16.05.2020 |
| 2.1.1 | Database & Test Data |  |  |
| 2.1.2 | DAO’s |  |  |
| 2.1.2 | Service |  |  |
| 2.1.3 | REST |  |  |
| 2.1.4 | GUI/Frontend |  |  |
| 2.1.5 | Angular-Service and Rest Client |  |  |
| 2.2 | Jour Fixe | 11.05.2020 | 11.05.2020 |
| 3.1 | Implementation Sprint 2 | 16.05.2020 | 25.05.2020 |
| IR-1 | Internal Review 1 | 18.05.2020 | 18.05.2020 |
| 3.2 | Jour Fixe | 18.05.2020 | 18.05.2020 |
| 3.3 | Jour Fixe | 25.05.2020 | 25.05.2020 |
| MR-2 | Management Review 2 | 25.05.2020 | 25.05.2020 |
| 4.1 | Implementation Sprint 3 | 25.05.2020 | 08.06.2020 |
| 4.2 | Jour Fixe | 01.06.2020 | 01.06.2020 |
| 4.3 | Jour Fixe | 08.06.2020 | 08.06.2020 |
| 5.1 | Implementation Sprint 4 | 08.06.2020 | 22.06.2020 |
| IR-2 | Internal Review 2 | 15.06.2020 | 15.06.2020 |
| 4.5 | Jour Fixe | 22.06.2020 | 22.06.2020 |
| MR-3 | 100% implemented, Project presentation | 22.06.2020 | 22.06.2020 |

## **Milestone Description**

* **MR-1: Project contract 04.05.2020**
  + Analysis of requirements and the environment
  + Project structuring and basic documentation (features and iceberg list, domain, project plan, milestones)
  + Work structuring (distribution of roles and responsibilities)
  + Definition of project boundaries and project layers
  + Definition of user stories
* **First system check** 
  + The basis for project structure, configuration, and design
  + Adapt project definition to changes after management review
  + Setup project as well as configuration (maven, including rest of tech stack)
  + Creating a final database model
  + Creating interfaces and DTOs
* **IR-1: ca. 25% of user stories implemented 18.05.2020**
  + Finish implementing Sprint 1
  + Start Sprint 2 implementation
  + Testing and verification of prototype
* **MR-2: ca. 50% of user stories implemented 25.05.2020**
  + Creating a test plan
  + Implementation of version 1 completed
  + Testing and verification of prototype
  + Development of user Documentation
  + Finish implementing Sprint 2
* **IR-2: ca. 80% of user stories implemented 15.06.2020**
  + Finish Sprint 3 implementation
  + Implementation of version 2 completed
  + Testing and verification of prototype
  + Development of user Documentation
  + Start Sprint 4 implementation
* **100% MR-3 of user stories implemented 22.06.2020**
  + Finish Sprint 4 implementation
  + Implementation of version 3 completed
  + Testing and verification of key components
  + Development of user Documentation
  + System and acceptance tests as well as test reports
  + Development completed, product delivery

## Roles

See Project name & Development team in the [Project Proposal](https://reset.inso.tuwien.ac.at/repo/2020ss-SEPM-group/ss20_sepm_qse_07/-/wikis/Project-Proposal).

**Horizontal Responsibility**

**Team Coordinators:** Sofiia Zholubak, Moritz Leander Großfurtner

Responsible for project management, keeping the project management artifacts up to date.

* Organizing & Planning
  + Project Plan, WBS (see Rough Plan)
  + Risk assessment, Iceberglist
* Controlling & Tracking
  + Timetable, Organisational Tickets of the Tracker
  + Reviews regarding the Status of the Project
* Managing the Workload and distribution
* Primary Contact Point for Client
* Organizing internal and external meetings
  + group meetings (Jour-Fixe), meetings with the Tutor, IR’s, MR’s
  + see Information System

**Technical Architects:** Ana Kubat, Andreas Krystallidis

Technical Architects organize the Infrastructure of the Project. That includes for example: Folder structure and Dependencies to Libraries.

The horizontal Responsibilities are:

* Project Object Model (Maven pom.xml)
  + Dependency Management, primarily Versions
  + Plugins
  + Usage of external Maven repositories
* Knowledge on all of the Technologies used for development.
* Define Coding Guidelines (Checkstyle).
* Define Guidelines for Source-Code documentation
* Handle the Enhancement-Tickets in the Tracker.
* Designing the architecture of the program and its components.
  + UML Component-diagram
  + Deployment Diagram

**Documentation Managers:** Omar Cehajic, Ana Kubat

Documentation manager is responsible for:

* Availability of the documentation, e.g. via Mercurial and Maven site.
* Integration of Java API documentation with the Maven Javadoc plugin
  + Every Java-Package has to be documented. A package-info.java file must be created for each Java package.
* Ensure that all classes, variables and methods are documented in English
* Creation of documentation guidelines (format and formatting guidelines, specification of code conventions, creation of templates, ...)
* Verification of compliance with documentation guidelines
* Checking the completeness of documents
* Organization and archiving of documents in the SCM

**Test Managers:** Andreas Krystallidis, Manuel Hofmann

Responsibilities of Test Manager:

* Test infrastructure
* Test libraries, test data (second DB)
  + Test Suites: Integration with Spring Framework to manage test data
  + Ensure that test code is separated from production code
* Creation of the test plan
  + Test procedure (code framework), planning of test runs
  + Effects in the event of errors, disclosure Behavior in the event of errors (exception handling)
  + Check compliance with test guidelines
* Monitoring of integration and system tests (strong cooperation with the technical architect)
* Regular review of all unit tests

**Build/Release Managers:** Manuel Hofmann, Omar Cehajic

Responsibilities of Build/Release Manager:

* Development environment infrastructure
  + Scripts to initialize, test or distribute (deploy)
  + Tagging in Source Code Management for every milestone
  + Build, deploy targets from Maven - project can be checked out and compiled at any time
* Expert knowledge of the build and SCM tools (Maven & Mercurial)

**UI Designer:** Moritz Leander Großfurtner, Sofiia Zholubak

Responsibilities of UI Designer:

● GUI designing

● Determining page order and page layout (page/element structure)

● Agree upon used frameworks/technologies (with Technical Architect)

● Provide design guideline and basic templates

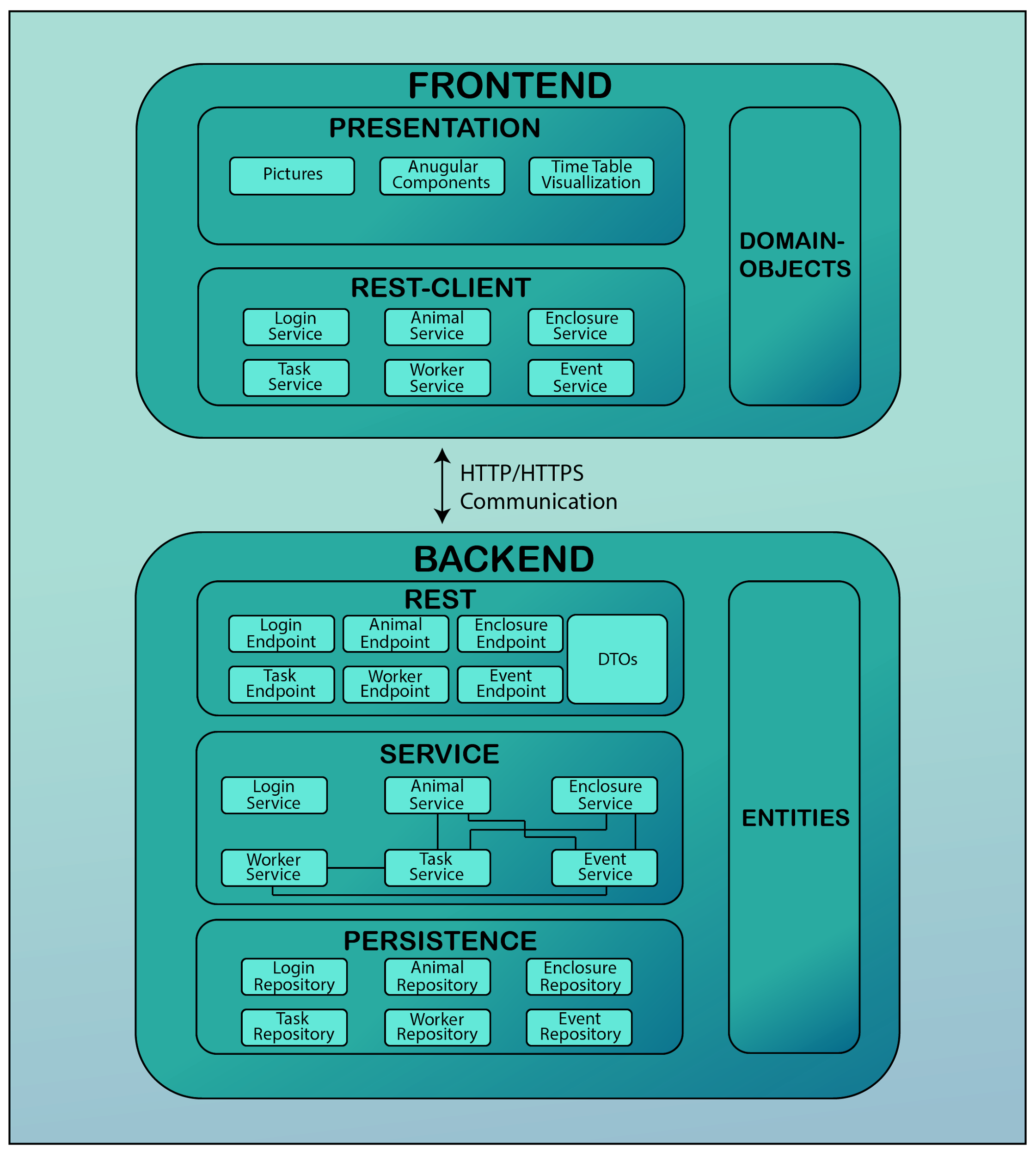
● Control and feedback on design and usability

## **Project boundaries**

The project’s boundaries will stay as described above and won’t be extended within the

scope of this course. As this project is not intended for money management, than to make zoo work structure easier, in scope of this project operations such as buying tickets won't be handled. Furthermore collecting statistical data, zoo ratings and giving feedback will also be excluded.

## **Layer Diagram**



## **Delivery Components**

**Software Components**

* The software will be delivered to the customer as an executable application that is available as a repository at: <https://reset.inso.tuwien.ac.at/repo/2020ss-SEPM-group/ss20_sepm_qse_07>
* After completion of the project, the full range of functions is available as defined in the functional and non-functional requirements described.
* Pre-Configured H2 database setup will also be delivered and it will be documented in code.

**Documentation Components**

* Project order
* Project proposal
* User guide
* REST Api documentation
* Technical documentation
* List of user-stories
* Test plan, functional test cases, test documentation
* Project management documentation

**Not Provided**

* Web-Page Design Mockups
* UML-Diagramms (except Layer Diagram and Domain Model provided in Project Order and Project Proposal)

## **Nonfunctional Requirements**

* Availability - The application can be used by a worker, an administrator, and it must be possible to serve clients. Availability should be provided at all times.
* Recovery - The data must be completely restorable in the event of a system failure (data is saved in the database).
* Usability - The primary Task is to give the workers an overview about the tasks that need to be done. This should be visualized on a single page. In addition to a clearly structured graphical user interface, a uniform structure is to be created, which runs through the entire application.
* Maintainability - The application should be implemented so that later maintenance is as easy as possible. This includes meaningful comments on the program code and a clean programming style.
* System compliance - The interface must at least be tested on Windows 10 and Windows 7 and be sufficiently usable to use all features without error.
* Browser compliance - The user interface must at least be tested on the most frequently used browsers at their latest versions (Firefox 75.0, Google Chrome Version 81, Microsoft Edge Version 44, Safari 13.1) and be sufficiently usable to use the features without errors.
* Scalability - Fundamental scalability makes sense, it should always be possible to add new tasks/workers.
* Data security - The data may only be viewed and changed by the operators. Passwords won’t be stored in plain text. All database operations will be designed in prepared statement form (database immune to sql-injections).

## **Risk Assessment**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Nr.** | **Type** | **Priority** | **Probability of occurrence** | **Following risks** | **Assignee** | **Description** | **Countermeasure** |
| **1** | always | high | medium | 2,3 | Sofiia | team member absence | select new role owner, balance workload between available team members |
| **2** | development | high | low |  | Ana, Andreas | project not in time | reduce feature set |
| **3** | management | high | low | 2 | Omar | absence of internal knowledge | good documentation coverage in code and git |
| **4** | planning, development | medium | medium | 2 | Manuel | design does not match the requirements | consult team, use of modular design concepts to reduce the risk and layered architecture |
| **5** | planning, development | medium | low | 2 | Moritz | team member low skills | reduce workload on lower skilled member and increase workload on higher skilled members |

## **Information system**

The information structure of the project will look the following:

* Weekly Meetings with the tutor (Jour Fixe)
* Regular group intern meetings using discord
* A total of 5 reviews (2 IRs, 3 MRs) on the 5 milestones
* Electronic communication synchronised using WhatsApp, Discord and Google Docs, unsynchronised using the Tracker
* Communication with the clients and tutor via mailing list

The provided infrastructure contains:

* Git Repository
* Gitlab
* Code template

## **Special Features**

1. Architecture

In this project we will use 5-layer architecture as the architectural style. Due to the distributed architecture, communication between client and browser takes place via the HTTP / HTTPS protocol. When it comes to frameworks, Spring and Angular will be used.

1. Task overview

From the employee aspect, it may be hard to keep on track with all upcoming obligations and events. To help everyday struggles of this kind, we will implement a task calendar[.](https://www.youtube.com/watch?v=dQw4w9WgXcQ) With this tool employees can see all finished and unfinished tasks

1. Work Balance

For the management of large companies such as zoos, it is essential to reflect a good working atmosphere that requires good working relations. In order not to overload any workers, we will create an automatic task assignment for workers. It consists of an algorithm that monitors the schedule of workers' obligations and assigns a new task to the worker who has been assigned the fewest tasks at that point. This way the duties will be distributed evenly to the workers.