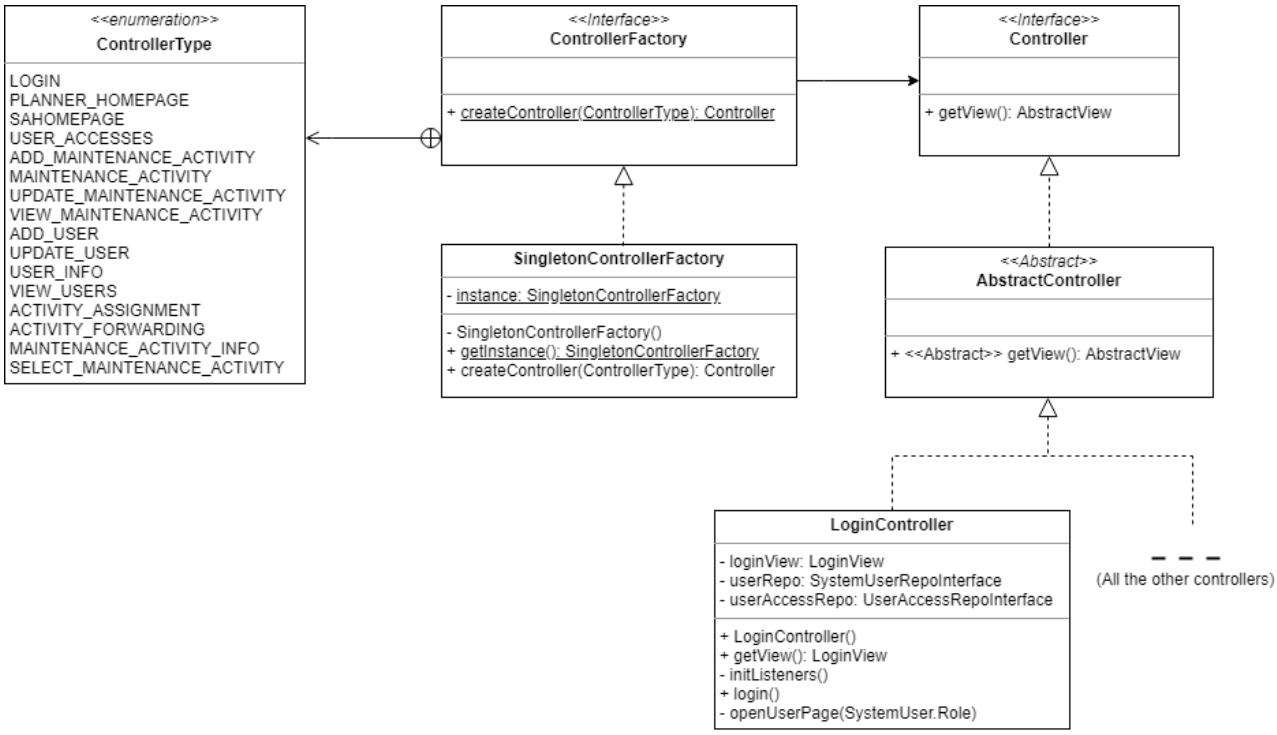


System Architecture

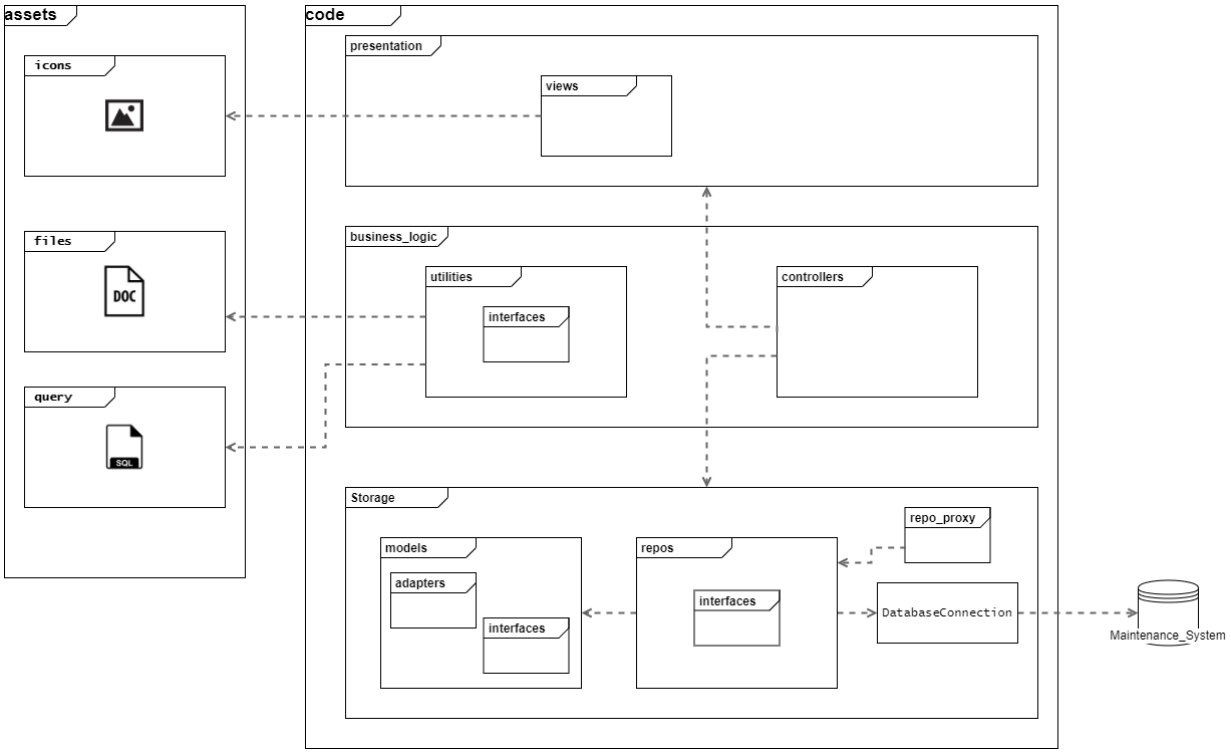
Problem Introduction

The proposed software is an application for the management of maintenance procedures in a industrial context. It is required the realization of a collection of user-friendly interfaces for the management of the employees and the assignment of the scheduled maintenance activities.
It is also requirement an infrastructure for the storage of informations.

Architecture Model

The architectural model of the application is based on a three-layer structure.

- **Presentation layer**: it manages the interaction with the user and the representation on screen of the informations.
- **Business logic layer**: it defines the business logic behind the interactions occurred between the presentation layer and the user.
- **Storage layer**: it defines the way data are collected and manipulated by the application layer.



MVC Structural Patter

The implementation of the Architecture has been performed through the realization of a Model-View-Controller based System in which the controller has a “mediator” role between the data models and the way they are represented on the views.

Each controller performs the transition to another “context” instantiating another controller, that builds a new view for the user. This passage is mediated by a centralized Controller factory that’s responsible for their construction.

Subsystems interaction

The architecture provides a further division by functionalities, in subsystems:

- Login;
- System Administrator Functionalities:
    - User Management.
    - Accesses Management.
- Planner Functionalities
    - Activities Management.
    - Task assignment functionalities.

Each functionality is distributed among the three layer and provides services to access and control in different ways different kind of data.

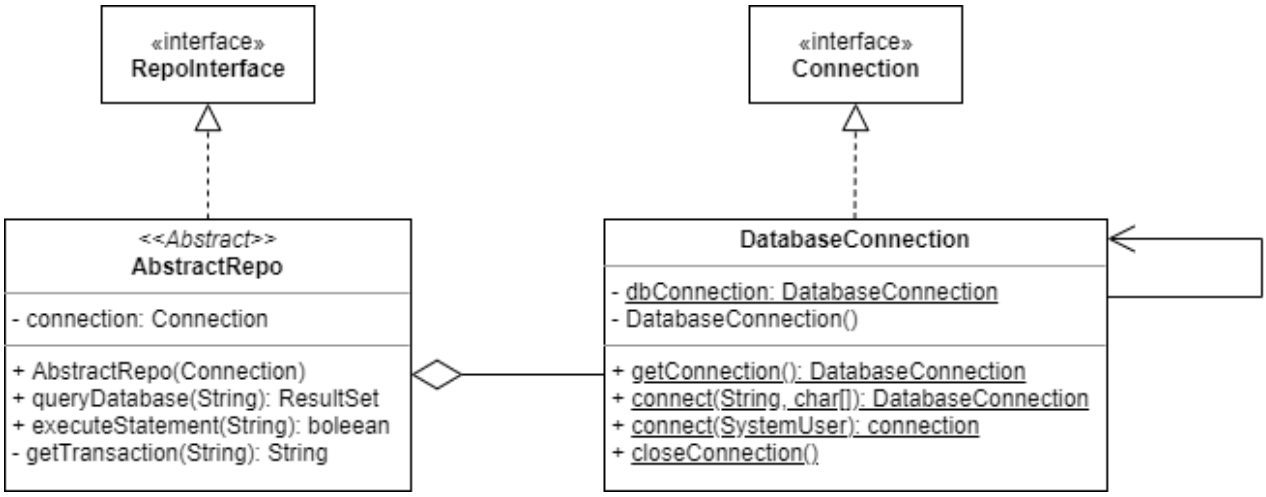
Each subsystem accedes services of another subsystem by calling a new controller and providing it, eventually, the date it needs to continue their elaboration.

Design patterns

In this section, it will be listed a collection of design pattern implemented in the system architecture.

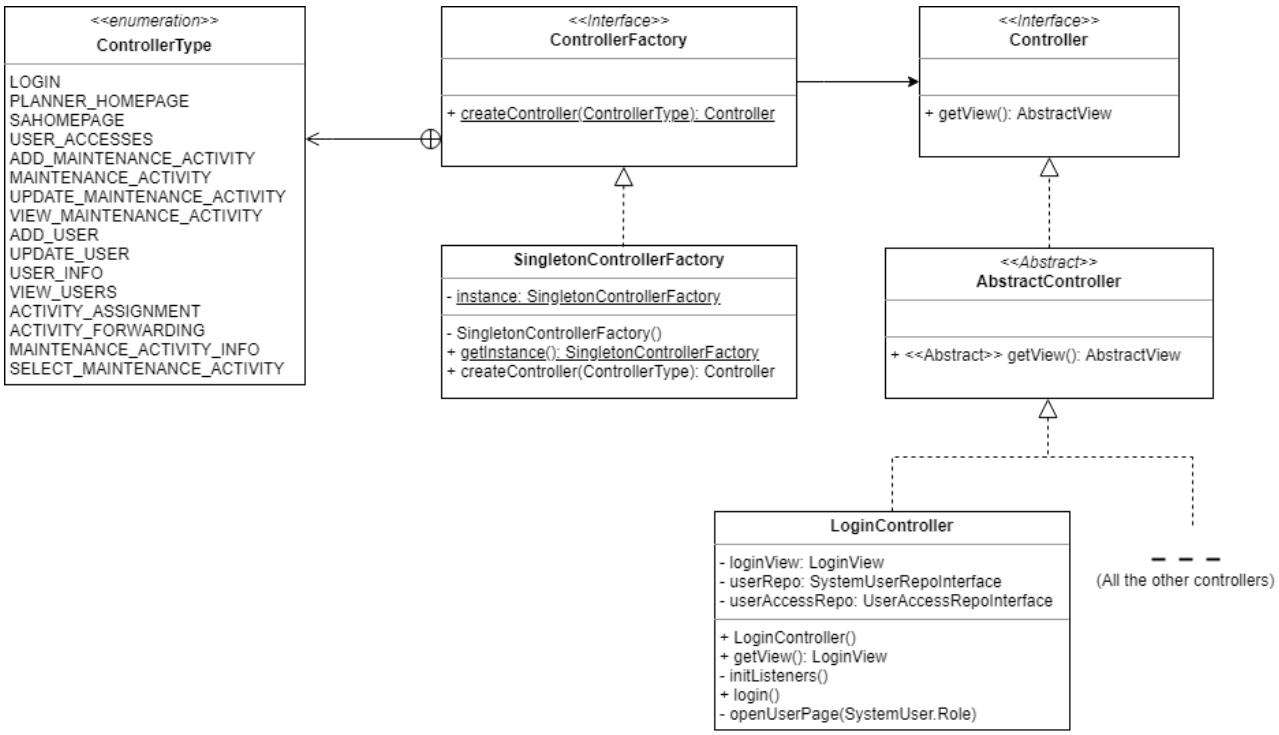
Singleton

- `SingletonControllerFactory`: is a centralized implementation of a `ControllerFactory` that generates instances of `Controller`.
- `DatabaseConnection`: is a centralized implementation of a `Connection` providing access to the database, to the storage level classes.



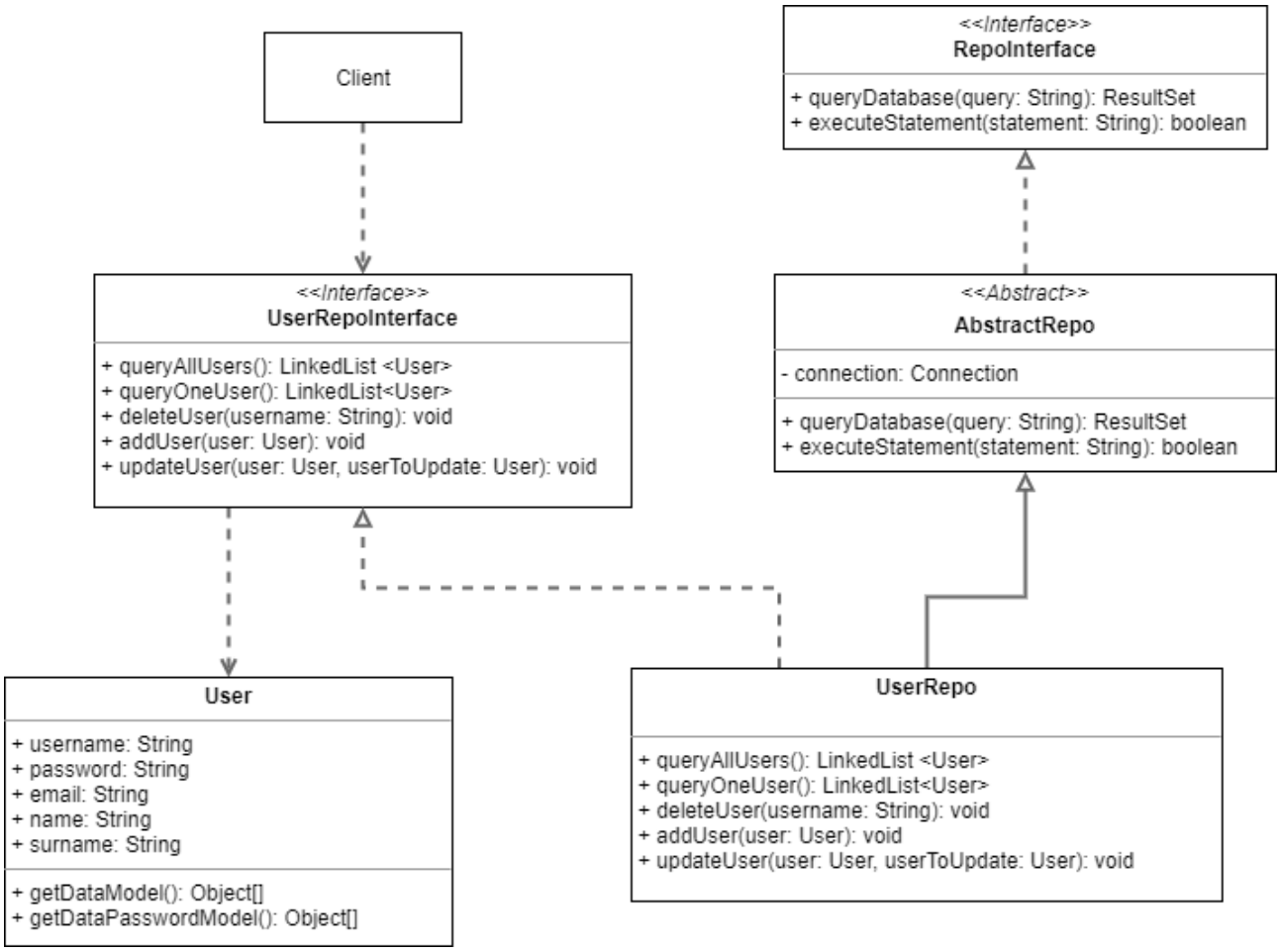
Factory

- `ControllerFactory:` provides a creational method to generate a generic `Controller` abstracting, when possible, from the implementation details.



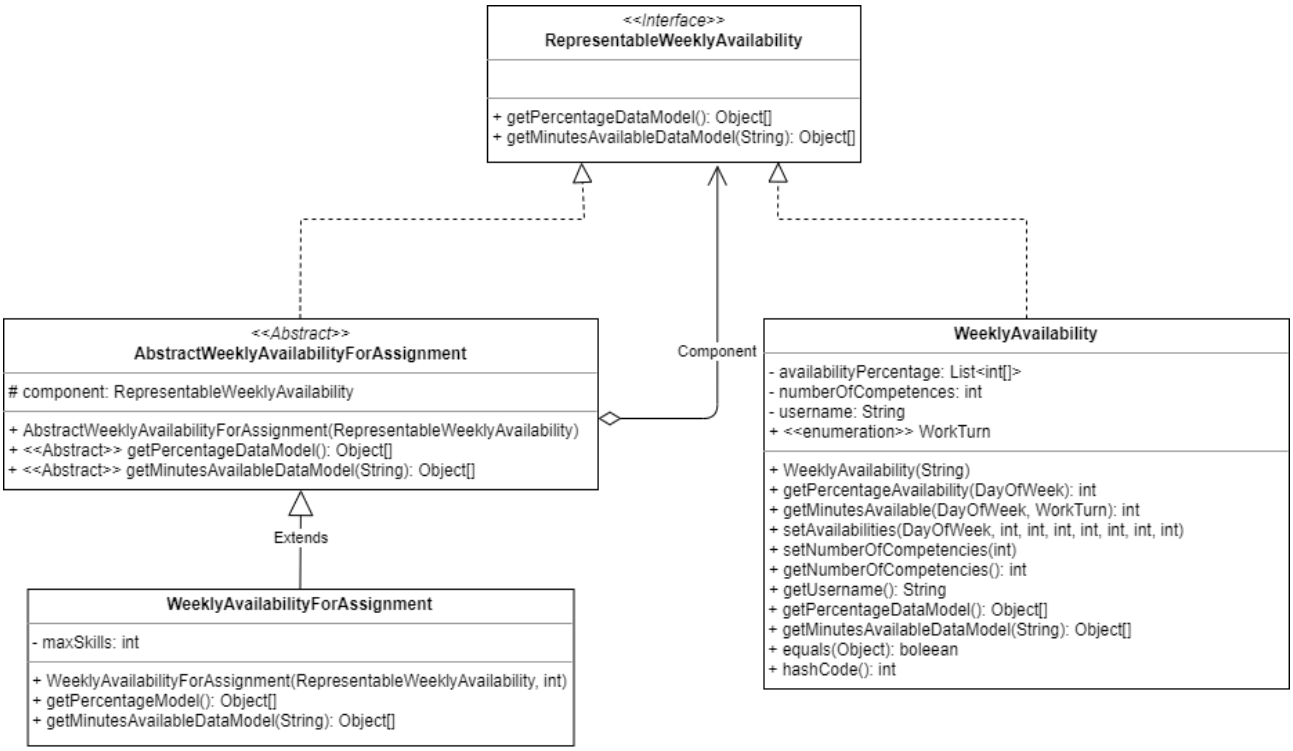
DAO

- `UserRepo`: provides all the Data Access methods required to manipulates and transfer the User informations between model and database *(There's a DAO for each model category)*.



Decorator

- `WeeklyAvailabilityForAssignment`: is a decorator that adds a post-processing of the `WeeklyAvailability` model's data, preparing it to be exposed on the `ActivityAssignmentView`.
- MaintenanceActivityForAssignment: is a decorator that adds a post-processing of the `MaintenanceActivity` model's data, preparing it to be exposed on the `ActivityAssignmentView`.



Proxy

- `UserProxyRepo:` is a Proxy implementation of a `UserRepo` that instantiates the target of the DAO requests only when it's really necessary, by instantiating it *(There's a Proxy for each DAO Repo class)*.