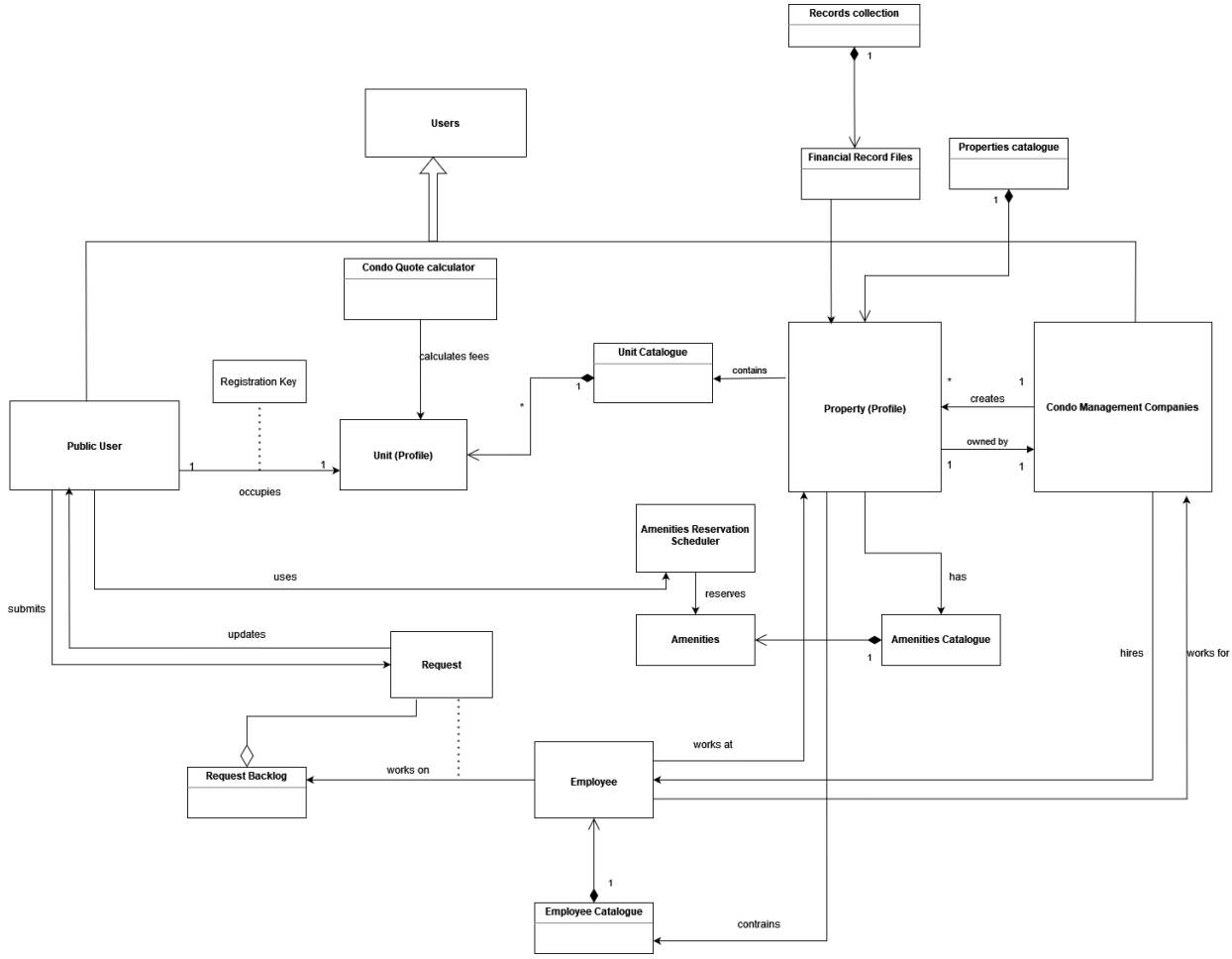
Software Architecture Document

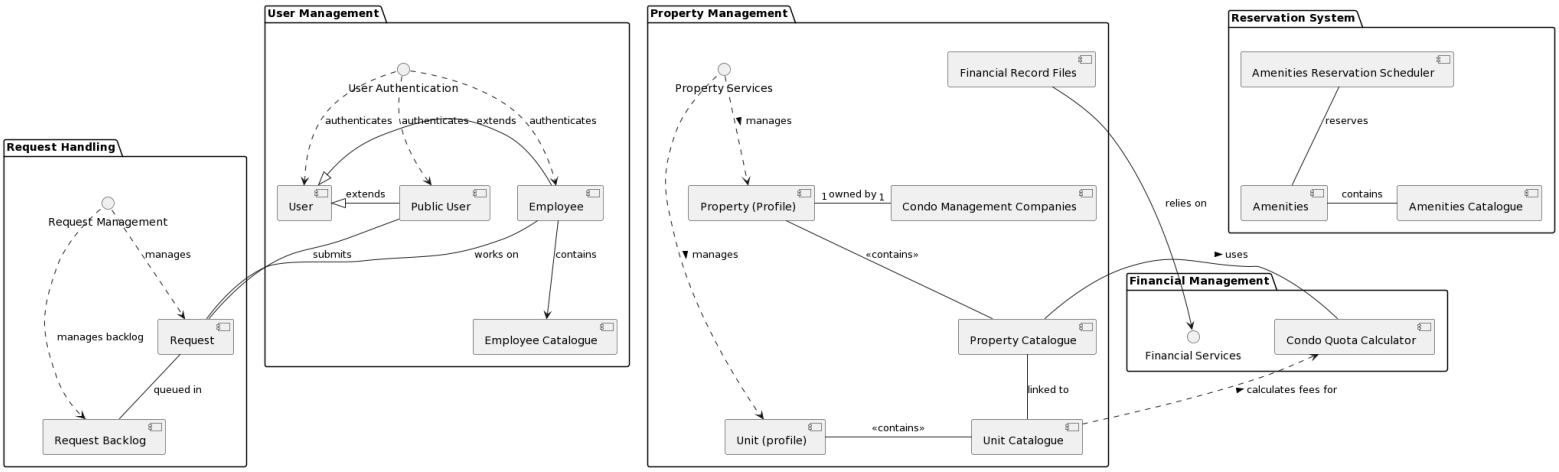
Domain Model



This domain model represents our current system and its entities, as well as the flow of data between said entities. The domain model helps to provide a strong foundation upon which the team can develop the architecture envisioned. It helps clear up ambiguities and keeps the entire team on the same page regarding the structure of our system.

Many of the requirements and/or features can be mapped in this model. For example, in the case of reserving a sauna room (reserving an amenity), a “Public User” will be interacting with the “Amenities Reservation Scheduler”, which in turn will reserve an “Amenity” that can be found in the “Amenity Catalogue”. All of our other user stories and required features can be mapped similarly on this diagram. In the case that a feature is not supported by this diagram, that will change in the near future during further sprints.

Component Diagram

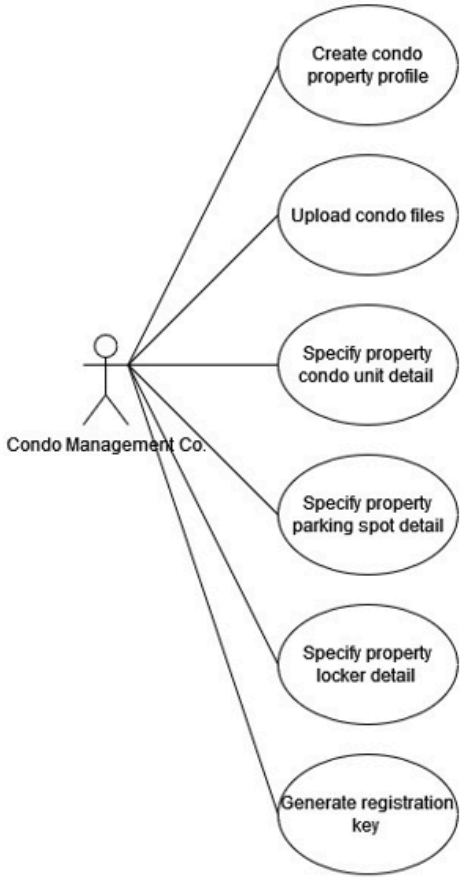


A component diagram represents a high level abstraction of the different components and working parts of our system. It is less specific in terms of components and how they are designed in comparison to the domain model, but it is far more interested in the interactions between components. Our system can be broken down into five major parts: “Request Handling”, “User Management”, “Property Management”, “Financial Management”, and “Reservation System”.

Use Case Diagrams

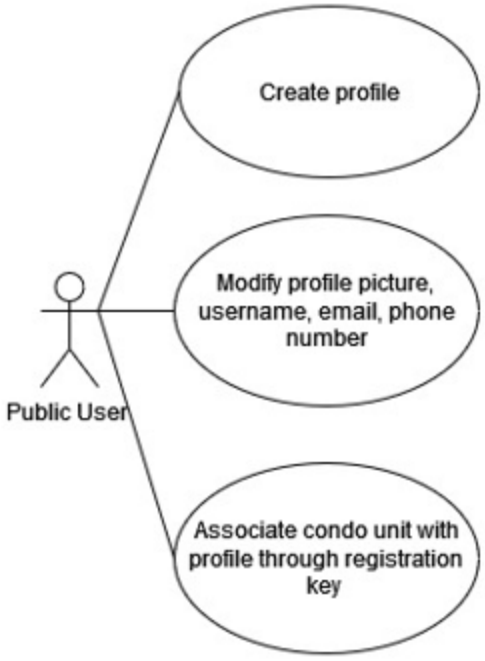
Use cases relevant to the current sprint's features were selected and made into diagrams to better illustrate them.

Use Case 1 : A condo management company can create and manage profiles for their properties

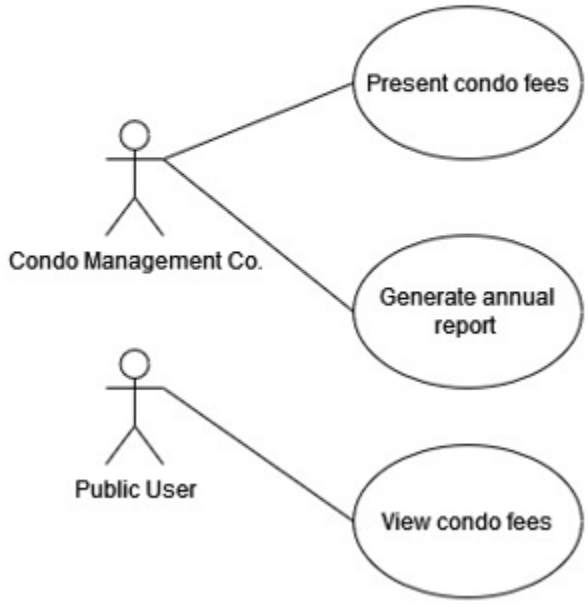


Co

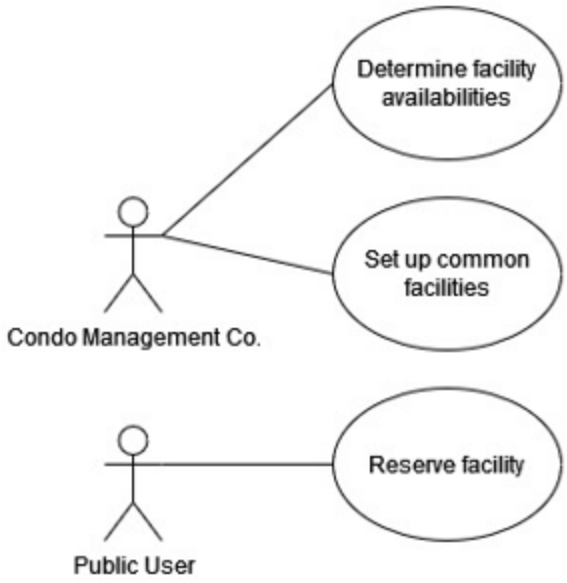
Use case 2: A public user can manage their condo profile



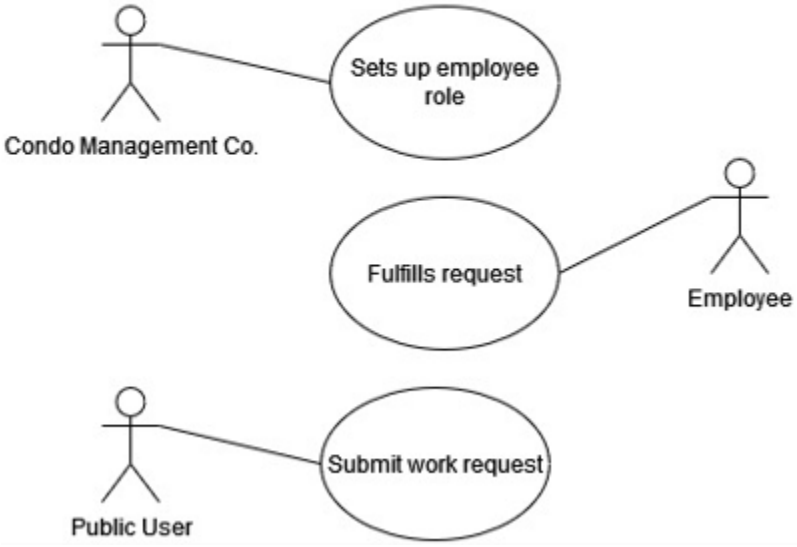
Use Case 3: Calculating condo unit financials



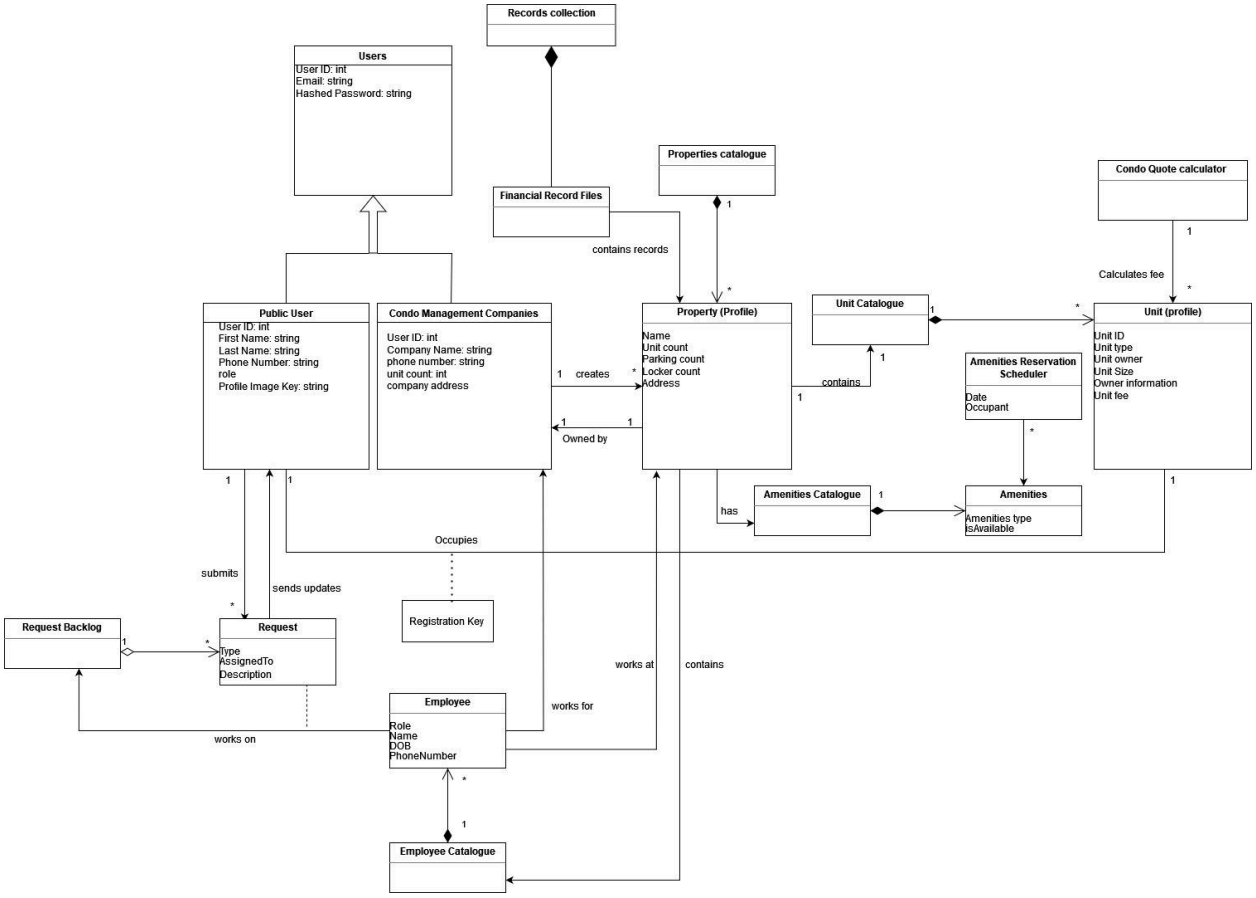
Use case 4: Common facilities can be reserved for usage



Use case 5: Work requests are treated according to type

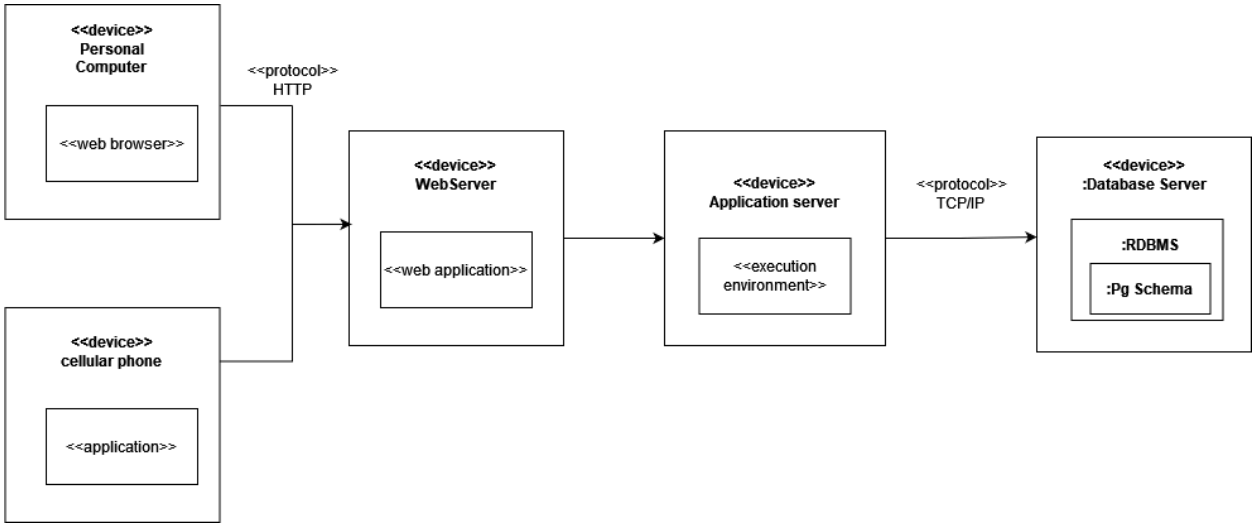


Class Diagram



The class diagram breaks down our important classes along with their attributes and methods. This is also helpful for identifying associations and dependencies in our system.

Deployment Diagram



This deployment diagram represents the deployment structure of our system consisting of a web, application, and database server. Client devices are personal computers accessing the system through our web app, and cellular phones through our android app. The physical client devices communicate with the web server through HTTP requests. The web server acts as an entry point for the client, as well as handling tasks such as routing, and serving frontend elements to devices. The application server serves mainly as a connection to the database by receiving HTTP requests from the web server, interpreting them, and performing operations on the database. It also serves to execute some application frameworks like Node.js. Finally, the database server hosts our database and all the data found within it. Our Relational Database Management System (RDBMS) runs PostgreSQL.