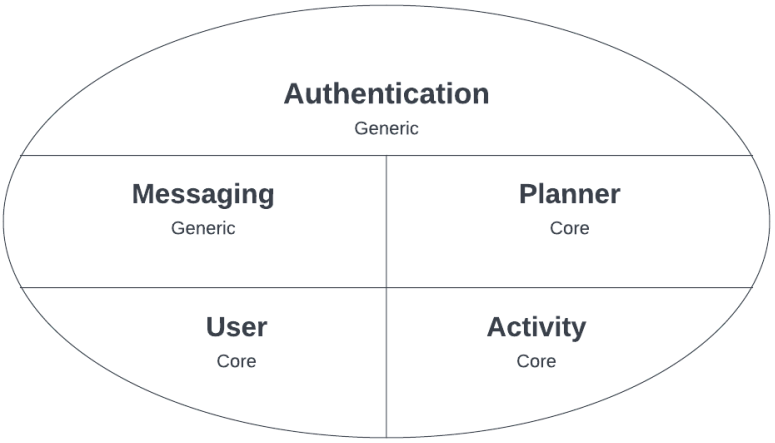
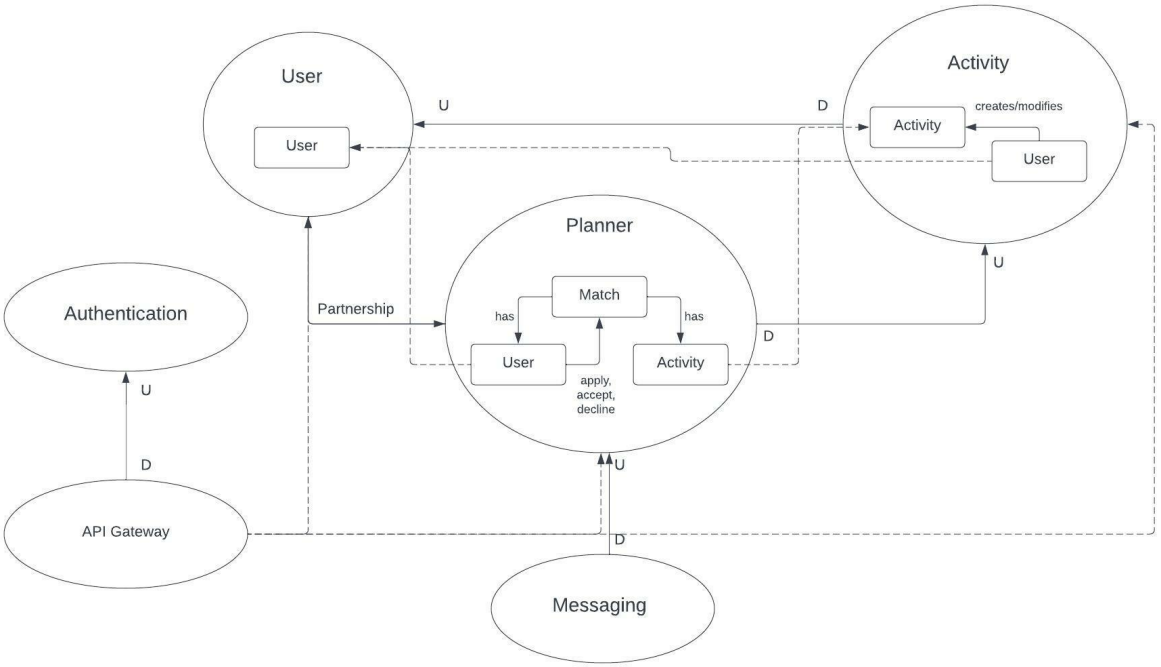
Bounding Context



- Authentication (Generic)
- Messaging (Generic)
- Users (Core)
    - username
    - first/last-name
    - password
    - availability
    - gender
    - certificate
    - organization
- Activities (Core)
    - time (start and end)
    - type - competition/training
    - certificate
    - optional filters (gender/professional level/organization)
- Appointment (Core)
    - User
    - Activity
    - Timeslot

Context Map

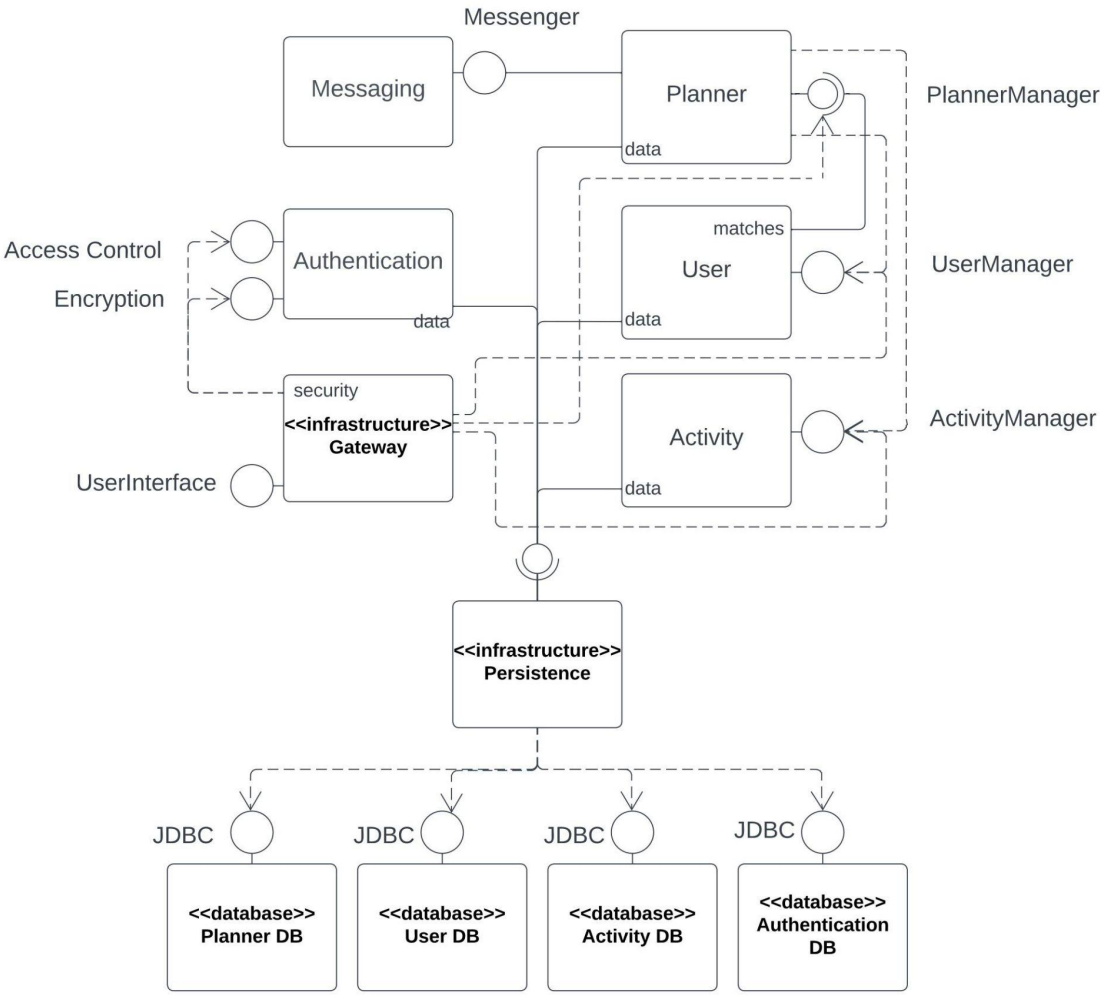


Microservices

- **User Microservice:**
    - It is a core component of the system because it provides all the functionality necessary to manage the user entities. The microservice exposes an interface which allows for the creation, removal or editing of an entity. It has an upstream relation with the Messaging microservice to allow for the retrieval of user specific information/data (e.g: a user's email address). It has a partnership relationship with the Planner microservice as it both supplies and demands information to it. Thus it can provide the Planner with the necessary user information to create all possible matches, but it also receives from the Planner all the possible activities that a user can enroll into. Inside of the User microservice there is no distinction between an activity owner and a simple activity participant; The most relevant information to this microservice is a user's availability and positions.
- **Activity Microservice:**
    - This microservice manages the Activity entities
    - It is a core component of the system because it provides essential functionality of the program, namely:
        - Users can create an Activity (Training or Competition)
        - Users who created an Activity become owners of such Activity, which entitles them to later edit or delete it
    - Activity Microservice is an upstream for the Planner as it supplies it with all available Activities
    - It also has downstream relationship with User Microservice, as it provides Activity Microservice with essential information to create an Activity
    - The planner would update an activity's info. When a user accepts a match, the request gets sent to the planner together with the user's id and the accepted activity's id. The Planner in turn, makes a call to the Activity microservice to update the activity's details accordingly: e.g adding the user in a specific role. The activity microservice handles the data internally; after it had received the request from the Planner it can update an activity's information accordingly, or even mark that activity as "fully booked", if that was the last position to be filled
- **Planner Microservice**
    - It is a core component of the system because it provides the functionality which is the purpose of the program - matching eligible rowers to training and competitions
    - Once the user requests for the eligible activities, it initially shows all available activities. Then, only when the user applies for it, it saves the match containing the following:
        - user id: reference to one of the users in the User microservice
        - match id : reference to one of the activities in the Activity microservice
        - status: one of revoked, pending, and accepted in the applicant user's perspective
    - When the user - particularly the activity owner - accepts or declines the applicant, the status of the match is updated correspondingly

    - It has a partnership with the User microservice. It provides matches based on the user's conditions. At the same time, it provides to the user about the matches available for the user to select. (Question regarding this relationship: Would User being the upstream content and Planner being the downstream content be more suitable?)
    - Activity is the downstream component for Planner since Planner only receives the list of applicable activities
    - Planner is the downstream component for Message since it provides information on the matches to send notification for and does not receive any data from it
- **Authentication Microservice:**
    - Authentication is a generic domain as it is not an essential part of the business logic (creating/joining activities), rather it facilitates a secure use of the core components of the system by authenticating users.
    - Authentication is provided by the Authentication Microservice which exposes an interface providing login, register and authentication functionality.
    - The authentication microservice will have an upstream relationship with the API Gateway, providing it with the JWT whenever a login operation is performed.
    - Whenever a request is made requiring authentication, the API gateway will first authenticate the JWT via the authentication microservice before routing the request to other microservices.
- **Messaging Microservice:**
    - Messaging is a generic domain as it is not an essential part of the business logic (creating/joining activities), rather it allows for sending messages to inform users whenever actions are performed by the core part of the system.
    - Messaging is provided by the Messaging Microservice which exposes an interface to send messages.
    - The messages will be push-based and implemented using the strategy design pattern, allowing for multiple strategies: in-app, e-mail etc.
    - The messaging microservice will have a downstream relationship with the planner, receiving messages that need to be sent.

UML Diagram


Messenger
Messaging
Planner
data
PlannerManager
Access Control
Encryption
Authentication
data
matches
User
data
UserManager
security
<<infrastructure>>
Gateway
UserInterface
Activity
data
ActivityManager
<<infrastructure>>
Persistence
JDBC
JDBC
JDBC
JDBC
<<database>>
Planner DB
<<database>>
User DB
<<database>>
Activity DB
<<database>>
Authentication
DB