instituto superior de engenharia de lisboa

Department of Electronical Engineering, Telecommunications and Computers

Database Documentation

50565: Ângelo Filipe Maia Azevedo (a50565@alunos.isel.pt)
50539: António Miguel Alves (a50539@alunos.isel.pt)

Database Documentation for the Remote Lab Project

Advisor: Prof. Pedro Miguens Matutino

June 2025

Contents

List of Figures

0.1 Introduction

This document provides a comprehensive overview of the database structure, including its entities, attributes, and relationships. It also details key implementation decisions.

0.2 Database overview

The database has been designed using an Entity-Relationship (ER) approach. This methodology enables a clear understanding of how different entities interact within the system. The following section presents the ER model diagram.

The database is implemented using PostgreSQL and has been tested with sample data in a Docker container.

The next section will present the ER model in detail.

0.3 Entity-Relationship Model

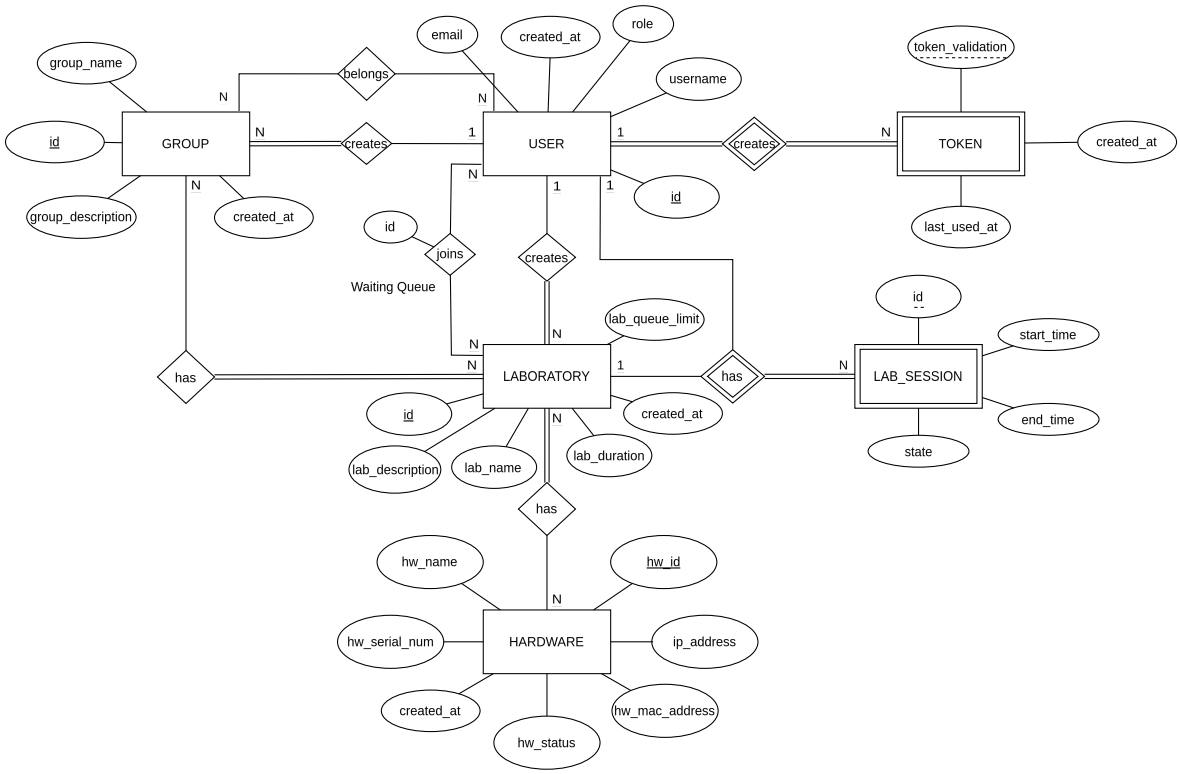

Figure 1: ER Model

0.4 Entities and Attributes

This section provides a comprehensive description of each entity and their attributes.

0.4.1 User

The **User** is a fundamental entity in the database that represents system users.

It has an **user_id** as its primary key, implemented as an identity column. An identity column automatically generates values from an implicit sequence. Therefore, whenever a new user is created, a unique id is automatically generated. The user_id is an integer data type, chosen to simplify database queries by using a simple numeric identifier instead of email or username.

The entity includes character sequence attributes: **username** and **email**. All these fields are required (not null), and the email must be unique. Initially, the username is automatically set to match the user's username returned by OAuth login. While usernames are used for display purposes and may not be unique, emails serve as unique identifiers for user searches. The specific constraints for these attributes are defined in the application domain.

Additionally, it includes a **student_nr** (student number) as a unique integer field and **created_at** as a required timestamp. The student number is optional, allowing the system to accommodate non-student users such as professors. Notably, there is no discriminator attribute as user type distinctions are handled through the Role-Based Access Control (RBAC) system.

0.4.2 Token

Token is implemented as a weak entity since it cannot exist independently and requires a user_id for identification. Each token is associated with a user and is essential for authentication purposes. Users must have a valid token when interacting with the system.

As a weak entity, it requires a partial key, which is the **token_validation** field. This attribute stores a randomly generated hash created by the system and is used to verify token validity. For instance, when a user's token is included in an authorization header, the system compares it with this stored value during each interaction. It is implemented as a required character sequence.

The entity also includes **created_at** and **last_used_at** timestamps, both required fields. These attributes enable token validity checking based on timing rules defined by the application domain.

0.4.3 Group

The **Group** entity represents various types of user groupings, such as student classes, work groups, or professor groups.

Each group has a **group_id** as its primary key, implemented as an auto-generated integer identity column. This unique identifier facilitates efficient querying and group identification.

The **group_name** is a required character sequence that stores the user-defined name for

the group. Group names can be modified after creation, with specific length and format restrictions defined by the application domain.

Groups can include an optional **group_description** stored as a text type. This field allows users to provide detailed information about the group's purpose or characteristics. While technically unlimited in length, practical limitations may be imposed by the application domain.

The entity also includes a **created_at** timestamp, which can be used for analytical and administrative purposes.

0.4.4 Laboratory

The **Laboratory** entity represents physical or virtual laboratory spaces within the system.

Each laboratory has a **lab_id** as its primary key, implemented as an auto-generated integer identity column. This unique identifier simplifies laboratory identification and database queries.

The **lab_name** attribute is a required character sequence that stores the laboratory's designation. Specific naming conventions and restrictions are defined by the application domain.

The entity includes a **lab_duration** attribute, which specifies the standard duration of laboratory sessions in minutes. This integer field is required and helps manage session scheduling.

Additionally, it includes a **created_at** timestamp for tracking laboratory creation and administrative purposes.

0.4.5 Lab Session

Lab Session is implemented as a weak entity that depends on both User and Laboratory entities. A session represents a specific time period during which a user can access and operate laboratory equipment.

The entity uses **session_id** as a partial key, which, combined with user_id and lab_id, uniquely identifies each session. The session_id is an auto-generated integer identity column that facilitates session tracking and queries.

Each session includes required **start_time** and **end_time** timestamps that define the session's temporal boundaries.

The **state** attribute is a required character sequence that indicates the session's current status (e.g., active, inactive, or scheduled). The system can be extended to support additional states as needed, with any such changes to be documented in future updates.

0.4.6 Hardware

The **Hardware** entity represents physical equipment within the laboratory system, such as computers or FPGAs.

Each piece of hardware is identified by a **hw_id** primary key, implemented as an auto-generated integer identity column to facilitate equipment tracking and queries.

The entity includes a required **hw_name** attribute as a character sequence for equipment identification.

A required **hw_serial_num** character sequence stores the hardware's serial number for inventory management.

The **ip_address** and **hw_mac_address** are character sequence fields that may be null depending on the hardware type, as not all equipment requires network connectivity.

A required **hw_status** character sequence tracks the current state of the hardware (e.g., available, occupied).

Like other entities, it includes a **created_at** timestamp for inventory tracking and administrative purposes.

0.5 Associations

This section details the relationships between entities, explaining how they interact within the system and what restrictions they have.

User associations include:

- **Token** - A weak association where each user may have multiple tokens (N:1). Tokens are automatically generated by the system as needed, with the user's primary key serving as part of the token's composite key.

- **App Invite** - A weak association where users can create multiple app invites (N:1). The user's primary key is incorporated into the app invite's key structure. Only a user with Professor's role can create app invites. It is worth noting that an Administrator can also hold the position of Professor due to hierarchical considerations.

- **Group** - Users have two distinct relationships with groups:

  - **creates** - Users can create multiple groups, but each group has exactly one creator (N:1). Only Professor can create groups.

  - **belongs** - Users can belong to multiple groups, and groups can have multiple members (N:N)

- **Laboratory** - Users have two types of laboratory associations:

  - **creates** - Users can create multiple laboratories, but each laboratory has one creator (N:1). Only Professor can create laboratories.
  - **joins** - Users can join multiple laboratories over time, and laboratories can be joined by multiple users (N:N). While the application domain may impose restrictions (e.g., limiting concurrent laboratory access), the database structure maintains this flexibility to support historical tracking.

- **Lab Session** - A weak association where users can have multiple sessions, but each session belongs to exactly one user (N:1)

App Invite maintains an additional association with the Group entity, linking invites to specific groups. Each invite corresponds to one group, while groups can have multiple associated invites (N:1). This enables automatic group assignment upon registration.

Group has an additional association with Laboratory, controlling laboratory access permissions. Groups can be associated with multiple laboratories, and laboratories can be accessible to multiple groups (N:N).

Laboratory maintains these additional associations:

- **Lab Session** - Laboratories can have multiple sessions, but each session is associated with exactly one laboratory (N:1)

- **Hardware** - Laboratories can be assigned multiple pieces of hardware, and hardware can be assigned to multiple laboratories over time (N:N)

Note that additional constraints and business rules are implemented at the web API level. Please refer to the web API documentation for detailed information about these restrictions.

Bibliography

[1] PostgreSQL Global Development Group. *PostgreSQL Documentation*. [Online]. Available: `https://www.postgresql.org/docs/`