Game Session and Stat Tracker

An app to keep track of scores and other stats during a board game to be seen on your profile for the future.

Feature 1:

Live Room Points and stats tracking

Feature 2:

Tracking for all stats of games you have played and tracked with the Live Room Tracker

The Librarian: Python API

This service acts as the foundational layer for all historical and user-related data. It will be a standard API built with Python, using a lightweight web framework.

- Core Responsibilities:
  - User Management: Handles user registration, login, and authentication. It will issue tokens (like JWT) that the Android app will use to authorize its requests.
  - Data Management: Provides standard CRUD (Create, Read, Update, Delete) endpoints for managing your library of games and players.
  - History & Statistics: Exposes endpoints for the app to retrieve past game session results and calculated statistics (e.g., a player's win/loss record).
- Example API Endpoints:
  - POST /auth/login
  - GET /users/{user_id}/stats
  - GET /sessions/history
  - POST /games

The Scorekeeper: Stateless Elixir Service

This service's sole purpose is to manage real-time communication for live game sessions. It will be built with Elixir and the Phoenix Framework, using Phoenix Channels for WebSocket communication.

- ## Stateless Architecture:
    - The key design principle here is that the Elixir service does not hold the score state in its memory. When a score update is received from a user:
    - The service receives the update via its WebSocket connection.
    - It immediately writes the new score to the live_scores table in the MySQL database.
    - It then broadcasts the new, complete scoreboard state (read from the database) to all other users connected to that specific game's channel.

This ensures that if the service restarts, no live game data is lost, as the database always holds the most current state.

- ## Core Responsibilities:
    - Manage WebSocket connections for active game channels.
    - Receive score update events from the "scorekeeper" player.
    - Persist live score changes to the database immediately.
    - Broadcast the updated state to all connected clients in that channel.

The Database: MySQL Schema

The MySQL database is the central source of truth for the entire application, serving both the Python and Elixir services. The schema will be organized to allow for storage minimization and query reduction. It is also set up to allow users to track any set of things they'd like during a game.

Assume fields are not nullable unless otherwise stated

- -- Schema BoardGameTracker
    - CREATE SCHEMA IF NOT EXISTS `BoardGameTracker` DEFAULT CHARACTER SET utf8 ;
    - USE `BoardGameTracker` ;
- -- Table `BoardGameTracker`.`user`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`user` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `username` VARCHAR(45) NOT NULL,
        - `email` VARCHAR(45) NOT NULL,
        - `password_hash` VARCHAR(45) NOT NULL,
        - `created_at` DATETIME NOT NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `user_id_UNIQUE` (`id` ASC) VISIBLE,
        - UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE)
    - ENGINE = InnoDB;
- -- Table `BoardGameTracker`.`game`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`game` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `game_name` VARCHAR(45) NOT NULL,
        - `max_player_count` INT NULL,
        - `min_player_count` INT NULL,
        - `description` TEXT NOT NULL,
        - `can_win` TINYINT NOT NULL DEFAULT 1,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `game_id_UNIQUE` (`id` ASC) VISIBLE,
        - UNIQUE INDEX `game_name_UNIQUE` (`game_name` ASC) VISIBLE)
    - ENGINE = InnoDB;
- -- Table `BoardGameTracker`.`user_game_stats_cache`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`user_game_stats_cache` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `game_id` INT NOT NULL,
        - `user_id` INT NOT NULL,

        - `total_times_played` INT NOT NULL,
        - `wins` INT NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `user_game_id_UNIQUE` (`id` ASC) VISIBLE,
        - INDEX `fk_user_game_stats_game1_idx` (`game_id` ASC) VISIBLE,
        - INDEX `fk_user_game_stats_user1_idx` (`user_id` ASC) VISIBLE,
        - CONSTRAINT `fk_user_game_stats_game1`
            - FOREIGN KEY (`game_id`)
            - REFERENCES `BoardGameTracker`.`game` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION,
        - CONSTRAINT `fk_user_game_stats_user1`
            - FOREIGN KEY (`user_id`)
            - REFERENCES `BoardGameTracker`.`user` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION)
    - ENGINE = InnoDB;
- -- Table `BoardGameTracker`.`session`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`session` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `game_id` INT NOT NULL,
        - `session_key` VARCHAR(6) NOT NULL,
        - `time_started` DATETIME NOT NULL,
        - `time_ended` DATETIME NULL,
        - `current_round` INT NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `live_session_id_UNIQUE` (`id` ASC) VISIBLE,
        - INDEX `fk_live_session_game1_idx` (`game_id` ASC) VISIBLE,
        - UNIQUE INDEX `session_key_UNIQUE` (`session_key` ASC) VISIBLE,
        - CONSTRAINT `fk_live_session_game1`
            - FOREIGN KEY (`game_id`)
            - REFERENCES `BoardGameTracker`.`game` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION)
    - ENGINE = InnoDB;
- -- Table `BoardGameTracker`.`session_player`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`session_player` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `session_id` INT NOT NULL,
        - `user_id` INT NULL,
        - `player_name` VARCHAR(45) NOT NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `live_session_user_id_UNIQUE` (`id` ASC) VISIBLE,

        - INDEX `fk_live_session_player_user1_idx` (`user_id` ASC) VISIBLE,
        - INDEX `fk_live_session_player_session1_idx` (`session_id` ASC) VISIBLE,
        - CONSTRAINT `fk_live_session_player_user1`
            - FOREIGN KEY (`user_id`)
            - REFERENCES `BoardGameTracker`.`user` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION,
        - CONSTRAINT `fk_live_session_player_session1`
            - FOREIGN KEY (`session_id`)
            - REFERENCES `BoardGameTracker`.`session` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION)
    - ENGINE = InnoDB;
- -- Table `BoardGameTracker`.`scope`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`scope` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `scope` VARCHAR(45) NOT NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
        - UNIQUE INDEX `scope_UNIQUE` (`scope` ASC) VISIBLE)
    - ENGINE = InnoDB;
- -- Table `BoardGameTracker`.`data_type`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`data_type` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `data_type` VARCHAR(45) NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE)
    - ENGINE = InnoDB;
- -- Table `BoardGameTracker`.`session_tracked_stat`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`session_tracked_stat` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `session_id` INT NOT NULL,
        - `stat_name` VARCHAR(45) NOT NULL,
        - `data_type_id` INT NOT NULL,
        - `scope_id` INT NOT NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
        - INDEX `fk_session_tracked_stat_scope1_idx` (`scope_id` ASC) VISIBLE,
        - INDEX `fk_live_session_tracked_stat_data_type1_idx` (`data_type_id` ASC) VISIBLE,

        - INDEX `fk_session_tracked_stat_session1_idx` (`session_id` ASC) VISIBLE,
        - CONSTRAINT `fk_session_tracked_stat_scope1`
            - FOREIGN KEY (`scope_id`)
            - REFERENCES `BoardGameTracker`.`scope` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION,
        - CONSTRAINT `fk_live_session_tracked_stat_data_type1`
            - FOREIGN KEY (`data_type_id`)
            - REFERENCES `BoardGameTracker`.`data_type` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION,
        - CONSTRAINT `fk_session_tracked_stat_session1`
            - FOREIGN KEY (`session_id`)
            - REFERENCES `BoardGameTracker`.`session` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION)
    - ENGINE = InnoDB;
- ## -- Table `BoardGameTracker`.`player_stat_value`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`player_stat_value` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `session_tracked_stat_id` INT NOT NULL,
        - `session_player_id` INT NOT NULL,
        - `stat_value` VARCHAR(45) NOT NULL,
        - `recorded_at` DATETIME NOT NULL,
        - `round_number` INT NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
        - INDEX `fk_player_stat_value_session_tracked_stat1_idx` (`session_tracked_stat_id` ASC) VISIBLE,
        - INDEX `fk_player_stat_value_session_player1_idx` (`session_player_id` ASC) VISIBLE,
            - CONSTRAINT `fk_player_stat_value_session_tracked_stat1`
            - FOREIGN KEY (`session_tracked_stat_id`)
            - REFERENCES `BoardGameTracker`.`session_tracked_stat` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION,
        - CONSTRAINT `fk_player_stat_value_session_player1`
            - FOREIGN KEY (`session_player_id`)
            - REFERENCES `BoardGameTracker`.`session_player` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION)
    - ENGINE = InnoDB;

- ## -- Table `BoardGameTracker`.`table_stat_value`
  - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`table_stat_value` (
    - `id` INT NOT NULL AUTO_INCREMENT,
    - `session_tracked_stat_id` INT NOT NULL,
    - `session_id` INT NOT NULL,
    - `stat_value` VARCHAR(45) NOT NULL,
    - `recorded_at` DATETIME NOT NULL,
    - `round_number` INT NULL,
    - PRIMARY KEY (`id`),
    - UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
    - INDEX `fk_table_stat_value_session_tracked_stat1_idx` (`session_tracked_stat_id` ASC) VISIBLE,
    - INDEX `fk_table_stat_value_session1_idx` (`session_id` ASC) VISIBLE,
    - CONSTRAINT `fk_table_stat_value_session_tracked_stat1`
      - FOREIGN KEY (`session_tracked_stat_id`)
      - REFERENCES `BoardGameTracker`.`session_tracked_stat` (`id`)
      - ON DELETE NO ACTION
      - ON UPDATE NO ACTION,
    - CONSTRAINT `fk_table_stat_value_session1`
      - FOREIGN KEY (`session_id`)
      - REFERENCES `BoardGameTracker`.`session` (`id`)
      - ON DELETE NO ACTION
      - ON UPDATE NO ACTION)
  - ENGINE = InnoDB;
- ## -- Table `BoardGameTracker`.`game_tracked_stat_set`
  - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`game_tracked_stat_set` (
    - `id` INT NOT NULL AUTO_INCREMENT,
    - `game_id` INT NOT NULL,
    - `user_id` INT NOT NULL,
    - `set_name` VARCHAR(45) NOT NULL,
    - PRIMARY KEY (`id`),
    - UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
    - INDEX `fk_game_tracked_stat_set_game1_idx` (`game_id` ASC) VISIBLE,
    - INDEX `fk_game_tracked_stat_set_user1_idx` (`user_id` ASC) VISIBLE,
    - CONSTRAINT `fk_game_tracked_stat_set_game1`
      - FOREIGN KEY (`game_id`)
      - REFERENCES `BoardGameTracker`.`game` (`id`)
      - ON DELETE NO ACTION
      - ON UPDATE NO ACTION,
    - CONSTRAINT `fk_game_tracked_stat_set_user1`

            - FOREIGN KEY (`user_id`)
            - REFERENCES `BoardGameTracker`.`user` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION)
    - ENGINE = InnoDB;
- ## -- Table `BoardGameTracker`.`game_tracked_stat`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`game_tracked_stat` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `game_tracked_stat_set_id` INT NOT NULL,
        - `stat_name` VARCHAR(45) NOT NULL,
        - `data_type_id` INT NOT NULL,
        - `scope_id` INT NOT NULL,
        - INDEX `fk_table2_game_tracked_stat_set1_idx` (`game_tracked_stat_set_id` ASC) VISIBLE,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
        - INDEX `fk_table2_scope1_idx` (`scope_id` ASC) VISIBLE,
        - INDEX `fk_table2_data_type1_idx` (`data_type_id` ASC) VISIBLE,
        - CONSTRAINT `fk_table2_game_tracked_stat_set1`
            - FOREIGN KEY (`game_tracked_stat_set_id`)
            - REFERENCES `BoardGameTracker`.`game_tracked_stat_set` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION,
        - CONSTRAINT `fk_table2_scope1`
            - FOREIGN KEY (`scope_id`)
            - REFERENCES `BoardGameTracker`.`scope` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION,
        - CONSTRAINT `fk_table2_data_type1`
            - FOREIGN KEY (`data_type_id`)
            - REFERENCES `BoardGameTracker`.`data_type` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION)
    - ENGINE = InnoDB;
- ## -- Table `BoardGameTracker`.`follower`
    - CREATE TABLE IF NOT EXISTS `BoardGameTracker`.`follower` (
        - `id` INT NOT NULL AUTO_INCREMENT,
        - `user_id` INT NOT NULL,
        - `following_user_id` INT NOT NULL,
        - `followed_at` DATETIME NOT NULL,
        - PRIMARY KEY (`id`),
        - UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,

        - INDEX `fk_follower_user1_idx` (`user_id` ASC) VISIBLE,
        - INDEX `fk_follower_user2_idx` (`following_user_id` ASC) VISIBLE,
        - CONSTRAINT `fk_follower_user1`
            - FOREIGN KEY (`user_id`)
            - REFERENCES `BoardGameTracker`.`user` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION,
        - CONSTRAINT `fk_follower_user2`
            - FOREIGN KEY (`following_user_id`)
            - REFERENCES `BoardGameTracker`.`user` (`id`)
            - ON DELETE NO ACTION
            - ON UPDATE NO ACTION)
    - ENGINE = InnoDB;

The Frontend: Kotlin Android App

The user-facing part of the project will be a native Android application built with Kotlin. It will act as the client for both of your backend services.

- Core Responsibilities:
    - Provide a user interface for logging in, managing games, and viewing player stats.
    - Allow a user to start a "live session" or join an existing one using a session key.
    - Display a real-time scoreboard that updates automatically without needing to be refreshed.
- Backend Communication:
    - HTTP Requests: The app will use a library like Retrofit to communicate with the Python "Librarian" REST API. This is for all non-real-time actions like logging in or viewing history.
    - WebSocket Connection: For live sessions, the app will use a library like OkHttp to open a persistent WebSocket connection to the Elixir "Scorekeeper" service. It will send score updates and listen for broadcasted changes through this connection.