Auction Game

Web-based multiplayer auction game built on Node.js and Angular.js.

Tech Stack

- Postgres for persistency
- Node.js + Express.js for HTTP server and API
- Socket.io for efficient client-server event-based communication
- Passport.js for JWT tokens
- Angular.js for MVVM
- Angular Material for Material design UI components
- Jade for templating
- Stylus for CSS
- Font Awesome and Flaticon icons
- Gulp for compiling UI
- Mocha.js and Sinon.js for testing
- Docker for running and deploying the application everywhere
- Patterns: Repository, MVVM, DI, Decorator

Data Model

Data model described as a set of tables in relational database:

- *players* - registered players and their coin balance
- *inventory* - list of items and their quantities assigned to player by id
- *auctions* - list of auctions, ordered by *created* timestamp with seller and winner player ids and winning bid.
- *player_sessions* - each logged in player has a unique key associated with its session, when player logs in, another session associated with the same player id (and name) automatically invalidated.

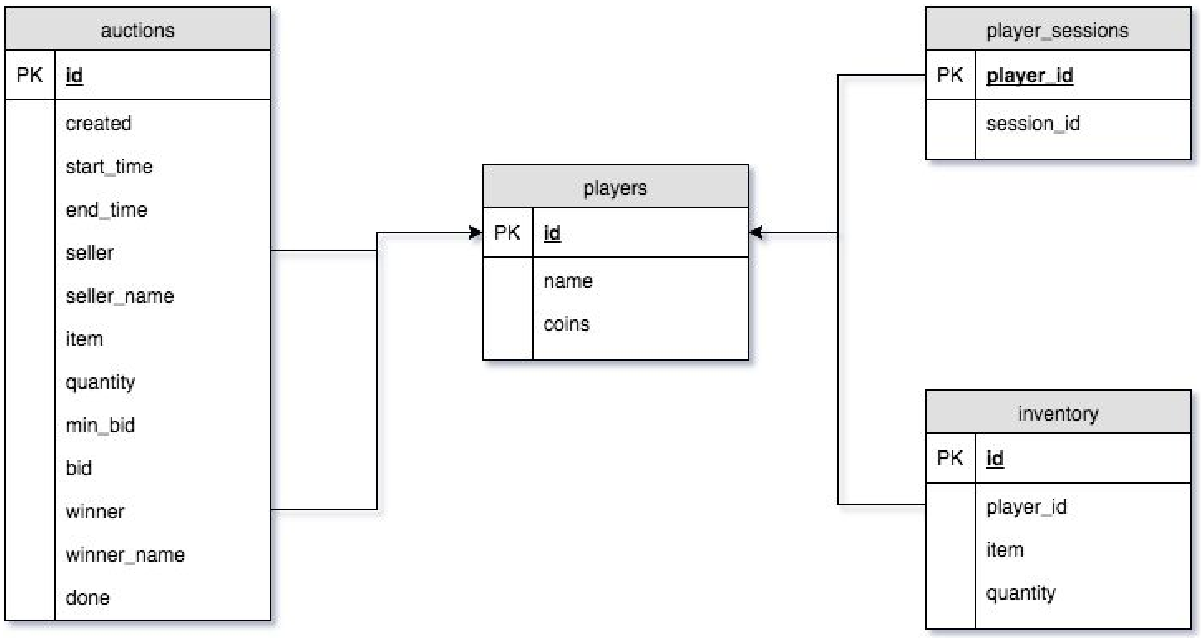

Fig 1. Auction Data Model

Architecture

The system consists of 3 layers: database, server and UI.

1. Database is a persistent layer for saving auction game state. All players, their balances, inventory and auctions are stored there. In this implementation, according to requirements, Postgres was chosen. Before first run you have to execute *schema.sql* script (attached to the solution) to create required tables.
2. Server application is a Node.js application consisting of 3 main parts:
    - HTTP API powered by Express.js framework,
    - Worker service which starts auctions from the queue, updates players and inventory after auction is finished and notifies UI about updates by Socket.io framework
    - UI static files: html, css, js, images, fonts.
3. UI or presentation layer
    - Completely independent from the server layer, HTTP server just used for serving required static files
    - MVVM powered by Angular.js
    - Meets Google Material design standards with help of Angular Material framework
    - Mobile friendly

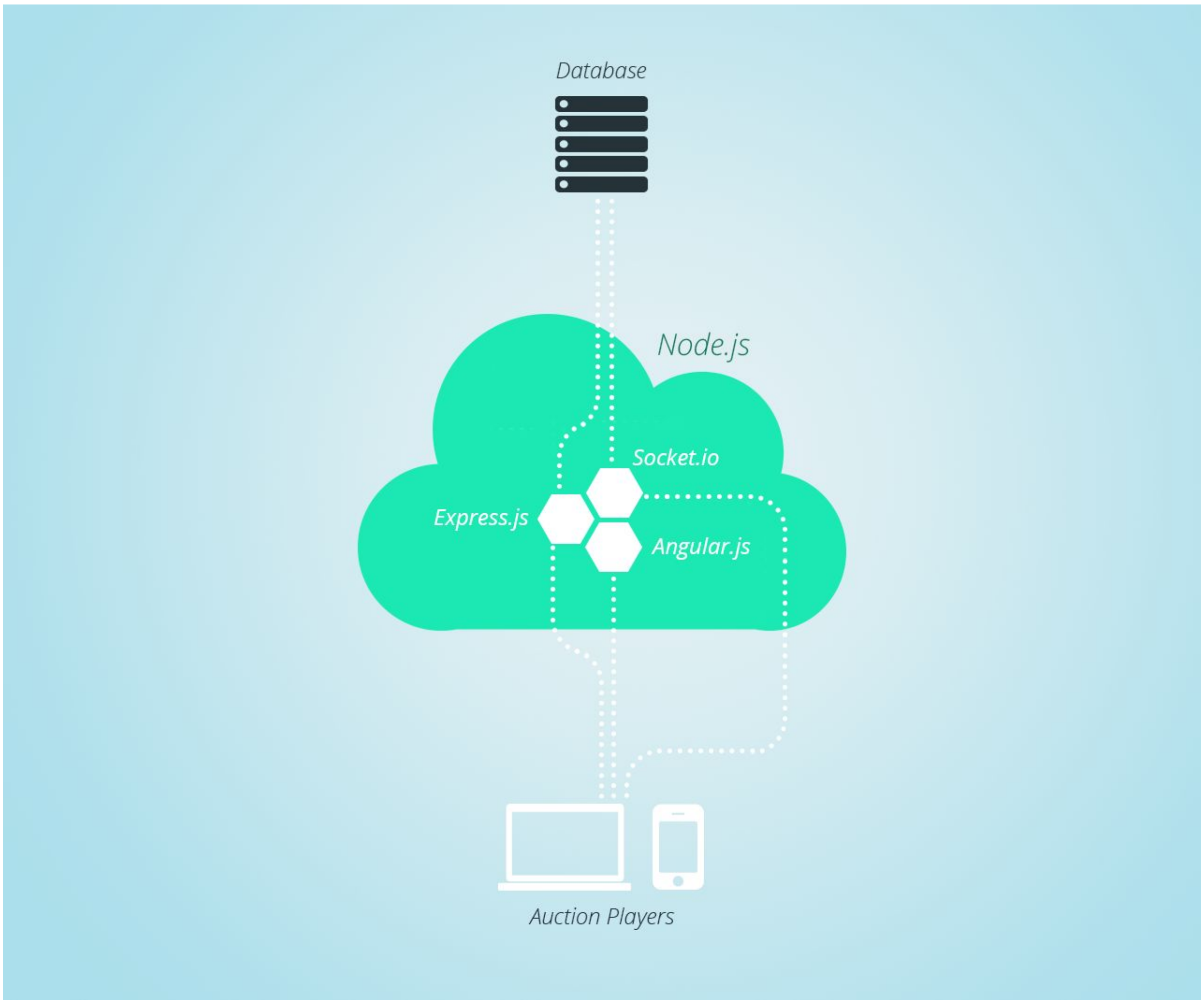

Fig 2. Auction Architecture Overview

When user opens the application in web browser or mobile client:

- Angular.js UI is downloaded to the client and *Login Form* appears. Application automatically connects to Socket.io to emit and receive events to/from the server.
- User should type his name in the *Input Field* and press the *Button.*
- Ajax request is executed against *Express.js Login API* and new *JWT* token is returned to the client. For all subsequent requests this token will be used for authentication. *User* becomes auction *Player* now.
- Player can see his balance as a number of coins (1000 by default), his default inventory and current auction if there is one.
- Player can sell something by specifying quantity and minimum bid for the inventory item. New auction will be added to the queue.
- When new auction started, updated or completed *Angular.js* application receives special events from the server which cause required UI updates.
- *Express.js HTTP API* and *Worker* service built on *Node.js* and *Socket.io* are connected to the same database.