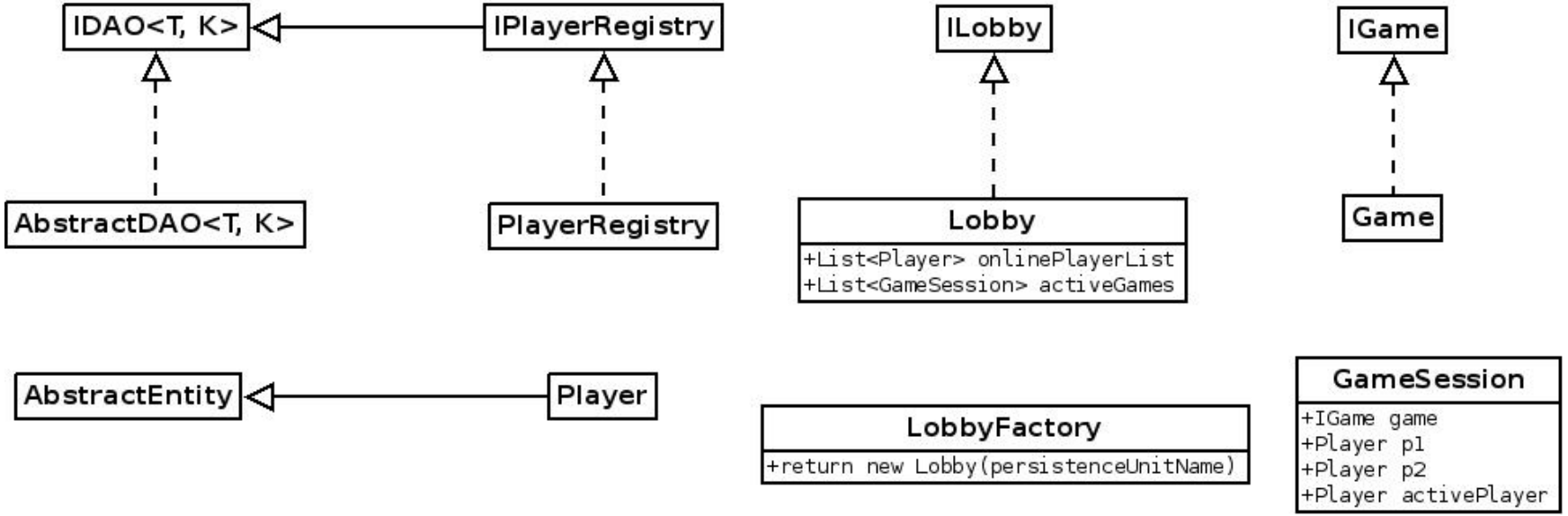
WebTicTacToe

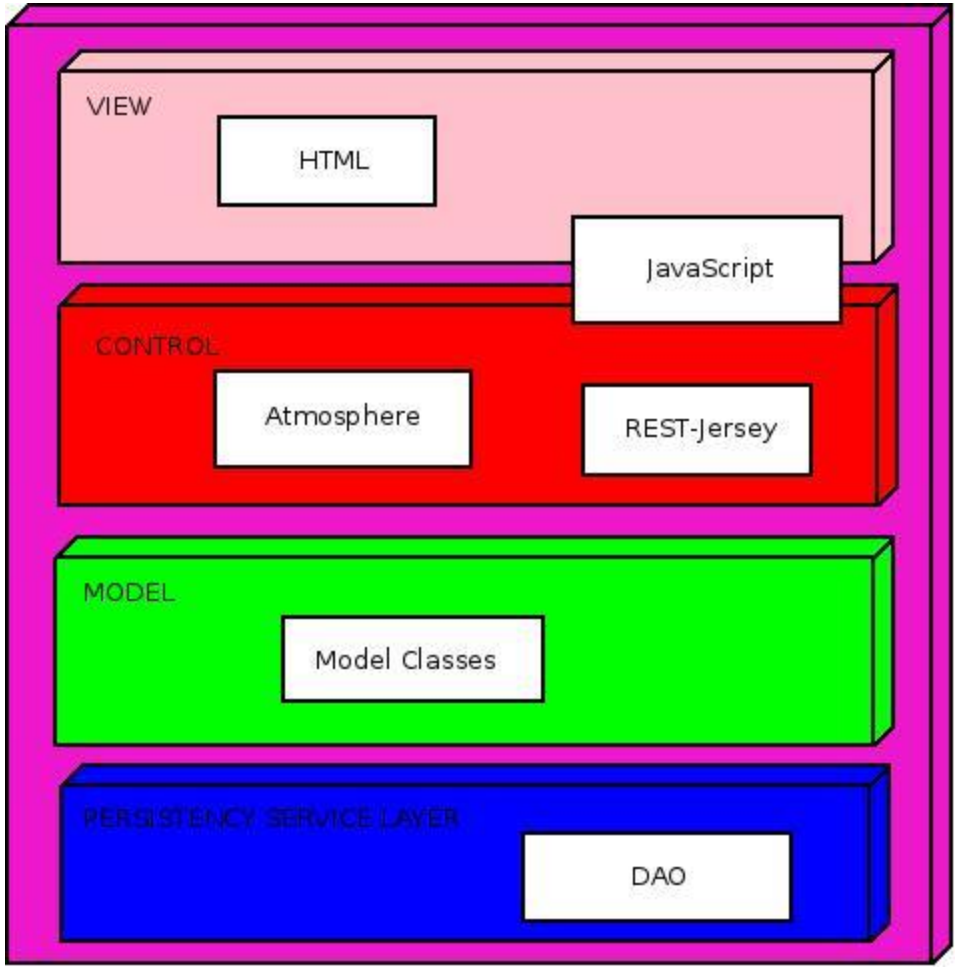
Online, Realtime Multiplayer Tic-Tac-Toe

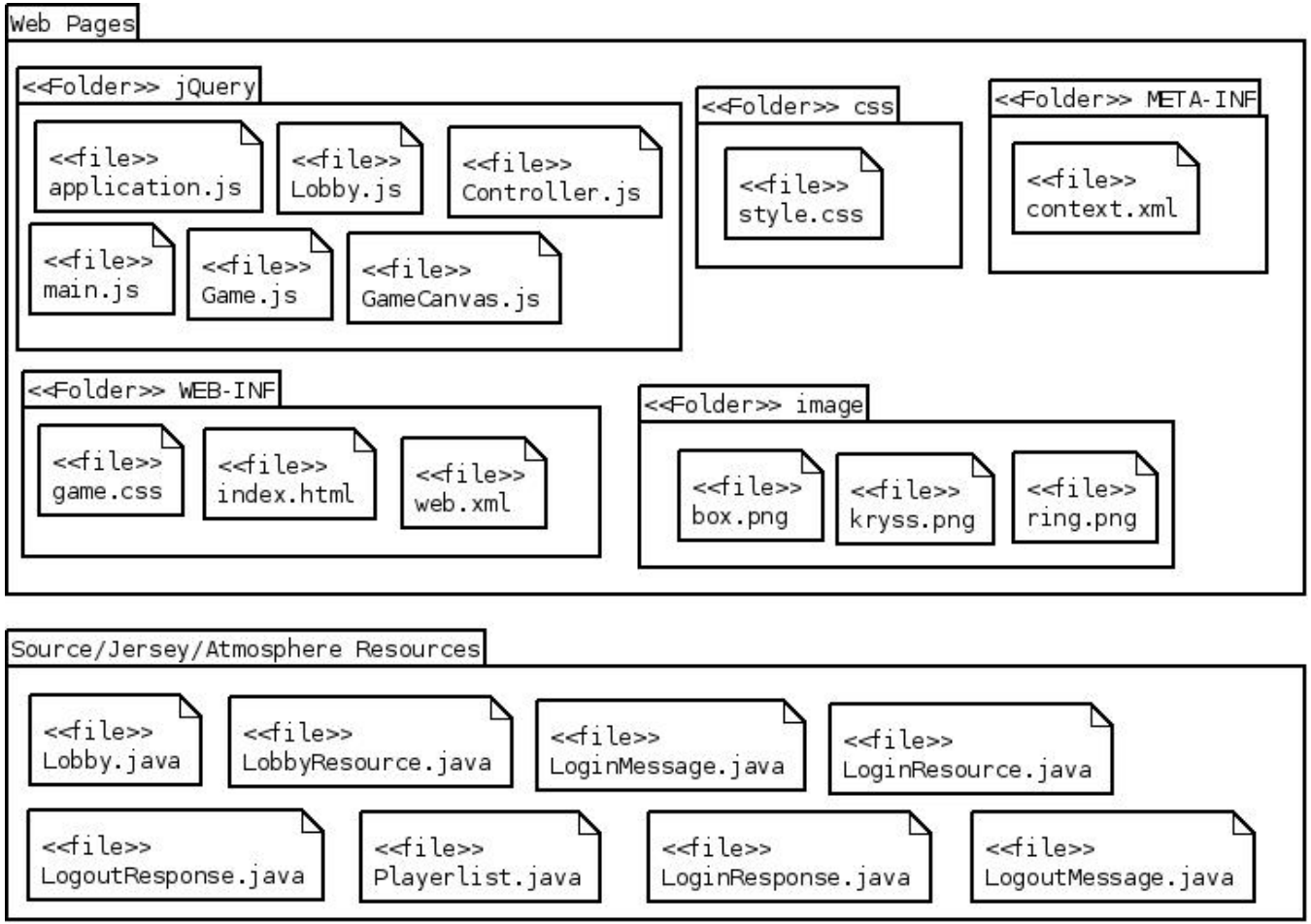
Background

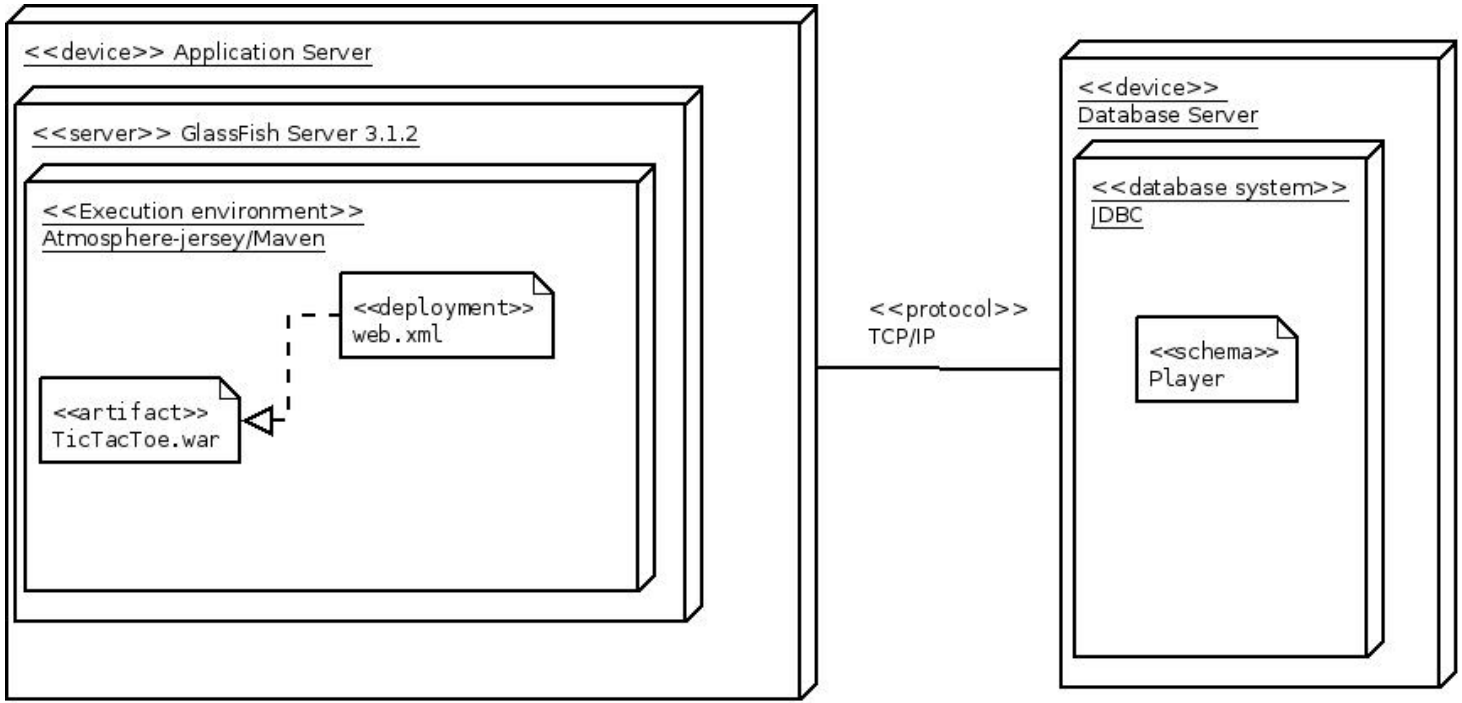
- Real Time
- Game
- Basic
- Tools

MODEL


IDAO<T, K>
IPlayerRegistry
ILobby
IGame
AbstractDAO<T, K>
PlayerRegistry
Lobby
+List<Player> onlinePlayerList
+List<GameSession> activeGames
Game
AbstractEntity
Player
LobbyFactory
+return new Lobby(persistenceUnitName)
GameSession
+IGame game
+Player p1
+Player p2
+Player activePlayer


VIEW
HTML
JavaScript
CONTROL
Atmosphere
REST-Jersey
MODEL
Model Classes
PERSISTENCY SERVICE LAYER
DAO


Web Pages
<<Folder>> jQuery
<<file>> application.js
<<file>> Lobby.js
<<file>> Controller.js
<<file>> main.js
<<file>> Game.js
<<file>> GameCanvas.js
<<Folder>> css
<<file>> style.css
<<Folder>> META-INF
<<file>> context.xml
<<Folder>> WEB-INF
<<file>> game.css
<<file>> index.html
<<file>> web.xml
<<Folder>> image
<<file>> box.png
<<file>> kryss.png
<<file>> ring.png
Source/Jersey/Atmosphere Resources
<<file>> Lobby.java
<<file>> LobbyResource.java
<<file>> LoginMessage.java
<<file>> LoginResource.java
<<file>> LogoutResponse.java
<<file>> Playerlist.java
<<file>> LoginResponse.java
<<file>> LogoutMessage.java


<<device>> Application Server
<<server>> GlassFish Server 3.1.2
<<Execution environment>>
Atmosphere-jersey/Maven
<<deployment>>
web.xml
<<artifact>>
TicTacToe.war
<<protocol>>
TCP/IP
<<device>>
Database Server
<<database system>>
JDBC
<<schema>>
Player

Atmosphere / Jersey

```
@POST
@Consumes("application/json")
@Produces("application/json")
@Path("/{id}")
public Response broadcastGamestate(@PathParam(value  = "id") Broadcaster
id, GameMessage gameMessage) {
    GameSession gameSession = gameSessionMap.get(id.getID());
```

Broadcaster

```
// Generate a new UUID.
UUID uuid = UUID.randomUUID();
// Map the new gamesession to the new uuid.
gameSessionMap.put(uuid.toString(), gameSession);
// Broadcast the UUID to the suspended requests that match both players names.
BroadcasterFactory.getDefault()
        .lookup(gameSession.getPlayerOne().getName())
        .broadcast new UUIDMessage(uuid.toString(), size, gameSession.getActivePlayer().getName()));
BroadcasterFactory.getDefault()
        .lookup(gameSession.getPlayerTwo().getName())
        .broadcast new UUIDMessage(uuid.toString(), size, gameSession.getActivePlayer().getName()));
```

```
// Try to make a move to the given position with the given player.
Boolean successfulMove = gameSessior.move(gameMessage.xPos, gameMessage.yPos, givenPlayer);
// If the move is successful, we convert the new gameboard to a response-friendly format.
if (successfulMove) {
    // Broadcast the name of the active player, the state of the board and, if available, the name of the
winner.
    if (gameSessior.getWinner() == null) {
        id.broadcast(new GameResponse(gameSessior.getActivePlayer().getName(), gameSessior.getBoard(),
"Undecided"));
    } else {
        // A player has won.
        id.broadcast(new GameResponse(gameSessior.getActivePlayer().getName(), gameSessior.getBoard(),
gameSessior.getWinner().getName()));
        gameSessionMap.remove(id.getID());
    }
    // The move was successful, so we return an OK response.
    return Response.ok().build();
}
```

Lobby.js - Atmosphere @ frontend

```
playerNameRequest = { url: baseuri + '/player/' + $.cookie('name'),
        contentType : "application/json",
        logLevel : 'debug',
        transport : 'long-polling' ,
        trackMessageLength : true};

playerNameRequest.onOpen = function (response) {
    console.log('connected to playernamerequest);
};
playerNameRequest.onMessage = function (response) {
      // ...
};

// Finally we subscribe to the request.
playerNameSocket = $.atmosphere.subscribe(playerNameRequest);
```

```
playerNameRequest.onMessage = function (response) {
    var message = response.responseBody;

    try {
        var json = $.parseJSON(message);
    } catch (e) {
        console.log('This doesn\'t look like a valid JSON: , message);
        return;
    }

    gameController.startGame(baseuri, json.uuid, jsorsize, json.startingPlayerName);
};
```

DEMO
Questions?