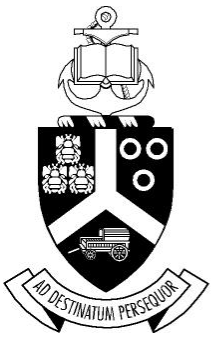


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS 301 - Software Engineering

OnlyRugby
Architectural Requirements

Authors:	*Student number:*
Herman Keuris	u13037618
Johan van Rooyen	u11205131
Estian Rosslee	u12223426
Ivan Henning	u13008219
Muller Potgieter	u12003672

February 17, 2016

Contents

1 Introduction

This section deals with the software architecture requirements of the OnlyRugby App. This includes:

- The architectural scope.
- Quality requirements.
- The integration and access channel requirements.
- The architectural constraints.
- Architectural patterns and styles used.
- The architectural tactics and strategies used.
- The use of reference architectures and frameworks.
- Access and integration channels.
- Technologies used.

2 Architecture Requirements

bla blalab aegea ageea gsaESEG EGgsa

3 Architecture Patterns or Styles

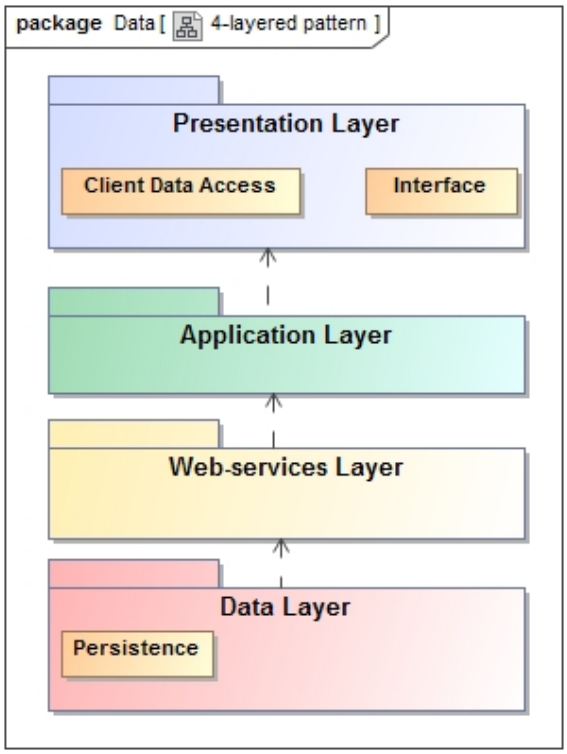
3.1 Layered Architectural Strategy

The OnlyRugby application will make use of a 4-tier layered pattern, as explained below:

1. Presentation Layer: The Presentation Layer is comprised of two main aspects:
   (a) Interface: Provides an interface/front-end through which users/clients can access and interact with the Application Layer.
   (b) Client Data Access: Captures the client's input and and passes it on to the Application Layer. Also responsible for validating input (but not for authentication).
2. Application Layer:
   - Provides back-end services of the system (i.e. all functions and forms of data processing/manipulation).
   - Manages access to the web-services layer.
   - Manages client login authentication.
   - Manages persistence to the main database.
   - If the app can not currently reach the database then this layer will provide a space to temporarily store all relevant data which needs to be uploaded to the database at a later time.
3. Web-services Layer: This layer is where the server is situated.
   - Receives the processed client input from the Application Layer and adds it to the database (i.e. allows reading from- and writing to database).
   - Provides other server-side computation/services such as:
     – Conflict resolution when two different clients upload contradicting match statistics.
     – Sending emails to clients (for example when registering with OnlyRugby).
4. Data Layer: This layer is where the database is situated.
   - Stores all added information, even when the client is not communicating with the database (i.e. persists data).

   Reasons for choosing a layered architecture:
   - Complexity is reduced by abstracting and separating a number of well defined layers which are then weakly coupled with each other. This allows the responsibilities of the system to be divided between the different layers and can prevent certain aspects of the system from becoming too dependant on (or needlessly intertwined with) another aspect of the system.

- It allows for the possibility of layers, at times, being re-used across the system (for example the Presentation Layer relies on the Application Layer to verify all input whereas the Web-services Layer relies on the Application Layer to format the input so that it could be stored in the database).
- Separating certain aspects (such as interface and implementation) makes it easier to separate test different parts of the system.
- The reduced dependency of various parts of the system on each other will also improve maintainability as it allows parts of the system to be updated without requiring any unnecessary changes to other parts of the system.



4 Architectural strategies and tactics

1. Authorization: The user's profile on the application is authorized against the one on the database. JSON web tokens are passed between the application and the database. These tokens are very small, but can contain all the relevant data that needs to be passed. Making use of these tokens enables the database not to be burdened with keeping sessions. These tokens are widely used, as they can be easily authorized and encrypted.

   The quality requirements addressed:

   - Performance: Because the tokens are small, thus it's easy and quick to process.
   - Security: By encrypting the tokens and authorizing the user, private information can be protected.
   - Auditability and Testability: Sending the information in these tokens will make it easier to see if the correct data was passed.

2. Queuing and scheduling: Queuing and scheduling can be used to maximize the performance of the application and the server. As new requests are received, they are placed into a queue and sequentially processed. This will assure fairness and that requests are processed in order. This will also be used if a user does not have internet access, but wishes to load new statistics. The data will be queued, until it can be sent.

   The quality requirements addressed:

   - Availability, by optimizing the processing of the requests.
   - Reliability and Auditability. This way, the order in which requests are processed can be guaranteed.
   - Auditability and Testability: Sending the information in these tokens will make it easier to see if the correct data was passed.

3. Ping / Echo: Ping is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network and to measure the round-trip time for messages sent from the originating host to a destination computer and back. By having the application send these pings on a regular basis, perhaps daily and when the user logs in, to the database, it can ensure that it still has access to the central server and that it is up to date with both the latest statistics, as well as the newest version of the application.

   The quality requirements addressed:

   - Security. As it can ensure that all its security policies are up to date and that it can report any faults/errors.
   - Availability. It will inform the user if it was unable to reach the server.
   - Reliability. Any issues that arise may be reported and it ensures that a connection is available.
   - Monitorability and Audibility. It can log faults detected.

- Testability. It will allow us to test the connection to the server and that it can make use of said connection.

4. Message integrity: We will be exchanging JSON objects between the server and the application. These files will include any new data that either party may need. By including a checksum, these files can be checked for any errors that may have occurred during transportation. Should it find a checksum error, a request will be send and a new JSON object will be transmitted.

   The quality requirements addressed:

   - Security. This will help to protect the system from corrupted data that may hamper the application and/or server's functionality.
   - Reliability: This will help ensure that erroneous data is not used.
   - Monitorability and Auditability: It will allow the system to test if the application and/or server occasionally sends erroneous data and address the issue.

5. Multi-Threading: By using multiple threads, the application can process a larger amount of data simultaneously. While the user may be busy updating a game's statistics, the application will assign another thread to process any incoming requests, improving the program's overall performance. Networking tasks and queries are being done in background threads, preventing a bottleneck in the system.

   The quality requirements addressed:

   - Performance. It will be able to process more data, simultaneously.
   - Flexibility. By creating new threads, the program will be able to handle multiple tasks more easily.
   - Usability. By keeping the user unaware of any background actions that are taking place, it will be easier for them to use the app.

5 Use of reference architectures and frameworks

The reference architecture that will be used is Java Platform Enterprise Edition. This architecture was chosen because our system is based on a layered architecture. The server framework that will be used is the Laravel PHP Framework as we will have a RESTful web service.

6 Access and integration channels

6.1 Human access channels

- Must be able to access services through the OnlyRugby application installed on tablets and phones running Android 4.0+
- The user can easily use the OnlyRugby application to communicate with the server (when registering, logging on and off of their profiles) and through the server they can communicate with the MySQL database.
- Quality requirements addressed:
  - Availability
  - Usability

6.2 System access channels

The OnlyRugby App system can only be directly accessed through a web page (i.e. using http) using the RESTful API provided by the server.
Quality requirements addressed:

- Security

6.3 Integration Channel Used

6.3.1 REST - Representational State Transfer

- Uses standard HTTP and thus simpler to use.
- Allows different data formats whereas SOAP only allows XML.
- Has JSON support (faster parsing than XML).
- Better performance and scalability with the ability to cache reads.
- Protocol Independent, can use any protocol which has a standardised Uniform Resource Identifier (URI) scheme.
- Quality requirements addressed:
  - Integrability
  - Maintainability

6.4 Protocols

6.4.1 HTTP - Hypertext Transfer Protocol

- Used to respond to requests and transfer data

6.4.2 IP - Internet Protocol

- Allows communications between client and server
- In charge of sending, receiving and addressing data packets.

6.4.3 SMTP - Simple Mail Transfer Protocol

- Used to send emails (useful for registration confirmation and password reset services)
- MIME (Multi-purpose Internet Mail Extensions) which allows SMTP to send multimedia files.

6.4.4 TSL - Transport Layer Security

- Alternative to SSL.
- Newer and more secure version of SSL.

7 Technologies

7.1 Programming technologies

7.1.1 Android Studio

Android studio will be used as the IDE and the app will be backwards compatible from Android 4.0.

7.1.2 MySQL

MySQL will be used for the web server database.

7.1.3 SQLite

SQLite will be used for the device's database.

7.2 Web technologies

7.2.1 RESTful web service

Laravel PHP framework will be used to create a RESTful web API to interact with the web server database and provide data persistence