Software Architecture

Andries Bingani, 856807 Jackson Dyora, 747772
Thabiso Leeuw, 814690 Khutso Mpaketsane, 806383
Lesego Seitshiro ,792986

October 18, 2016

1 Introduction

1.1 Scope

The scope of this SAD is to depict the architecture of the online event hosting application created by the company Turnt App.

1.2 Definitions, Acronyms and Abbreviations

RUP: Rational Unified Process
UML: Unified Modeling Language
SAD: Software Architecture Document

1.3 References

1.4 Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.
Section 2: describes the use of each view
Section 3: describes the architectural constraints of the system
Section 4: describes the functional requirements with a significant impact on the architecture
Section 5: describes the most important use-case realization. Will contain the Analysis Model and the Design Model
Section 6: describes design's concurrency aspects
Section 7: describes how the system will be deployed. Will contain the Deployment Model
Section 8: describes the layers and subsystems of the application
Section 9: describes any significant persistent element. Will contain the Data Model
Section 10: describes any performance issues and constraints
Section 11: describes any aspects related to the quality of service (QoS) attributes

2 Architectural Representation

This document details the architecture using the views defined in the "4+1" model [KRU41], but using the RUP naming convention. The views used to document the Turnt App. application are:

2.1 Logical Architecture

Audience: Designers.
Area: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations.
Related Artifacts: Design model

2.2 Process view

Audience: Integrators.
Area: Non-functional requirements: describes the design's concurrency and synchronization aspects.
Related Artifacts: (no specific artifact).

2.3 Development view

Audience: Programmers.
Area: Software components: describes the layers and subsystems of the application.
Related Artifacts: Implementation model, components

2.4 Physical view

Audience: Deployment managers.
Area: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects.
Related Artifacts: Deployment model.

2.5 Use Case view

Audience: all the stakeholders of the system, including the end-users.
Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system.
Related Artifacts : Use-Case Model, Use-Case documents

3 Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture

3.1 Technical Platform

The Turnt App will be deployed onto the lamp server (the wits lamp server, as it is the only server that we have access to).

3.2 Transaction

The Turnt App is transactional, leveraging the technical platform capabilities. Transaction management model of the lamp will be reused intensively.

3.3 Security

The system must be secured, so that a customer can view events on the app. The application must implement basic security behaviours:

- Authentication: Login using at least a user name and a password
- Authorization: according to their profile, online customer must be granted or not to perform some specific actions.

For internet access, the following requirements are mandatory

- Confidentiality: sensitive data must be encrypted (passwords)
- Data integrity : Data sent across the network cannot be modified by a tier
- Auditing: Every sensitive action can be logged
- Non-repudiation : gives evidence a specific action occurred

3.4 Persistence

Data persistence will be addressed using a relational database.

3.5 Reliability/Availability (failover)

The availability of the system is a key requirement by nature, as it is a selling system. The candidate architecture must ensure failover capabilities. Targeted availability is 12/5: 12 hours a day, 5 days a week The time left (8 hours) is reserved for any maintenance activities

3.6 Performance

The login process must be under 5 seconds.

3.7 Internationalization (i18n)

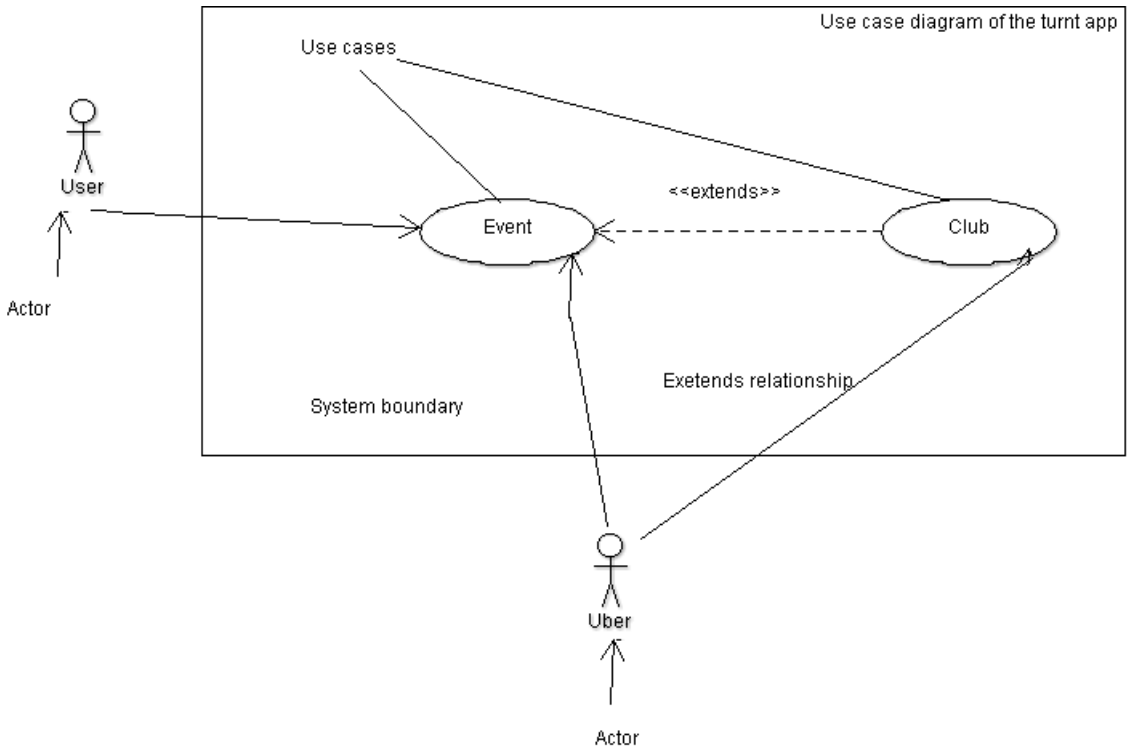
layers must be generic enough to work with any internationalization context.

4 Scenarios

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system. The only use-case studied here on the Turnt App architecture is the one related to computicket. It includes a search feature as well as a call to external services (uber options)

4.1 Events

A customer accesses the Turnt App application and search for the available events. The customer chooses from a list of events and selects which event she/he wants to attend. Then, the customer performs a computicket payment to get the tickets. Once the payment has been validated, the customer confirms attendance, enters her/his location (address, coordinates etc..) and all the relevant information is sent to the Google maps app then the directions are given.



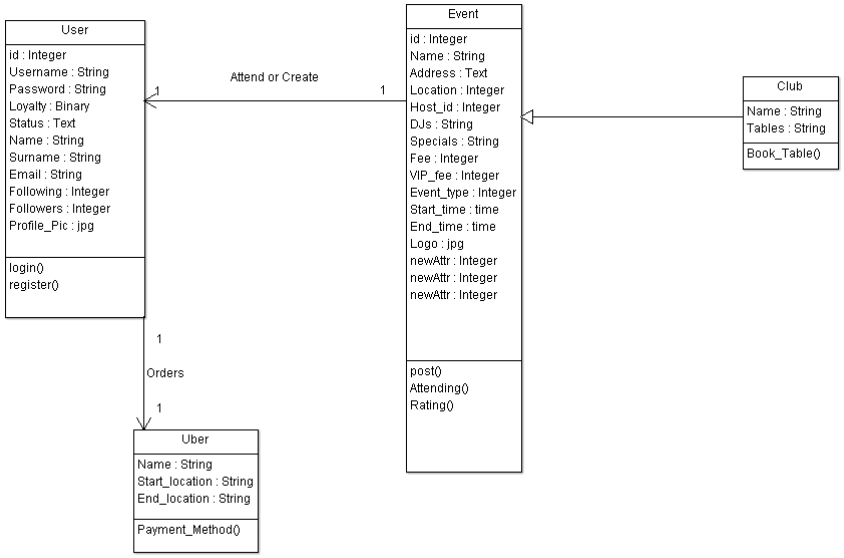
4.2 Use-Case Realizations

Refers to section 5.2 to see how design elements provide the functionalities identified in the significant use-cases

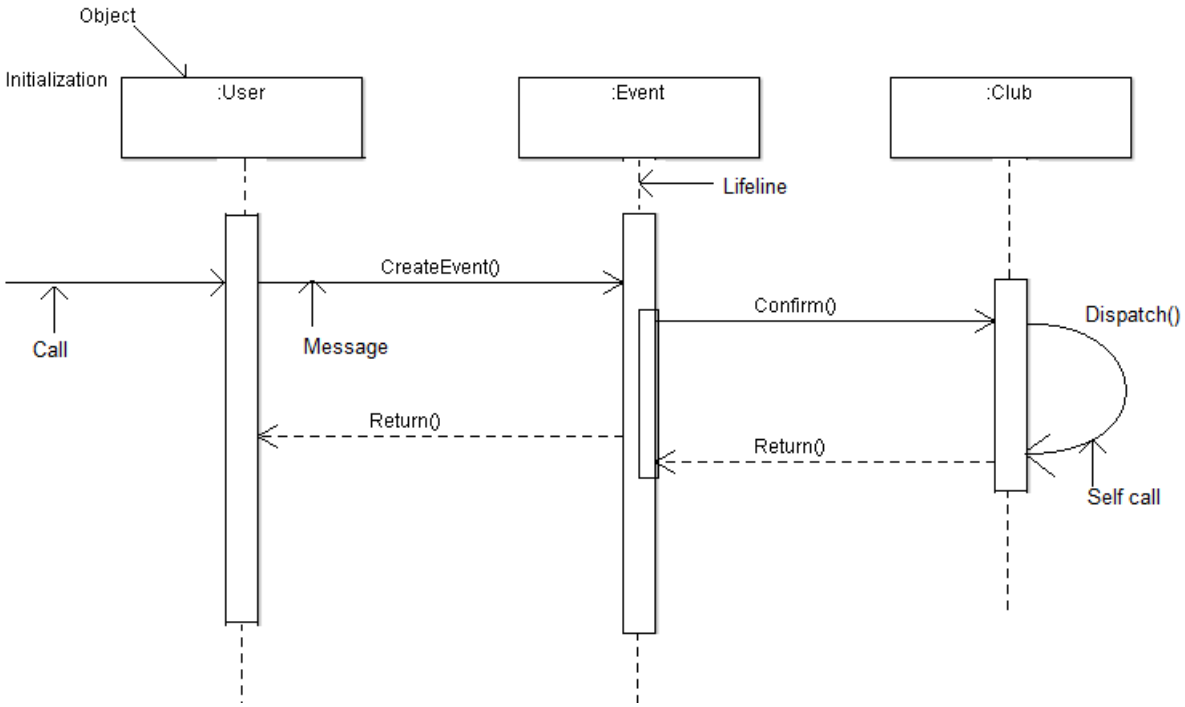
5 Logical View

5.1 Overview

The Turnt App application is divided into layers based on the N-tier architecture.
The layering model of the Turnt App application is based on a responsibility layering strategy that associates each layer with a particular responsibility. This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development and maintenance.



5.2 Architectural Layer Dependencies



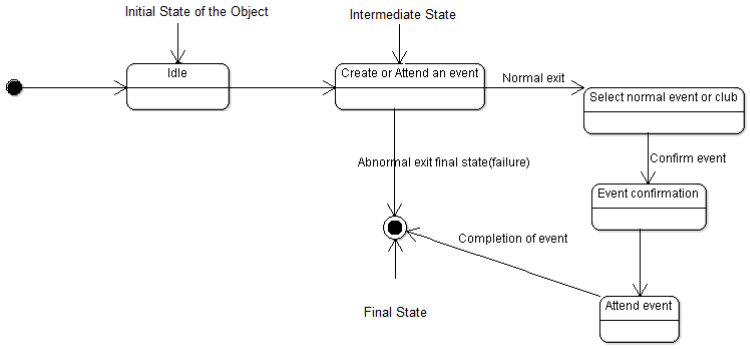
Each layer has specific responsibilities.

- The presentation layer deals with the presentation logic and the pages rendering
- The control layer manages the access to the domain layer
- The resource layer (integration layer) is responsible for the access to the enterprise information system (databases or other sources of information)
- The domain layer is related to the business logic and manages the accesses to the resource layer.
- The Common Elements layer gathers the common objects reused through all the layers

6 Process View

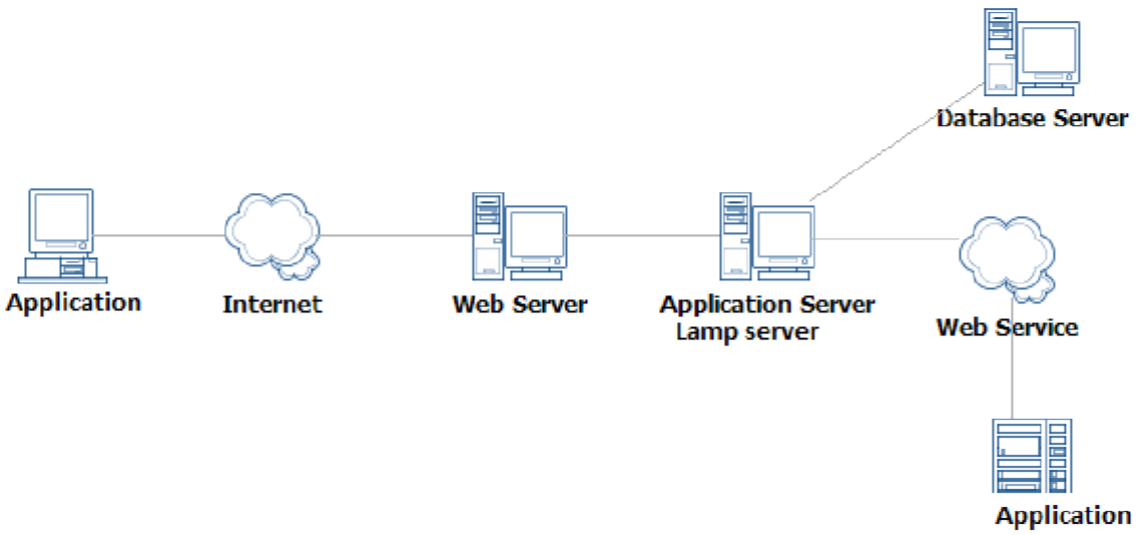
The following diagram describes the state of the program when the user creates or attends and event. The system will be in idle until the an event is created or a user selects to attend an event. Then if the action is unsuccessful then the program will move to the final state. Otherwise the program will exit normally to the select a club or normal event - where the user will decide weather they want to attend a club or a normal event. Then on confirmation, the state will

change to order confirmation. Then on to attend event. After all that is done, the program will move to the final state.



7 Physical View

The application is Android based. It is assumed to register it's users using a database stored on the lamp server. So the application would need to connect to the internet first, then be directed to web servers that are going to grant it access to the lamp server. Then from the lamp server the application will access the database. Redirect everything that it got from those databases back to the web services then back to the application.

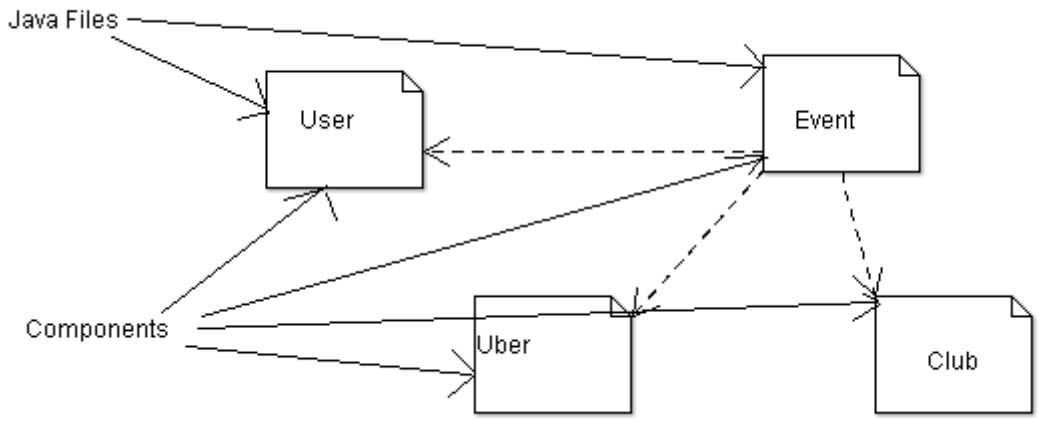


8 Development View

8.1 Overview

The Development view depicts the physical composition of the implementation in terms of Implementation Subsystems, and Implementation Elements (directories and files, including source code, data, and executable files). Usually, the layers of the Development view do fit the layering defined in the Logical view

It is unnecessary to document the Development view in great details in this document. For further information, refer to the On-line Catering Service 1.0 workspace in Rational Software Architect.



8.2 Layers

8.2.1 Presentation Layer

The Presentation layer contains all the components needed to allow interactions with an end-user. It encompasses the user interface

8.2.2 Control Layer

The Control layer contains all the components used to access the domain layer or directly the resource layer when this is appropriate.

8.2.3 Resource Layer

The Resource layer contains the components needed to enable communication between the business tier and the enterprise information systems (Database, external services, ERP, etc. . .)

8.2.4 Domain layer

The Domain layer contains all the components related to the business logic. It gathers all the subsystems that meet the needs of a particular business domain. It also contains the business object model.

8.2.5 Common Elements Layer

The Common Element layer contains the components re-used within several layers.