Università degli Studi di Napoli Federico II

Corso di Laurea in Informatica

Ingegneria del Software

Anno Accademico 2017/18

Progetto EM17

Gruppo 3
Raffaele Sorrentino N86001875
Maurizio Moraca N86001917
Riccardo Iervolino N86001608

Document’s revision table

Date	Version	Author	Description
2018/02/27	0.1	Raffaele Sorrentino	Added the table of contents and the requirement and analysis documentation
2018/02/28	1.0	Raffaele Sorrentino	Initial version

Table of contents

Sommario

Document structure

Requirements and analysis documentation

The purpose of this section is to analyze and formalize both the functional and non-functional requirements established in the stakeholder's initial instructions and in our meetings with him, mainly with the use of UML and other standard formalisms.

Requirements

This section contains a list of functional and non-functional requirements.

Use case diagrams

This section contains UML Use-Case diagrams describing all the functional requirements, a list of all types of user who can use the software and a list of functions available for each of them.

Use case descriptions

This section contains a description for each use-case, created by following the cockburn's formalism.

User interface

This section contains a graphic overview of what the software should look like and of how the user will interact with the software.

Project plan

This section contains a project plan in form of various Gantt diagrams, representing all the work that needs to be done.

BCE Class diagrams

This section contains various class diagrams representing boundary objects (UI elements that the user will use to interact with the software), control objects (the logic) and entity objects (mostly "real-life" objects that will be represented in form of code).

Sequence diagrams

This section contains various sequence diagrams, one for each function or part of a function, representing how each component of the app interacts with other components in order to complete a task.

Activity diagrams

This section contains various activity diagrams, representing everything the user has to do in order to complete a task "from the user point-of-view".

Design documentation

The purpose of this section is to define in greater detail how the system and his components will be structured. UML and other formalism are used in this section too, but each object will be represented in a more detailed way. Not every single detail has been represented, to make the diagrams as clear and easy to understand as possible, but there should be enough details to understand everything that is needed.

Architecture analysis

This section contains an overview and an explanation of the architecture used to implement the system.

Class diagrams

This section contains a greatly detailed description of all the classes and the components of the system, of their attributes and functions and of the design patterns used.

Sequence diagrams
This section contains a greatly detailed description of how each component of the system interacts with other components, hence of how each function or part of a function is completed from its beginning to its end.

State diagrams
This section contains various state diagrams, representing the different statuses an object can be in and the way the object pass from a status to another.

CRC cards
This section contains, for each class, a brief summary of the class' responsibilities and collaborators

Implementation documentation
Android application source code
This section contains the source code of the ticket inspector's app

Event manager's platform source code
This section contains the source code of the event manager's platform (backend)

Testing documentation
Test plan
This section contains a list of tests to do with the system and of what the result of each test should be

JUnit test code
This section contains the source code of the ticket inspector's app

Requirement and analysis documentation

Requirements

Functional requirements

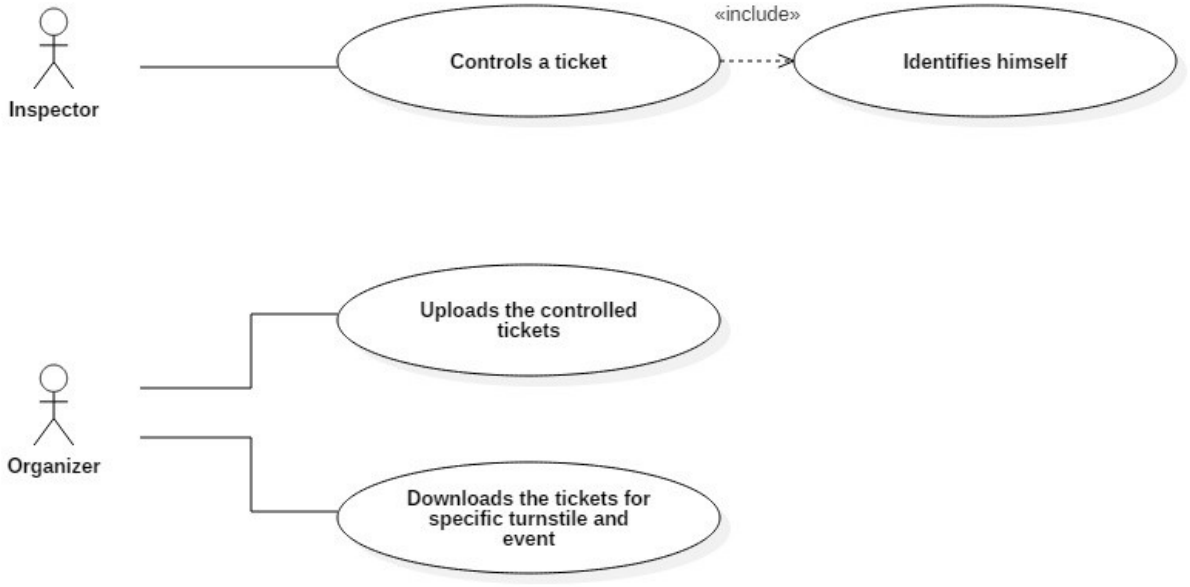
- Event manager's platform
    - The event organizer must log in to use the system
    - The event organizer must be able to add new events to the system
    - The event organizer must be able to modify existing events
        - If he does, customers who already bought a ticket must be notified
    - The event organizer must be able to delete existing events from the system
        - If he does, customers who already bought a ticket must be notified
    - The event organizer must be able to view some statistic informations for a single event and for all events together
- Ticket inspector's Android application
    - The event organizer must be able to select an event, a sector and a turnstile of its location, and to download the tickets that will be controlled from the inspector guarding that turnstile
    - The ticket inspector must be able to identify himself
        - When identified, the ticket inspector must be able to control a customer's ticket and to get an immediate visual feedback to know if the control was successful or not
    - The event organizer must be able to upload to the cloud all the tickets that were controlled since the last time he did an upload
        - For each ticket, it's needed to send to the cloud not only which ticket was controlled but also which inspector controlled it and how that inspector was identified

Non-functional requirements

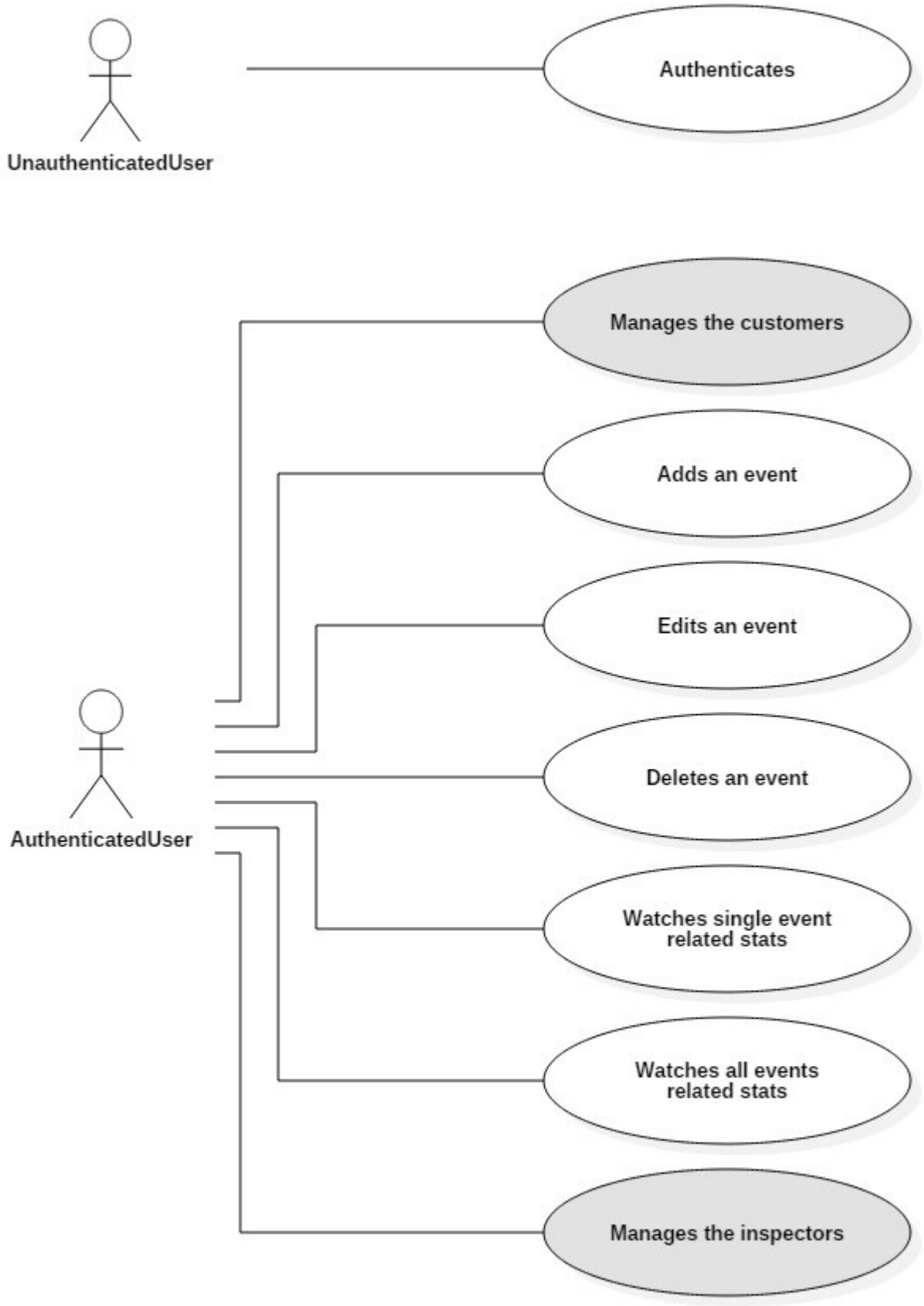
- The system must be easily interfaceable with any type of database: plain text, relational, non-relational, …
- Every single functionality (except the ticket inspector controlling a customer's ticket) needs an internet connection to work correctly
- The ticket inspector must be able to identify and control tickets while being completely offline

Use case diagrams

Android application



Event manager platform



Use case descriptions

Android application

USE CASE #1	Identifies himself		
GOAL IN CONTEXT	The inspector wants to identify himself		
BLACK BOX	//		
LEVEL	Subfunction		
PRECONDITIONS	//		
SUCCESS END CONDITION	The inspector logs into the control making screen		
FAILED END CONDITION	//		
PRIMARY ACTOR	Inspector		
TRIGGER	User clicks on “Control tickets” in “Main”		
DESCRIPTION	STEP #...	ACTOR	SYSTEM
	1.		Shows “ControlMaking_Identification”
	2.	Inserts his personal data	
	3.	Clicks on the continue button	
	4.		Shows “ControlMaking_Scanning”
SUBORDINATES	UC2_Controls a ticket		

USE CASE #2	Controls a ticket		
GOAL IN CONTEXT	The inspector controls the tickets		
BLACK BOX	QR code reader		
LEVEL	Primary task		
PRECONDITIONS	The inspector is identified		
SUCCESS END CONDITION	The app returns to the previous screen (“ControlMaking_Scanning”)		
FAILED END CONDITION	//		
PRIMARY ACTOR	Inspector		
TRIGGER	The inspector identifies himself and clicks on the confirm button (check UC #1)		
DESCRIPTION	STEP #...	ACTOR	SYSTEM
	1.	Scans the QR code	
	2.		Shows “ControlMaking_Success” along with ticket data
	3.	Clicks on the ok button	
	4.		Shows “ControlMaking_Scanning”
SUBVARIATION #1 Ticket is not valid or it refers to another turnstile	STEP #...	ACTOR	SYSTEM
	1.2		Shows “ControlMaking_Failure”
	1.3	Clicks on the ok button	
	1.4		Shows “ControlMaking_Scanning”
Superordinates	UC1_Identifies himself		

USE CASE #3	Downloads the tickets for specific turnstile and event		
GOAL IN CONTEXT	The organizer downloads the tickets that will be validated by an inspector		
BLACK BOX	//		
LEVEL	Primary task		
PRECONDITIONS	//		
SUCCESS END CONDITION	Tickets are downloaded and the app shows the main menu (“Main”)		
FAILED END CONDITION	The organizer sees an error message		
PRIMARY ACTOR	Organizer		
TRIGGER	The organizer clicks “Download tickets” in “Main”		
DESCRIPTION	STEP #...	ACTOR	SYSTEM
	1.		Shows “DownloadTickets_SelectEvent”
	2.	Inserts the event id	
	3.	Clicks on “Confirm”	
	4.		Shows “DownloadTickets_SelectSector”
	5.	Selects the sector	
	6.		Shows “DownloadTickets_SelectTurnstile”
	7.	Selects the turnstile	
	8.		Shows “DownloadTickets_ConfirmSelection”
	9.	Clicks on “Confirm”	
	10.		Shows “DownloadTickets_Downloading”
	11.		When download is completed, shows “Main”
SUBVARIATION #1 Event id is not valid	STEP #...	ACTOR	SYSTEM
	1.4		Shows “DownloadTicket_PriorError”
	1.5		Go to the step #2 of the main flow
SUBVARIATION #2 No connection or not enough free memory	STEP #...	ACTOR	SYSTEM
	2.10		Shows “DownloadTicket_Error”
	2.11		Go to the step #4 of the main flow
SUBVARIATION #3 Lost connection during download	STEP #...	ACTOR	SYSTEM
	3.11		Shows “DownloadTicket_Error”
	3.12		Go to the step #4 of the main flow

USE CASE #4	Uploads the controlled tickets		
GOAL IN CONTEXT	The organizer uploads the controls done by the inspector		
Black Box	//		
LEVEL	Primary task		
PRECONDITIONS	//		
SUCCESS END CONDITION	Controls are uploaded and the app goes back to the main menu (“Main”)		
FAILED END CONDITION	The organizer sees an error message		
PRIMARY ACTOR	Organizer		
TRIGGER	The organizer clicks on “Upload controls” in “Main”		
DESCRIPTION	STEP #...	ACTOR	SYSTEM
	1.		Shows “UploadControls_Confirm”
	2.	Clicks on “Confirm”	
	3.		Shows “UploadControls_Uploading”
	4.		When upload is completed, shows “Main”
SUBVARIATION #1 No connection	STEP #...	ACTOR	SYSTEM
	1.3		Shows “UploadControls_ConnectionError”
SUBVARIATION #2 Lost connection during the upload	STEP #...	ACTOR	SYSTEM
	2.4		Shows “UploadControls_ConnectionError”

Event manager's platform

USE CASE #1	Authenticates		
GOAL IN CONTEXT	An unauthenticated user wants to authenticate so that he can access the service		
Black Box	//		
LEVEL	Primary task		
PRECONDITIONS	//		
SUCCESS END CONDITION	The user becomes able to access the various software functionalities		
FAILED END CONDITION	The user sees an error message		
PRIMARY ACTOR	Unauthenticated User		
TRIGGER	The user starts the software		
DESCRIPTION	STEP #...	ACTOR	SYSTEM
	1.	Writes his username and his password	
	2.	Clicks on the confirm button	
	3.		Shows "EventManager"
SUBVARIATION #1 User credentials are not valid	STEP #...	ACTOR	SYSTEM
	1.3		Shows "Login_Error"

USE CASE #2	Adds an event		
GOAL IN CONTEXT	The operator wants to add an event		
Black Box	//		
LEVEL	Primary task		
PRECONDITIONS	The operator must be authenticated		
SUCCESS END CONDITION	The event is added and the operator is presented with the all events summary (“EventManager”)		
FAILED END CONDITION	//		
PRIMARY ACTOR	AuthenticatedUser		
TRIGGER	From “EventManager” the user clicks on the add event button		
DESCRIPTION: Adds an event in an existing location	STEP #...	USER	SYSTEM
	1.		Shows “AddEditEvent_BasicInformations”
	2.	Inserts the event basic informations	
	3.	Clicks on “Next”	
	4.		Shows “AddEditEvent_ChoosePrice”
	5.	Specify the price for each sector	
	6.	Clicks on “Done”	
	7.		Shows the updated “EventManager”
SUBVARIATION #1 Adds an event in a new location or in an existing unused location (by unused, we mean that the location has no event associated with it)	STEP #...	USER	SYSTEM
	4.1.		Shows “AddEditEvent_SectorSelection”
	4.2.	Fills the add sector form and submit	
	4.3.		Updates the table on the upper left
	4.4.	Selects a sector	
	4.5.		Shows “AddEditEvent_TurnstileSelection”
	4.6.	Fills the add turnstile form and submit	
	4.7.		Updates the table on the upper right
	4.8.	Selects a turnstile	
	4.9.		Shows “AddEditEvent_TurnstileSelected”
	4.10.	Clicks on “Next”	
	4.11.		Go to the step #4 of the main flow

USE CASE #3	Edits an event		
GOAL IN CONTEXT	The user wants to modify an existing event		
Black Box	//		
LEVEL	Primary task		
PRECONDITIONS	The user must be authenticated		
SUCCESS END CONDITION	The event is modified and the operator is presented with the all events summary (“EventManager”)		
FAILED END CONDITION	//		
PRIMARY ACTOR	AuthenticatedUser		
TRIGGER	From “EventManager” the user clicks on the modify event button for an event		
DESCRIPTION	STEP #	USER	SYSTEM
	1.		Shows “AddEditEvent_BasicInformations”
	2.	Inserts the event basic informations (name, description and location can’t be modified)	
	3.	Clicks on “Next”	
	4.		Shows “AddEditEvent_ChoosePrice”
	5.	Specify the price for each sector (none can be modified if there is at least one sold ticket)	
	6.	Clicks on “Done”	
	7.		Shows the updated “EventManager”

USE CASE #4	Deletes an event		
GOAL IN CONTEXT	The user wants to delete an existing event		
Black Box	//		
LEVEL	Primary task		
PRECONDITIONS	The user must be authenticated		
SUCCESS END CONDITION	The event is deleted and the operator is presented with the all events summary (“EventManager”)		
FAILED END CONDITION	//		
PRIMARY ACTOR	AuthenticatedUser		
TRIGGER	From “EventManager” the user clicks on the delete event button for an event		
DESCRIPTION	STEP #...	ACTOR	SYSTEM
	1.		Shows “EventManager_DeletionRequested”
	2.	The user clicks on “Yes”	
	3.		Shows “EventManager”

USE CASE #5	Watches single event related stats		
GOAL IN CONTEXT	The user wants to see all the stats related to a specific event		
BLACK BOX	//		
LEVEL	Primary task		
PRECONDITIONS	The user must be authenticated		
SUCCESS END CONDITION	The user watches the statistics he is interested to		
FAILED END CONDITION	//		
PRIMARY ACTOR	AuthenticatedUser		
TRIGGER	From “EventManager” the user clicks on the view stats button for an event		
DESCRIPTION	STEP #...	ACTOR	SYSTEM
	1.		Shows “SingleEventStats”

USE CASE #6	Watches all events related stats		
GOAL IN CONTEXT	The user wants to see all the stats related to all events		
BLACK BOX	//		
LEVEL	Primary Task		
PRECONDITIONS	The user must be authenticated		
SUCCESS END CONDITION	The user watches the statistics he is interested to		
FAILED END CONDITION	//		
PRIMARY ACTOR	AuthenticatedUser		
TRIGGER	The user clicks on the “Statistics” tab of the leftmost menu		
DESCRIPTION	STEP #...	ACTOR	SYSTEM
	1.		Shows “AllEventsStats”

User interface

Android application

Event manager’s platform

Project plan

Requirements and analysis

Nome attività	Durata	Predecessori	Nomi risorse
Organizzazione raccolta dei requisiti	**1,25 g**		
Creazione diagramma di Gantt per la raccolta dei requisiti	8 h		Raffaele Sorrentino
Revisione diagramma di Gantt per la raccolta dei requisiti	2 h	2	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Use cases app Android	**1,5 g**	**1**	
Definire l'use cases diagram dell'app Android	3 h		Maurizio Moraca
Dettagliare ogni use case diagram dell'app Android in una tabella di Cockburn	6,5 h	5	Riccardo Iervolino;Maurizio Moraca
Revisione use cases dell'app Android	1 h	5;6	Maurizio Moraca;Riccardo Iervolino;Raffaele Sorrentino
Dettagli use cases app Android	**2,19 g**	**4**	
Creare gli activity diagrams per ogni caso d'uso dell'app Android per cui è necessario	2,5 h		Maurizio Moraca;Riccardo Iervolino
Creare i sequence diagrams per ogni caso d'uso dell'app Android	6,5 h		Maurizio Moraca;Riccardo Iervolino
Revisione dettagli use cases dell'app Android	2 h	9;10	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Use cases del programma di backend	**4,06 g**	**1**	
Definire l'use cases diagram del programma di backend	3 h		Riccardo Iervolino
Dettagliare ogni use case diagram del programma di backend in una tabella di Cockburn	6,5 h	13	Riccardo Iervolino;Maurizio Moraca
Revisione use cases del programma di backend	1 h	13;14	Raffaele Sorrentino;Riccardo Iervolino;Maurizio Moraca
Dettagli use cases del programma di backend	**1,38 g**	**12**	
Creare gli activity diagrams per ogni caso d'uso del programma di backend per cui è necessario	2,5 h		Riccardo Iervolino;Maurizio Moraca
Creare i sequence diagrams per ogni caso d'uso del programma di backend	6,5 h		Riccardo Iervolino;Maurizio Moraca
Revisione dettagli use cases del programma di backend	2 h	17;18	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Mockups	**1,38 g**	**1**	
Mockups dell'app Android	5 h	1	Raffaele Sorrentino
Mockups del programma di Backend	5 h		Raffaele Sorrentino

Revisione mockups	1 h	21;22	Maurizio Moraca;Riccardo Iervolino;Raffaele Sorrentino
Class diagram app Android	**4,94 g**	**4**	
Definizione classi con relazioni, attributi e metodi indicativi, dell'app Android	13 h		Raffaele Sorrentino
Revisione class diagram dell'app Android	2 h	25	Raffaele Sorrentino;Riccardo Iervolino;Maurizio Moraca
Definizione dei vincoli OCL dell'app Android dove necessari	2 h	26	Maurizio Moraca;Riccardo Iervolino
Creare uno state diagram per le classi dell'app Android per cui è necessario farlo	3 h	26	Maurizio Moraca;Riccardo Iervolino
Revisione vincoli OCL e state diagram dell'app Android	1 h	26;27;28	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Class diagram programma di backend	**3,13 g**	**12**	
Definizione classi con relazioni, attributi e metodi indicativi, del programma di backend	13 h		Raffaele Sorrentino
Revisione class diagram del programma di backend	2 h	31	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Definizione dei vincoli OCL dove necessari del programma di backend	2 h	32	Riccardo Iervolino;Maurizio Moraca
Creare uno state diagram per le classi del programma di backend per cui è necessario farlo	3 h	32	Riccardo Iervolino;Maurizio Moraca
Revisione vincoli OCL e state diagram del programma di backend	1 h	32;33;34	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Creazione diagramma di Gantt per la fase di design	**1 g**	**1**	
Creazione diagramma di Gantt per il design dell'app Android	8 h	4;8;23;26	
Creazione diagramma di Gantt per il design del programma di backend	8 h	12;16;23;32	
Revisione diagrammi di Gantt	2 h	37;38	
Revisione generale della fase di raccolta dei requisiti	**0,19 g**	**1;4;8;12;16;20;24;30;36**	
Inserimento use cases nel documento dei requisiti	1 h		Riccardo Iervolino;Maurizio Moraca;Raffaele Sorrentino
Inserimento mockups nel documento dei requisiti	1 h		Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Inserimento class diagrams nel documento dei requisiti	1 h		Raffaele Sorrentino;Maurizio Moraca;Riccardo Iervolino

Design

Nome attività	Durata	Predecessori	Nomi risorse
Organizzazione fase di design	**0,75 g**		
Creazione diagramma di Gantt per la fase di design	5 h		Raffaele Sorrentino
Revisione diagramma di Gantt per la fase di design	1 h	2	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Definizione dell'architettura dell'app	**0,88 g**	**1**	
Individuazione di una libreria per la lettura di codici QR	1 h		Riccardo Iervolino
Individuazione di una libreria per la lettura di impronte digitali	3 h		Riccardo Iervolino
Definizione dell'architettura dell'app vera e propria	3 h	5	Raffaele Sorrentino
Creazione class diagrams app	**0,88 g**	**4;1**	
Creazione class diagram per l'effettuazione dei controlli	5 h		Raffaele Sorrentino
Creazione class diagram per lo scaricamento dei biglietti	5 h		Riccardo Iervolino
Creazione class diagram per il caricamento dei biglietti	5 h		Maurizio Moraca
Revisione ed unione dei class diagrams dell'app	2 h	9;10;11	Maurizio Moraca;Riccardo Iervolino;Raffaele Sorrentino
Creazione sequence diagrams app	**1,13 g**	**1;4**	
Creazione sequence diagram per l'identificazione del controllore	3 h		Raffaele Sorrentino
Creazione sequence diagram per l'effettuazione dei controlli	4 h		Raffaele Sorrentino
Creazione sequence diagram per lo scaricamento dei biglietti	6 h		Riccardo Iervolino
Creazione sequence diagram per il caricamento dei biglietti	6 h		Maurizio Moraca
Revisione sequence diagrams dell'app	2 h	15;16;17;14	Maurizio Moraca;Riccardo Iervolino;Raffaele Sorrentino
Definizione dell'architettura del programma di backend	**2,75 g**	**1**	
Individuazione di una libreria grafica	1 h		Riccardo Iervolino
Individuazione di una libreria utile per disegnare grafici	2 h		Riccardo Iervolino
Definizione dell'architettura del programma di backend vera e propria	3 h	20	Raffaele Sorrentino
Creazione class diagrams programma di backend	**1,5 g**	**19;1**	
Creazione class diagram per la gestione degli eventi	5 h		Raffaele Sorrentino
Creazione class diagram per	5 h		Riccardo Iervolino

l'autenticazione			
Creazione class diagram per il cambio della funzione attiva	5 h		Riccardo Iervolino
Creazione class diagram per la consultazione delle statistiche di tutti gli eventi	5 h		Maurizio Moraca
Creazione class diagram per la consultazione delle statistiche di un singolo evento	5 h		Maurizio Moraca
Revisione class diagrams del programma di backend	2 h	24;27;28;25;26	Maurizio Moraca;Riccardo Iervolino;Raffaele Sorrentino
Creazione sequence diagrams programma di backend	**2,5 g**	**1;19**	
Creazione sequence diagram per l'inserimento di un nuovo evento	8 h		Raffaele Sorrentino
Creazione sequence diagram per la modifica di un evento esistente	8 h		Raffaele Sorrentino
Creazione sequence diagram per l'inserimento dei dettagli di una struttura	8 h		Maurizio Moraca
Creazione sequence diagram per la cancellazione di un evento	3 h		Riccardo Iervolino
Creazione del sequence diagram per la visualizzazione delle statistiche di un singolo evento	5 h		Maurizio Moraca
Creazione sequence diagram per la visualizzazione delle statistiche di tutti gli eventi	5 h		Maurizio Moraca
Creazione sequence diagram per il logout	3 h		Riccardo Iervolino
Creazione sequence diagram per l'autenticazione	3 h		Riccardo Iervolino
Revisione sequence diagrams del programma di backend	2 h	31;32;33;34;35;36;37;38	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Varie	**0,63 g**		
Scrittura delle CRC cards	5 h		Raffaele Sorrentino
Traduzione dei templates di Cockburn dall'italiano all'inglese	2 h		Maurizio Moraca
Traduzione di tutti i diagrammi di analisi dall'italiano dall'inglese	2 h		Riccardo Iervolino
Traduzione dei mockups dall'italiano all'inglese	2 h		Maurizio Moraca
Revisione generale della fase di design	**0,38 g**	**1;4;8;13;19;23;30**	

Revisione finale	3 h		Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino

Implementation

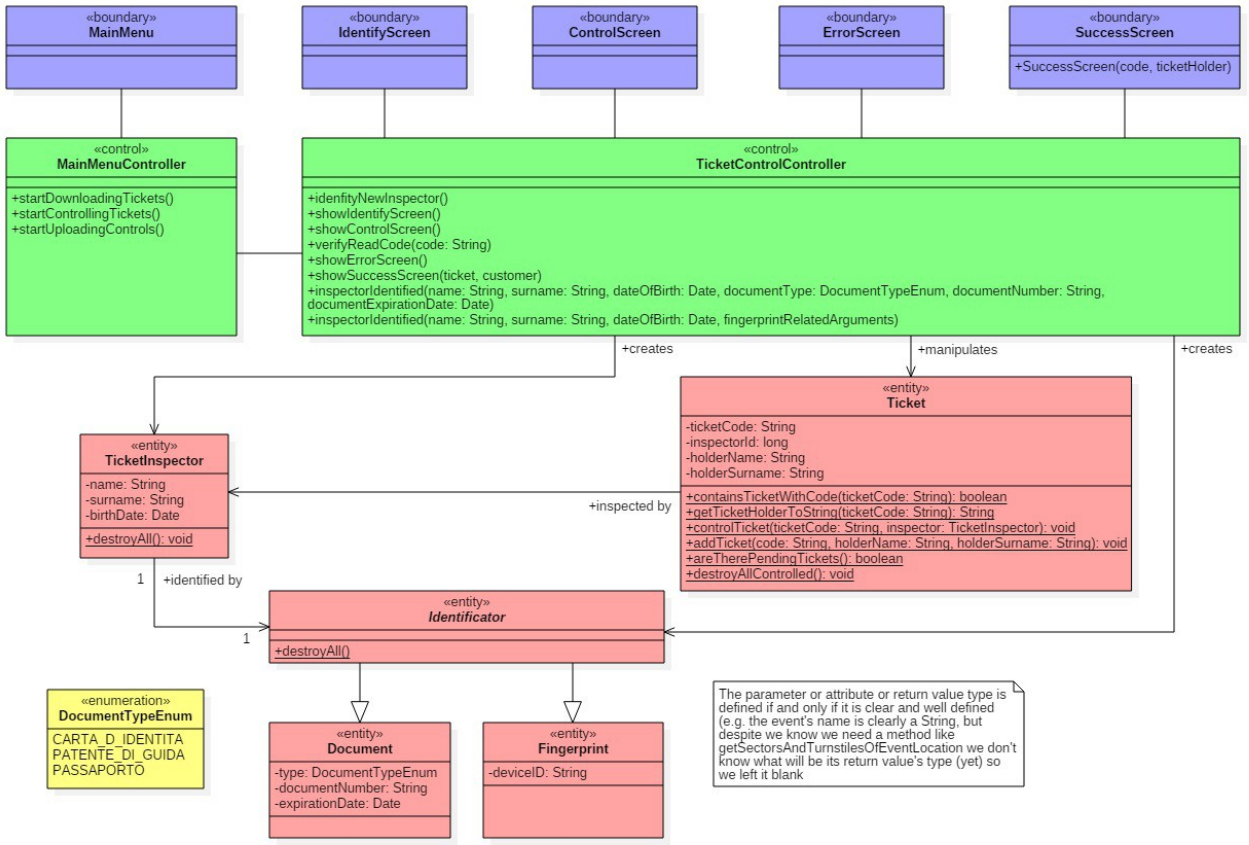
Nome attività	Durata	Predecessori	Nomi risorse
Realizzazione database backend	**2 g**		
Creazione delle tabelle e definizione vincoli del database di backend	6 h		Maurizio Moraca
Revisione tabelle backend	2 h	2	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Realizzazione database app	**1,75 g**		
Creazione delle tabelle e definizione vincoli del database dell'app	6 h		Maurizio Moraca
Revisione tabelle app	2 h	5	Maurizio Moraca;Raffaele Sorrentino;Riccardo Iervolino
Implementazione del backend	**2,5 g**		
Sviluppo delle classi e definizione dei metodi del backend	10 h		Raffaele Sorrentino
Implementazione dei metodi del backend	10 h		Riccardo Iervolino
Revisione del codice del backend	4 h	8;9	Maurizio Moraca;Riccardo Iervolino;Raffaele Sorrentino
Implementazione dell'app	**1,75 g**		
Sviluppo delle classi e definizione dei metodi dell'app	10 h		Riccardo Iervolino
Implementazione dei metodi dell'app	10 h		Maurizio Moraca
Revisione del codice dell'app	4 h	12;13	Riccardo Iervolino;Maurizio Moraca;Raffaele Sorrentino
Implementazione interfaccia grafica del backend	**6 g**	**7**	
Sviluppo interfaccia grafica del backend	10 h		Riccardo Iervolino
Revisione interfaccia grafica del backend	2 h	16	Riccardo Iervolino;Maurizio Moraca;Raffaele Sorrentino
Implementazioine interfaccia grafice dell'app	**2,13 g**	**11**	
Sviluppo interfaccia grafica dell'app	10 h		Raffaele Sorrentino
Revisione interfaccia grafice dell'app	2 h	19	Riccardo Iervolino;Maurizio Moraca;Raffaele Sorrentino
Implementazione testing di due metodi dell'app	**1,75 g**	**11**	
Individuazione della strategia di testing	5 h		Raffaele Sorrentino
Testing dei metodi dell'app	2 h		Maurizio Moraca
Revisione Testing	4 h		Raffaele Sorrentino;Maurizio Moraca;Riccardo Iervolino
Realizzazione documentazione	**1,38 g**	**1;4;7;11;15;18;21**	
Stesura documentazione	9 h		Riccardo Iervolino
Revisione documentazione	2 h		Raffaele Sorrentino;Maurizio Moraca;Riccardo Iervolino

BCE Class diagrams

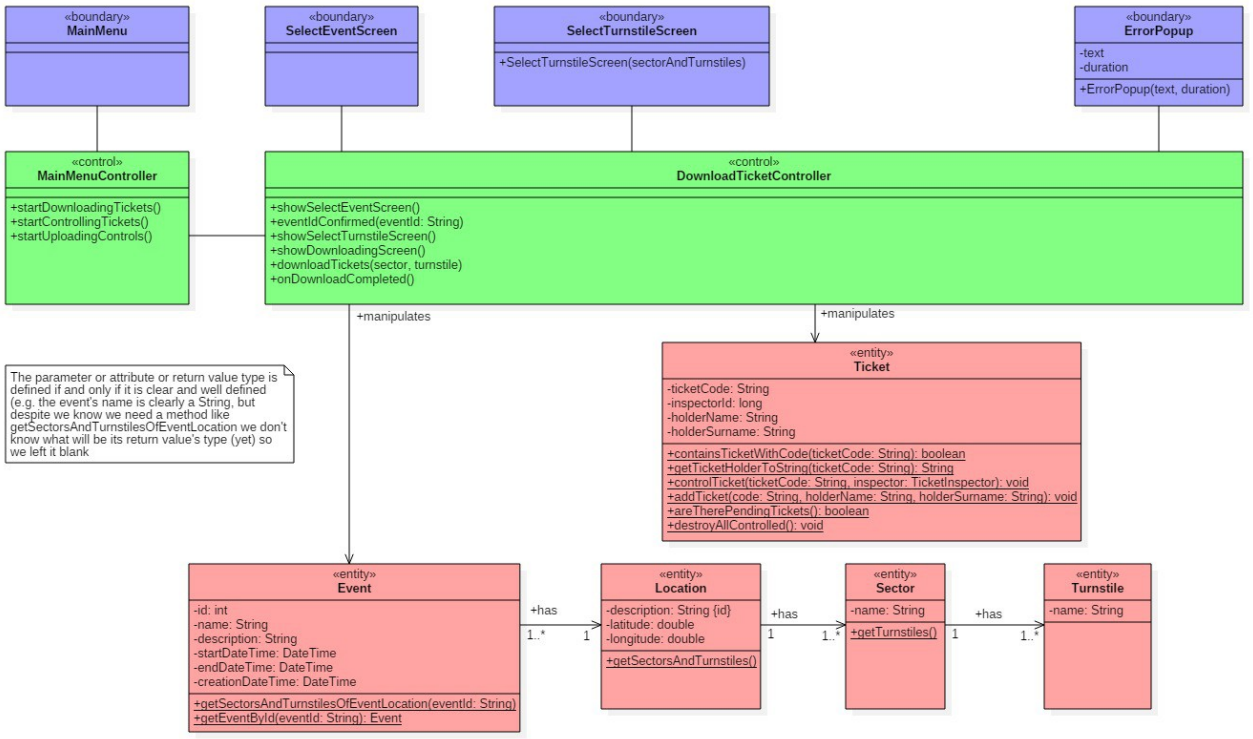
PLEASE NOTE: Due to the large amount of text in these diagrams, should anything not be clearly readable please check the high quality attachments

Android application

Inspector_ControlsATicket



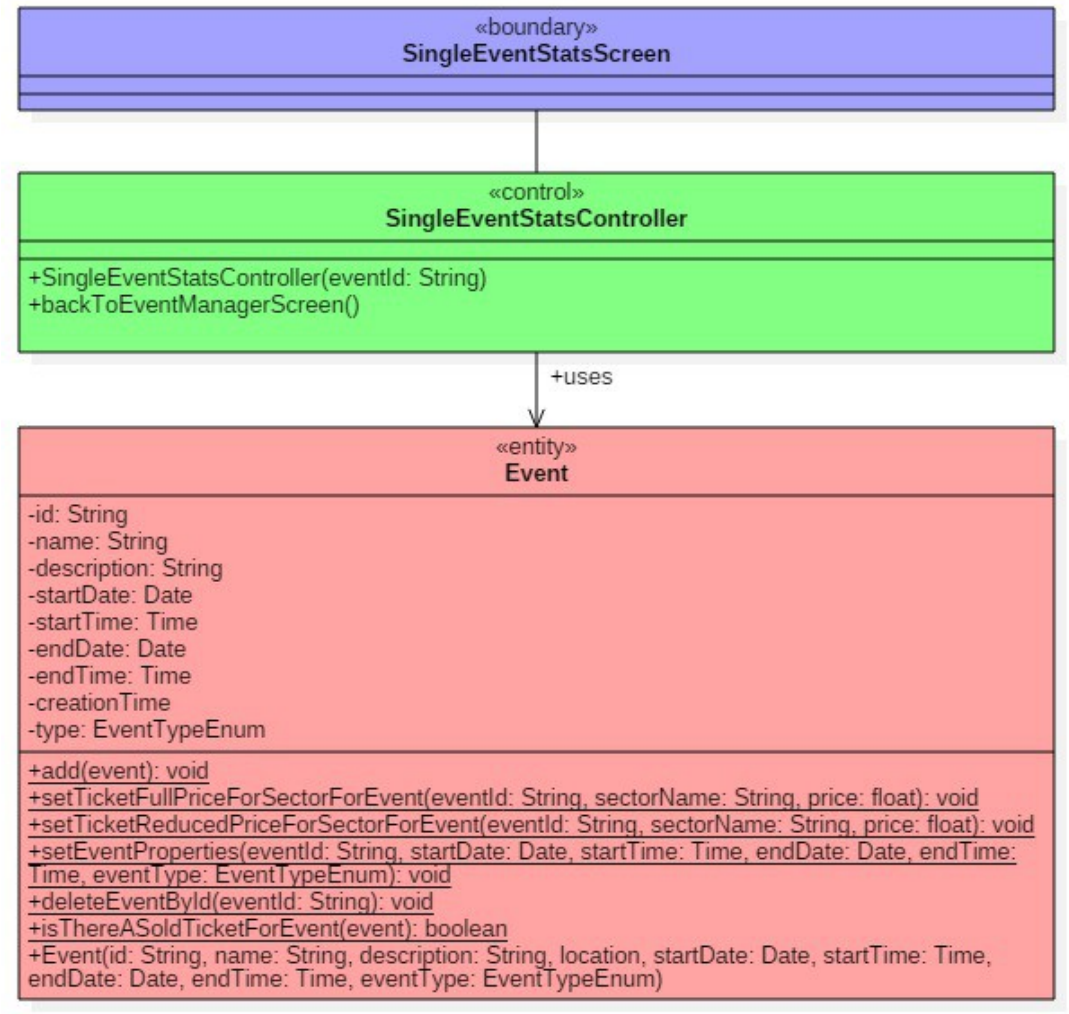
Organizer_DownloadsTheTicketsForSpecificTurnstileAndEvent



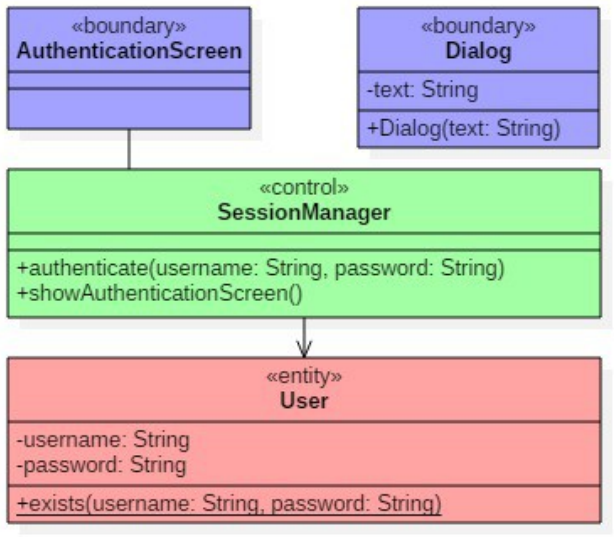
Organizer_UploadsTheControlledTickets

Event manager's platform

AuthenticatedUser_WatchStatisticsSingleEvent

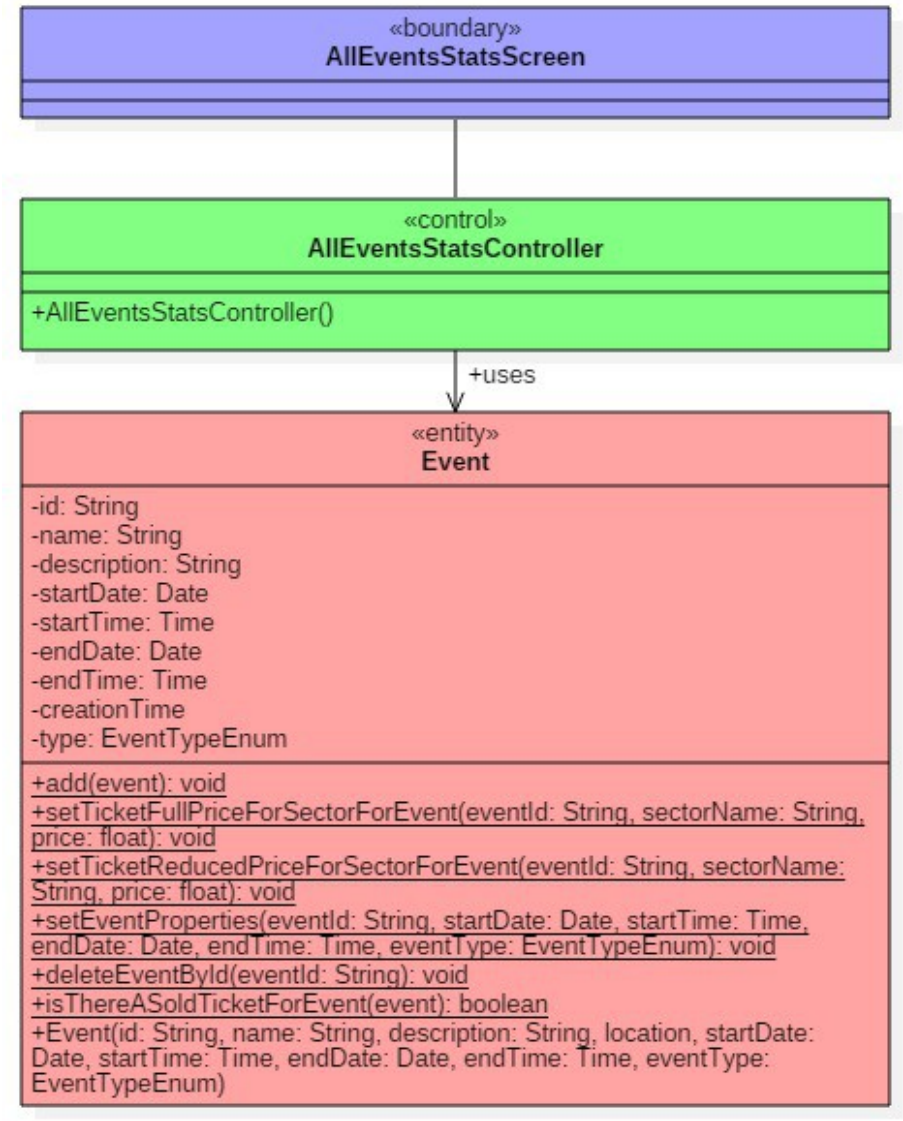


UnauthenticatedUser_Authenticates

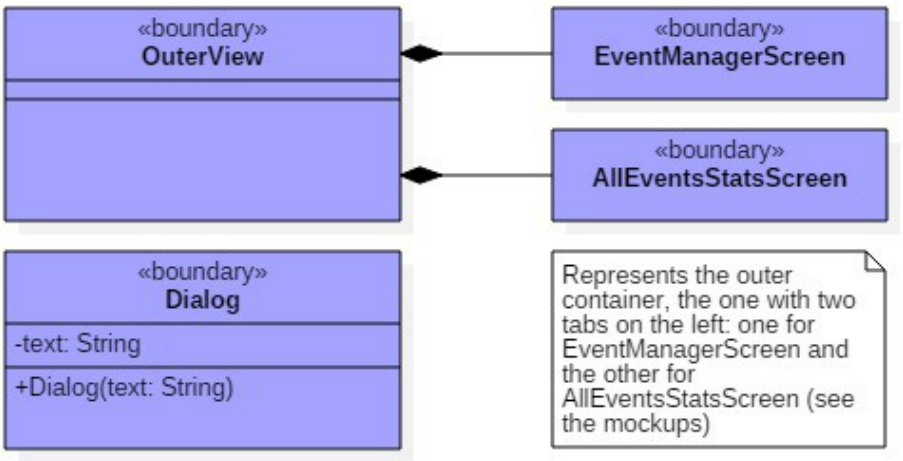


AuthenticatedUser_ManagesTheEvent

AuthenticatedUser_WatchStatisticsAllEvents



ActiveFunctionChanger

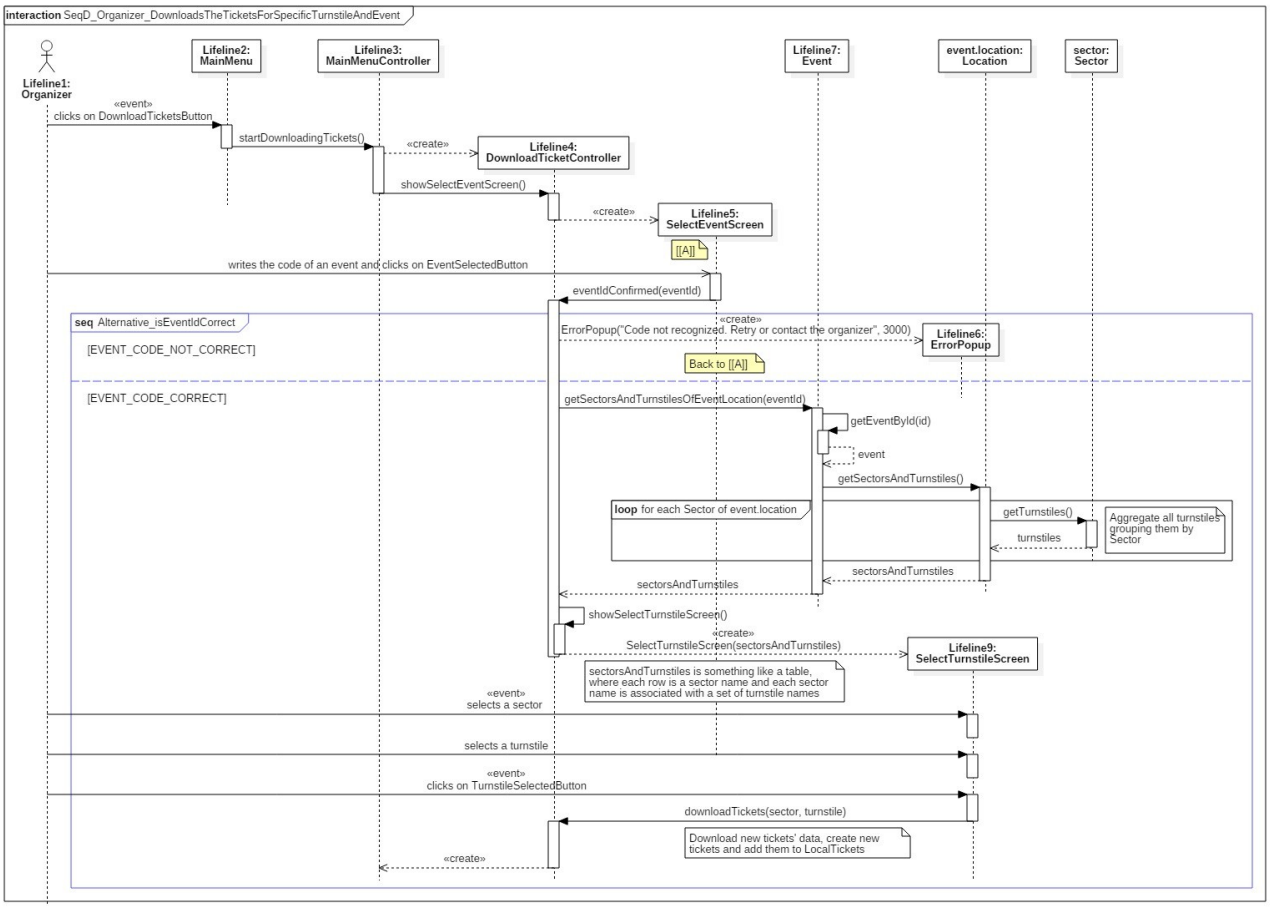


Sequence diagrams

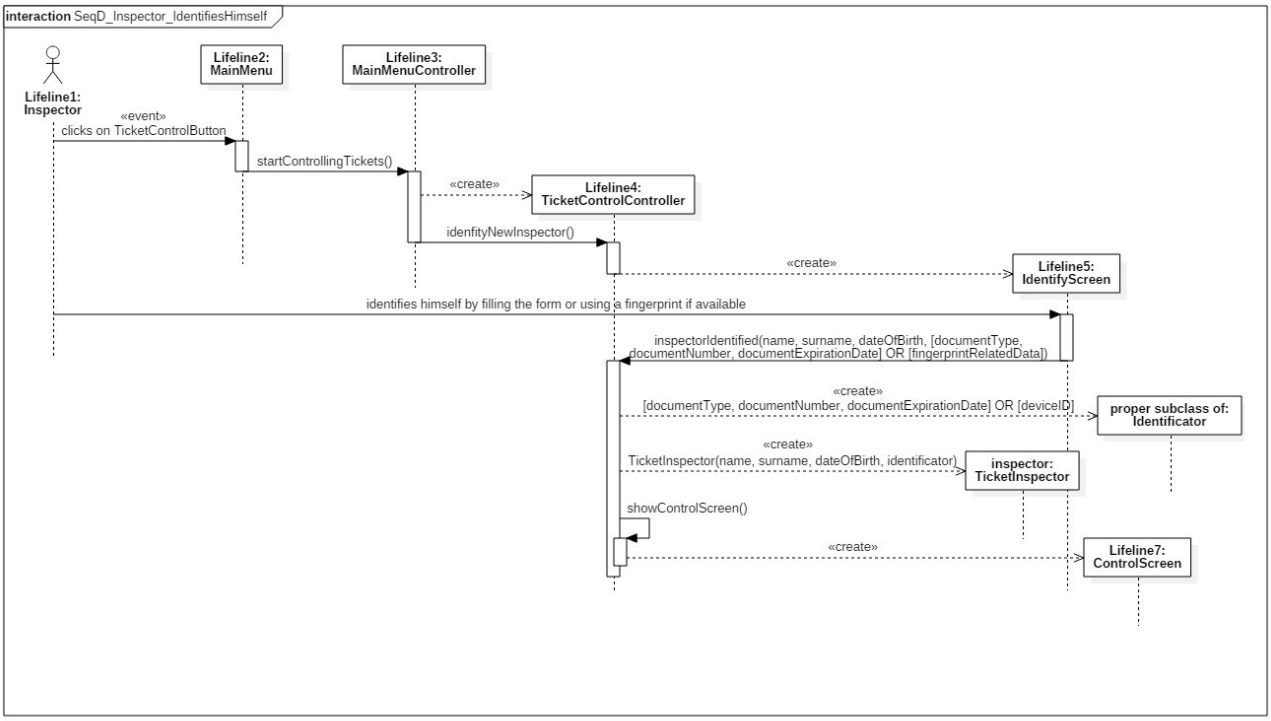
PLEASE NOTE: Due to the large amount of text in these diagrams, should anything not be clearly readable please check the high quality attachments

Android application

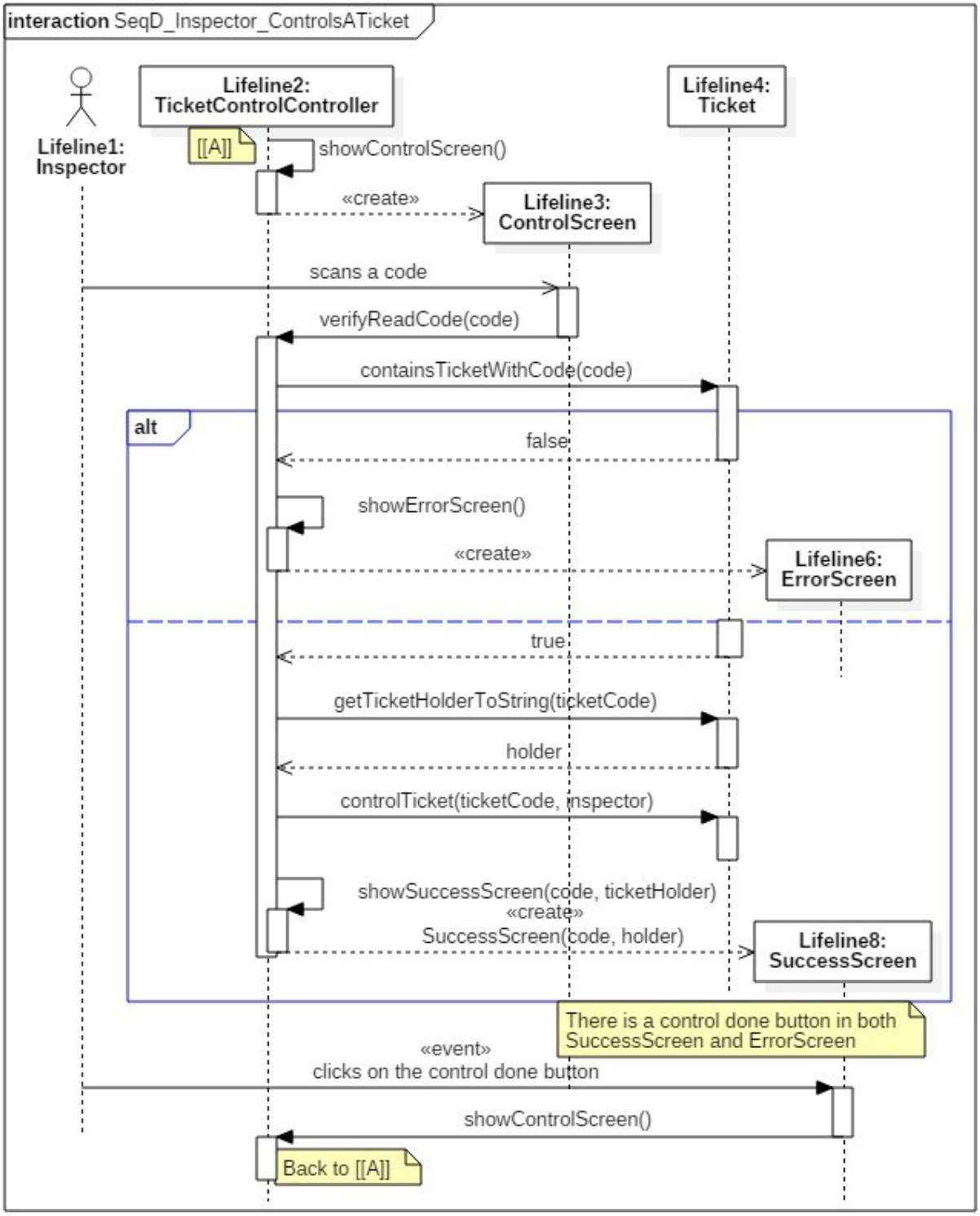
Organizer_DownloadsTheTicketsForSpecificTurnstileAndEvent



Inspector_IdentifiesHimself



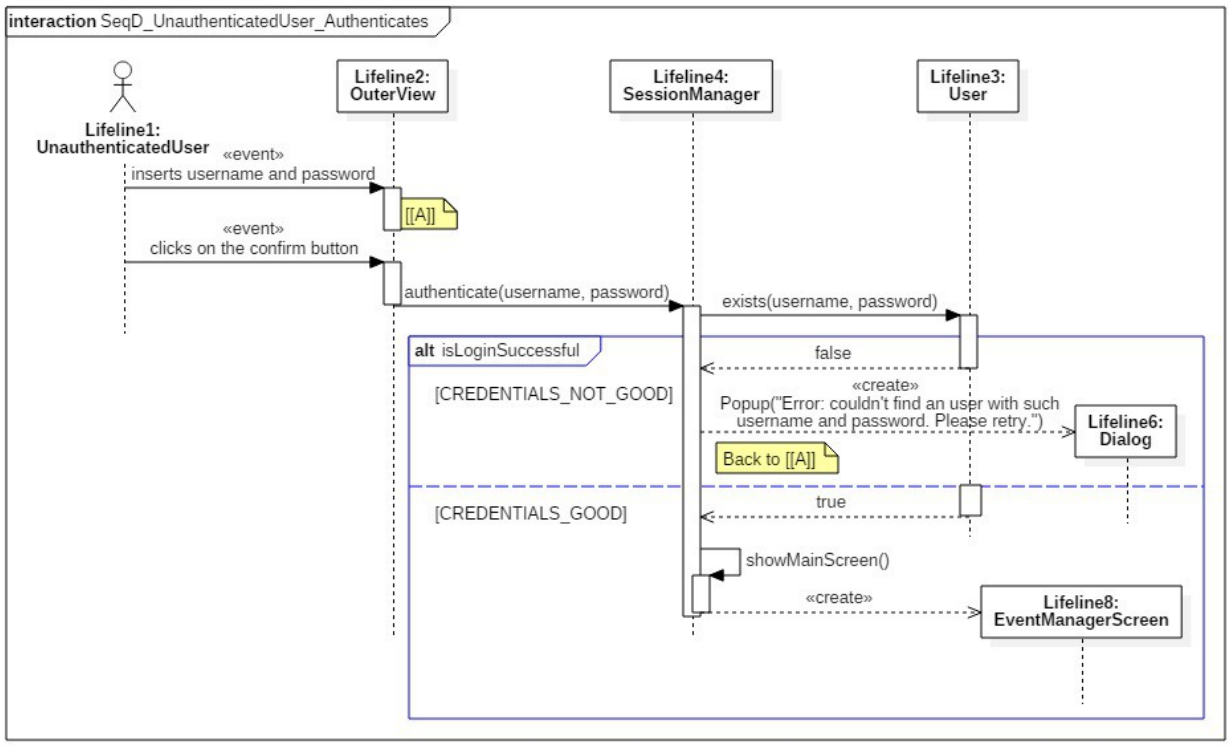
Inspector_ControlsATicket



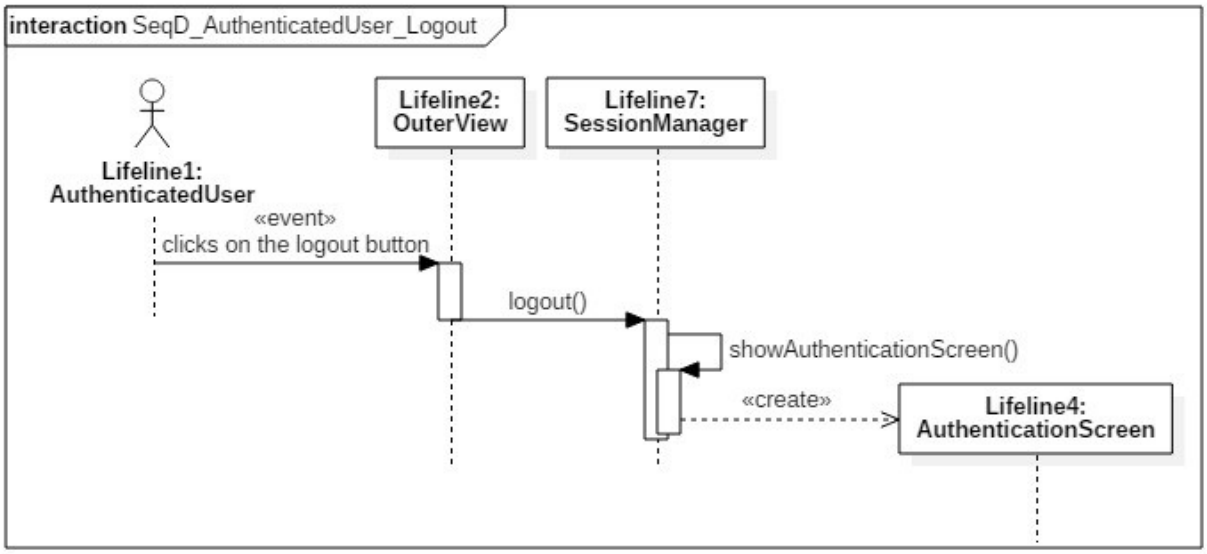
Organizer_UploadTheControlledTickets

Event manager's platform

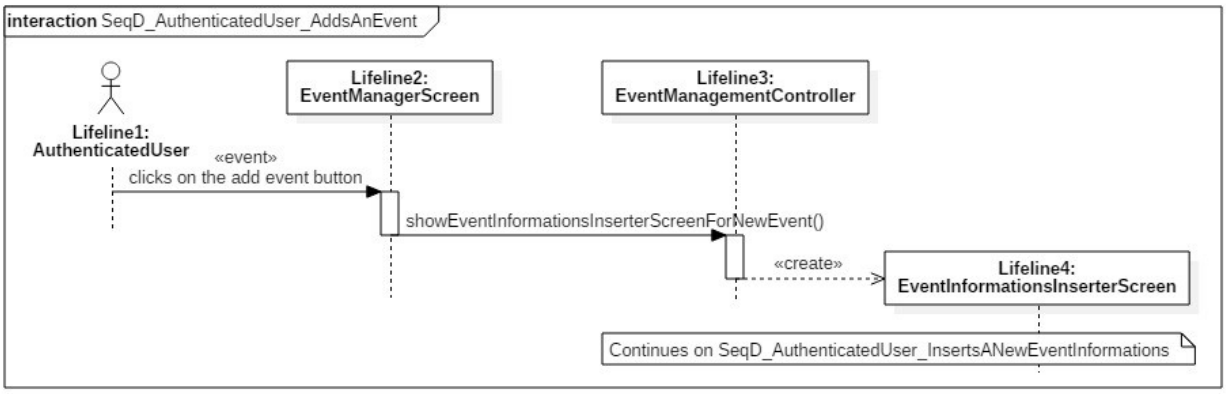
UnauthenticatedUser_Authenticates



AuthenticatedUser_Logout



AuthenticatedUser_AddsAnEvent



AuthenticatedUser_InsertsANewEventInformations

AuthenticatedUser_EditsAnEvent

AuthenticatedUser_InsertsAnExistingEventInformations

AuthenticatedUser_InsertsALocationDetails

AuthenticatedUser_WatchesAllEventsRelatedStats

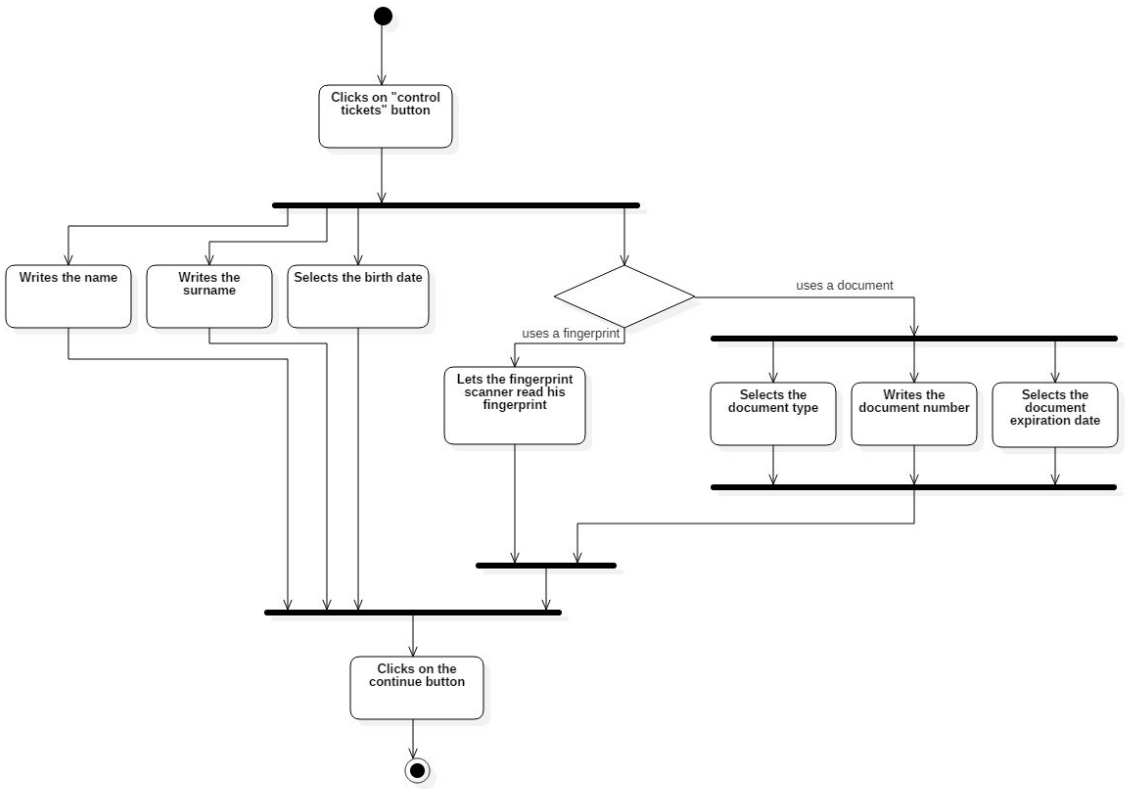
AuthenticatedUser_WatchesSingleEventRelatedStats

Activity diagrams

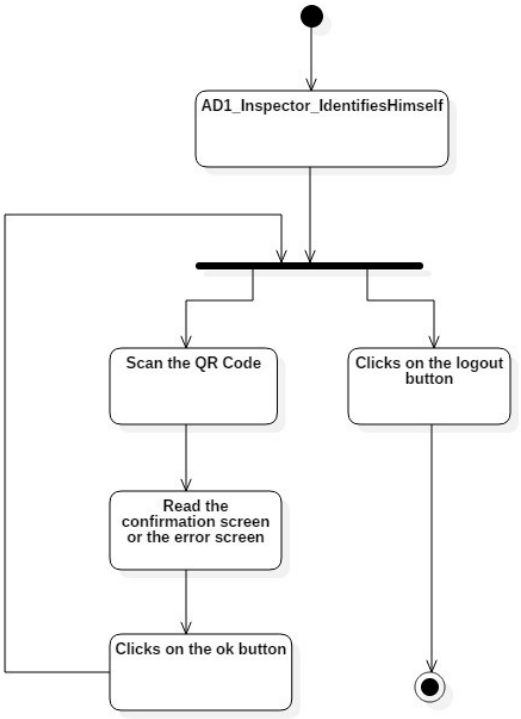
PLEASE NOTE: Due to the large amount of text in these diagrams, should anything not be clearly readable please check the high quality attachments

Android application

Inspector_IdentifiesHimself



Inspector_ControlsATicket



Organizer_DownloadsTheTicketsForSpecificTurnstileAndEvent

Organizer_UploadsTheControlledTickets

Event manager's platform

UnauthenticatedUser_Authenticates

AuthenticatedUser_WatchSingleEventRelatedStats

AuthenticatedUser_WatchAllEventsRelatedStats

Design documentation

Please note: not every single detail has been represented, in order to make the diagrams clearer. We intended to make the diagrams as easy to understand as possible: there should be enough details to understand everything, but not enough to get lost in.

Architecture analysis

EM17 is realized using a multi-tier architecture, which is composed of the following tiers (from the top):

Android application

- [T0] Client tier: the final user, interfaces with [T1]
- [T1] Presentation tier: formed by XML files, pieces of user interface, interfaces with [T0], [T2]
- [T2] UI-Adapter tier: formed by Fragments with an XML file attached
    - gather all of the user action from [T1] and sends it to [T3]
    - can read some informations from [T3] to manipulate [T1]
- [T3] Logic tier:
    - receives the user input, connects to [T4_L] and [T4_O] and manipulates them as needed
    - knows what is the next thing to do and can change the active user interface
- [T4_L] Local data tier: this is something like a cache for [T4_O]: in order to make the application work offline, data are downloaded from [T4_O], used without an internet connection and uploaded again to [T4_O] to commit all modifications; this does NOT interface with [T4_O] but only with [T3]; data are stored in a relational SQLite database

Event manager's platform

- [T0] Client tier: the final user, interfaces with [T1]
- [T1] Presentation tier: formed by FXML files, pieces of user interface, interfaces with [T0], [T2]
- [T2] UI-Adapter tier: formed by Java classes implementing the Initializable interface
    - gather all of the user action from [T1] and sends it to [T3]
    - can read some informations from [T3] to manipulate [T1]
- [T3] Logic tier:
    - receives the user input, connects to [T4_O] and manipulates them as needed
    - knows what is the next thing to do and can change the active user interface

Common

- [T4_O] Online data tier: data are stored in a relational MySQL database hosted on Amazon RDS

Notes

- The software can easily use any graphical user interface because the relationship between it and the logic tier is intermediated by [T2]. Should a new user interface be implemented, it would only be needed to link it to the adapter (e.g. make it so that when a button is pressed a method is called) and of course to manipulate it (but that depends on how is the user interface and not on the software)

- The software can easily use any kind of database for [T4_L] and [T4_O], as long as there are, for that kind of database, an implementation of the Database Manager and all needed implementations of DAOs. This particular feature was achieved using an Abstract DAO Factory design pattern

- There is no entity block, and this is not a forgetfulness: despite we have a block of classes marked as "entity" classes, we didn't feel the need to use complicated entities, hence these classes are not entities in the software-engineering sense of the term, rather they are custom data structures containing some strings, some numbers, and a getter method for each variable (no setter), nothing more and nothing less. In the Android application these classes are used by [T3], [T4_L] and [T4_O]. In the event manager's platform these classes are used by [T2], [T3], [T4_O]. As using strings and numbers in no different than using a custom data structure like a class of these, we picked the second option to improve code readability, but the software can exist without problems without these classes.

Class diagrams

PLEASE NOTE: Due to the large amount of text in these diagrams, should anything not be clearly readable please check the high quality attachments

Android application

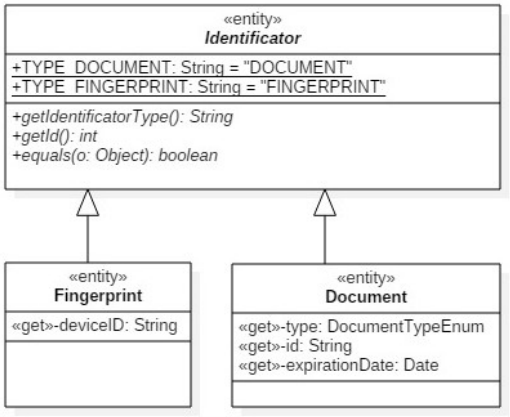
Organizer_DownloadsTheTIcketsForSpecificTurnstileAndEvent

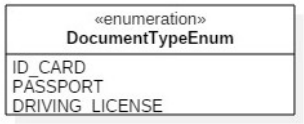
Organizer_UploadsTheControlledTickets

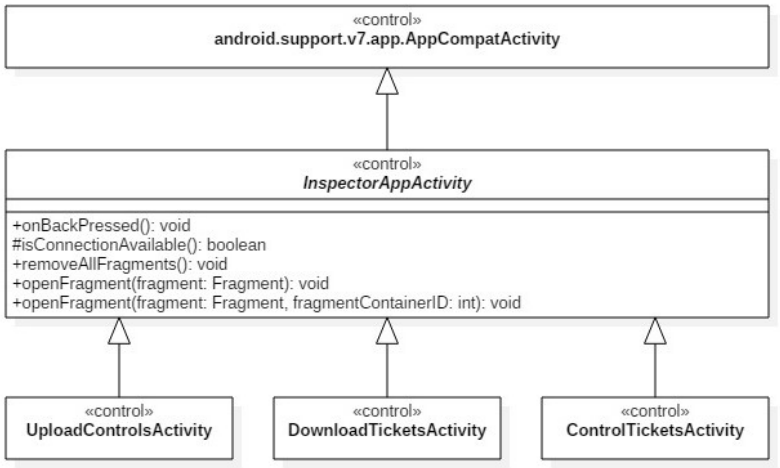
Inspector_ControlsATicket

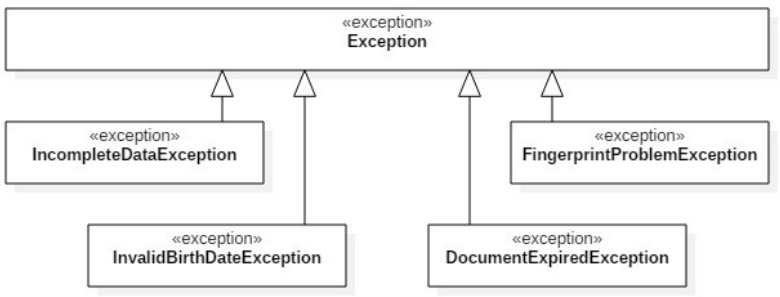
DAO

Others


«entity»
Identificator
+TYPE_DOCUMENT: String = "DOCUMENT"
+TYPE_FINGERPRINT: String = "FINGERPRINT"
+getIdentificatorType(): String
+getId(): int
+equals(o: Object): boolean
«entity»
Fingerprint
«get»-deviceID: String
«entity»
Document
«get»-type: DocumentTypeEnum
«get»-id: String
«get»-expirationDate: Date


«enumeration»
DocumentTypeEnum
ID_CARD
PASSPORT
DRIVING_LICENSE


«control»
android.support.v7.app.AppCompatActivity
«control»
InspectorAppActivity
+onBackPressed(): void
#isConnectionAvailable(): boolean
+removeAllFragments(): void
+openFragment(fragment: Fragment): void
+openFragment(fragment: Fragment, fragmentContainerID: int): void
«control»
UploadControlsActivity
«control»
DownloadTicketsActivity
«control»
ControlTicketsActivity


«exception»
Exception
«exception»
IncompleteDataException
«exception»
FingerprintProblemException
«exception»
InvalidBirthDateException
«exception»
DocumentExpiredException

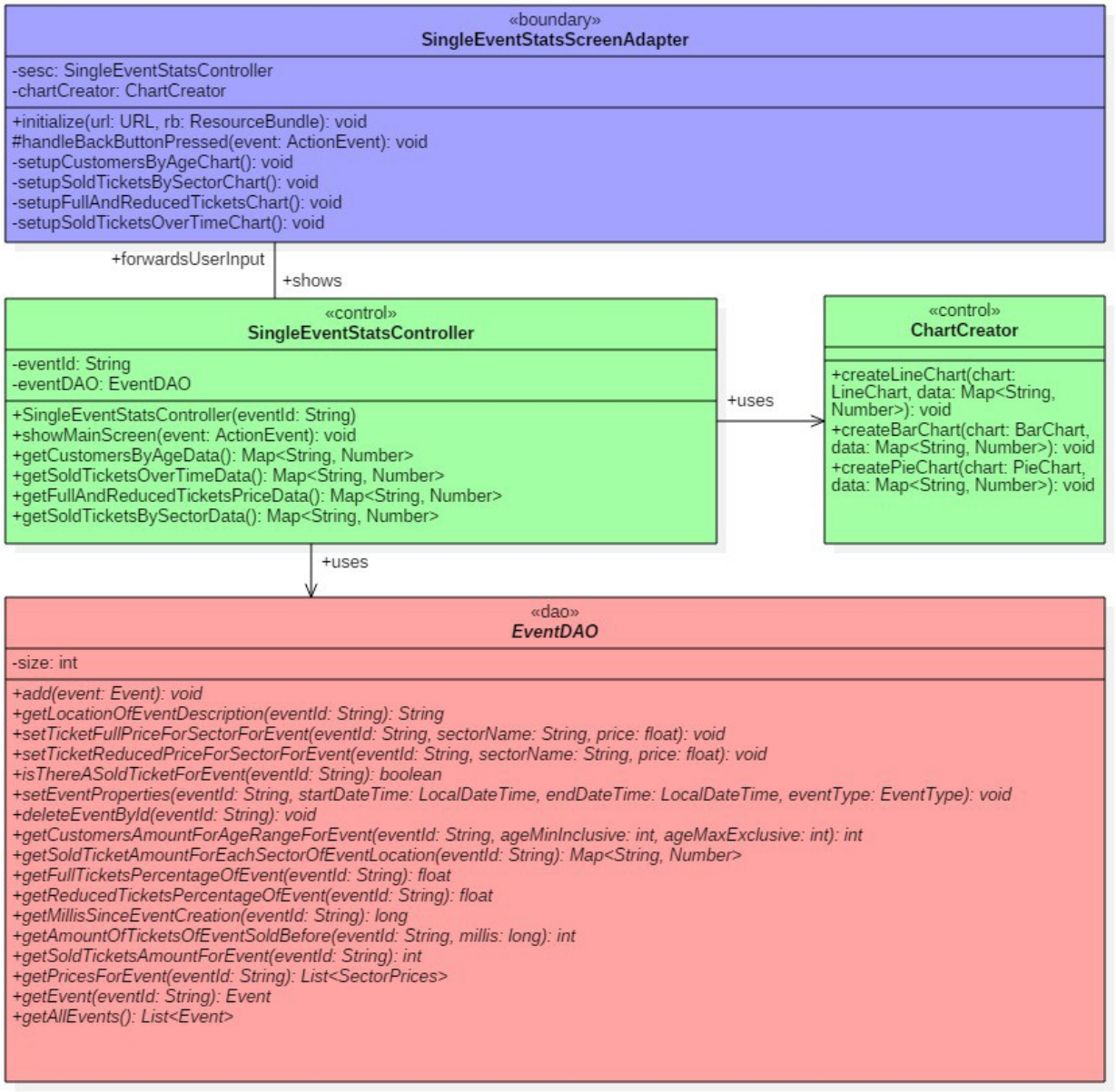
Event manager’s platform

UnauthenticatedUser_Authenticates

AuthenticatedUser_ManagesTheEvent

AuthenticatedUser_WatchStatisticsAllEvents

AuthenticatedUser_WatchStatisticsSingleEvent



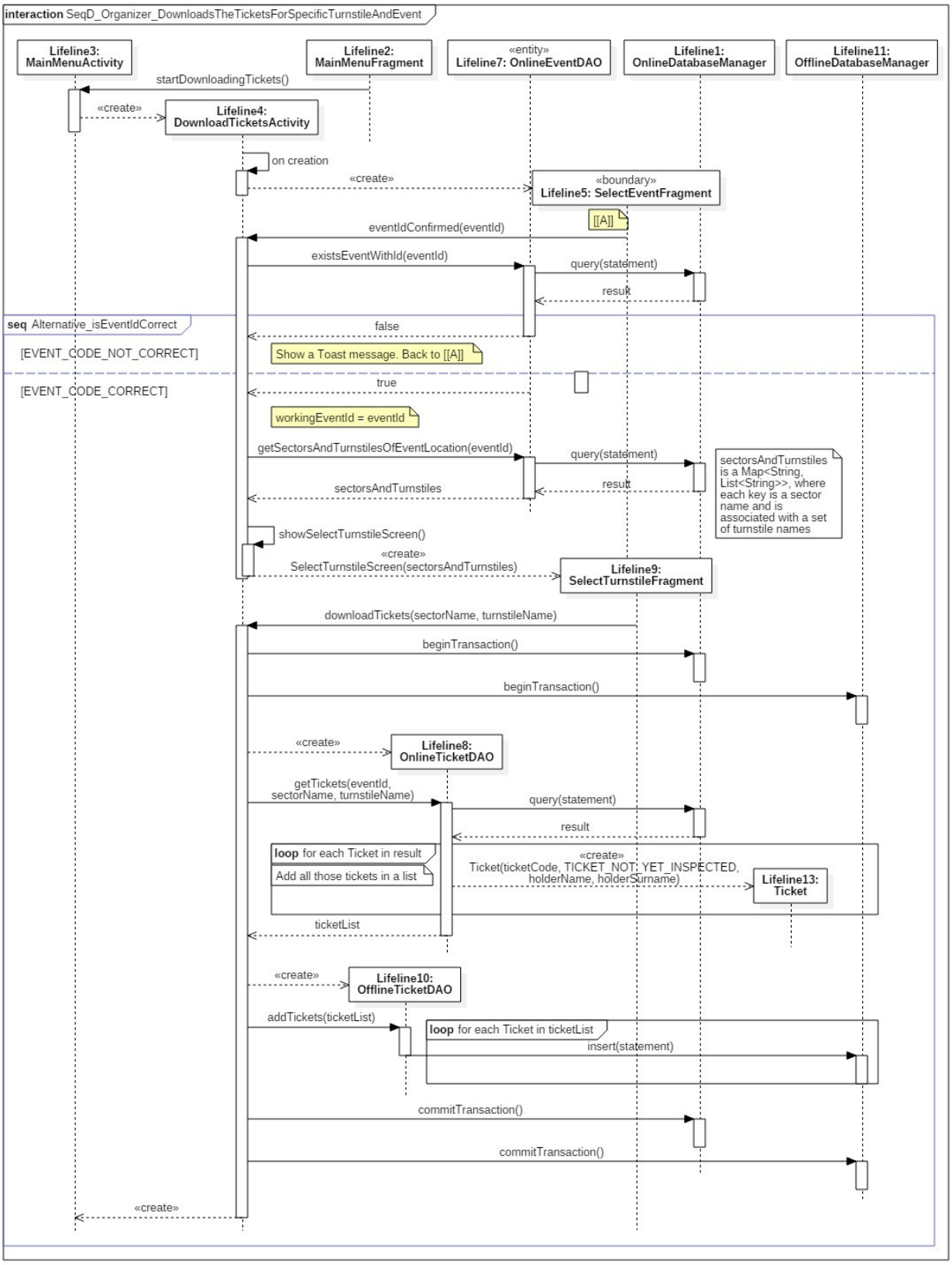
DAO

Others

Sequence diagrams

Android application

Organizer_DownloadsTheTicketsForSpecificTurnstileAndEvent



Inspector_IdentifiesHimself

Inspector_ControlsATicket

Organizer_UploadsTheControlledTickets

Event manager’s platform

UnauthenticatedUser_Authenticates

AuthenticatedUser_Logout

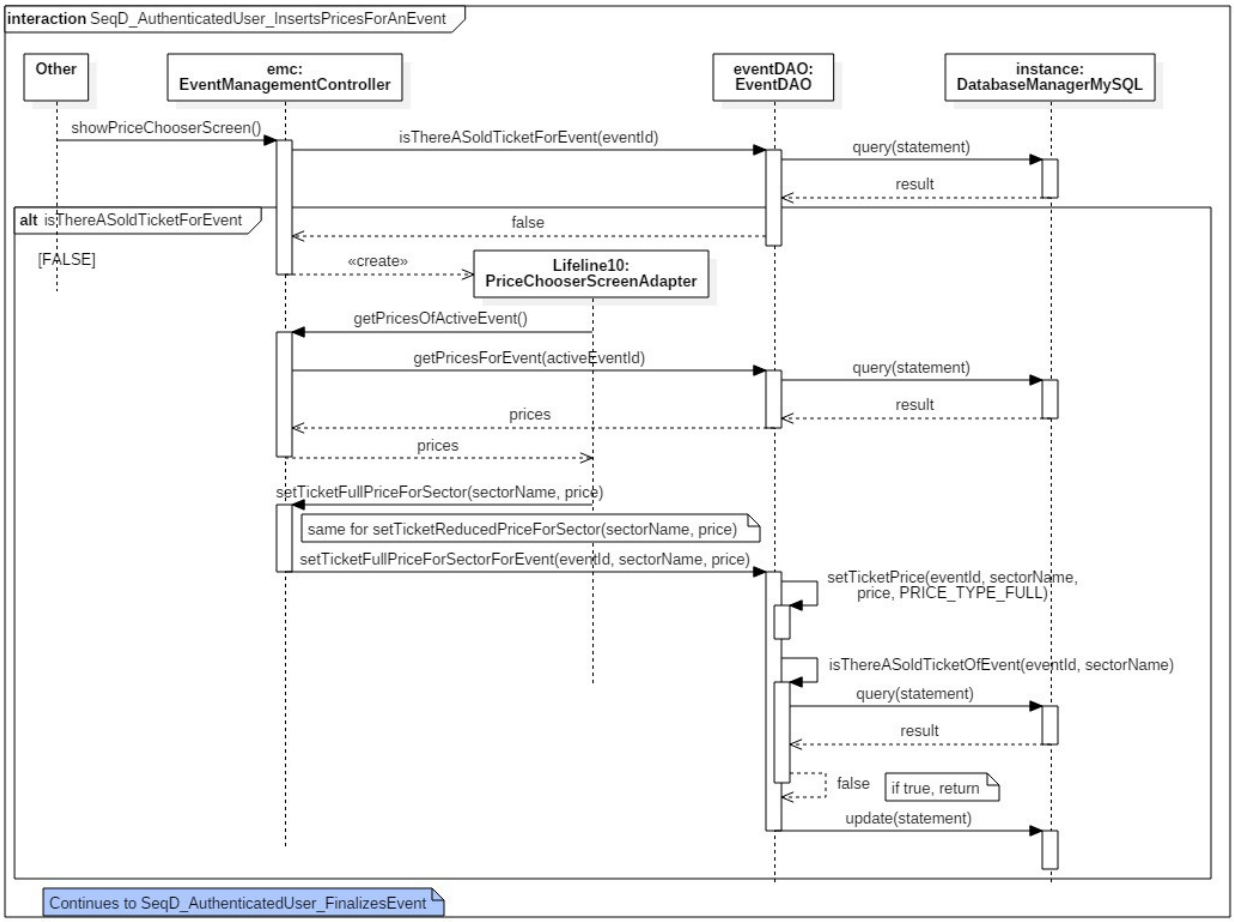
AuthenticatedUser_InsertsANewEventInformations

AuthenticatedUser_InsertsANewEventInformations_NewLocation

AuthenticatedUser_InsertsALocationDetails

AuthenticatedUser_InsertsAnExistingEventInformations

AuthenticatedUser_InsertsPricesForAnEvent



AuthenticatedUser_FinalizesEvent

AuthenticatedUser_DeletesAnEvent

AuthenticatedUser_WatchesSingleEventRelatedStats

AuthenticatedUser_WatchesAllEventsRelatedStats

State diagrams

EventManagementController

CRC Cards

Android application

ClassName:	AnagraphicDataFragment
Responsibilities:	Collaborations:
Shows a form for name surname and date of birth	ControlTicketsActivity

ClassName:	InsertDocumentFragment
Responsibilities:	Collaborations:
Shows a form for the ID InvalidBirthDateException	DocumentTypeEnum DocumentExpiredException IncompleteDataException

ClassName:	ScanFingerprintFragment
Responsibilities:	Collaborations:
Show a fingerprint icon shows a button that allows the user to insert a document as alternative	FingerprintProblemException IncompleteDataException InvalidBirthDateException

ClassName:	ControlFragment
Responsibilities:	Collaborations:
Shows the camera view	none

ClassName:	ErrorFragment
Responsibilities:	Collaborations:
Shows an error and shows a button to scan another ticket	none

ClassName:	SuccessFragment
Responsibilities:	Collaborations:
Shows ticket informations and shows a button to scan another ticket	none

ClassName:	SelectEventFragment
Responsibilities:	Collaborations:
Allows user to insert an event ID	IncompleteDataException

ClassName:	SelectTurnstileFragment
Responsibilities:	Collaborations:
Allows user to select sector and turnstile	IncompleteDataException

ClassName:	MainMenuFragment
Responsibilities:	Collaborations:
Shows a button to activate the download tickets function Shows a button to activate the control tickets function Shows a button to activate the upload controls function	

ClassName:	UploadConfirmationFragment
Responsibilities:	Collaborations:
Asks for user's authorization to start the upload	ToastMessage SessionManager

ClassName:	InspectorAppActivity
Responsibilities:	Collaborations:
Base activity of each app's activity Provides basic common functionalities (e.g. opening a new fragment)	none

ClassName:	MainMenuActivity
Responsibilities:	Collaborations:
App launcher	MainMenuFragment

ClassName:	DownloadTicketsActivity
Responsibilities:	Collaborations:
Manages the download tickets function	OnlineEventDAO OfflineTicketDAO OnlineDatabaseManager OnlineTicketDAO IncompleteDataException OfflineDatabaseManager SelectEventFragment SelectTurnstileFragment Ticket

ClassName:	ControlTicketsActivity
Responsibilities:	Collaborations:
Identifies the ticket inspector Starts the camera looking for a QR code Shows error and success messages Controls the tickets	OfflineIdentificatorDAO OfflineInspectorDAO OfflineTicketDAO OfflineDatabaseManager
Document - DocumentTypeEnum - Fingerprint - Identificator - TicketInspector - DocumentExpiredException - FingerprintProblemException - IncompleteDataException - InvalidBirthDateException - ControlFragment - SuccessFragment - ErrorFragment - AnagraphicDataFragment - AnagraphicDataFragmentEventsListener - InsertDocumentEventsListener - InsertDocumentFragment - ScanFingerprintEventsListener - ScanFingerprintFragment	

ClassName:	UploadControlsActivity
Responsibilities:	Collaborations:
Uploads the ticket inspectors, their identificators and the controlled tickets	OfflineIdentificatorDAO OfflineInspectorDAO OfflineTicketDAO - OnlineDatabaseManager - OnlineIdentificatorDAO - OnlineInspectorDAO - OnlineTicketDAO - OfflineDatabaseManager - UploadConfirmationFragment - TicketInspector - Identificator - Ticket

ClassName:	OfflineDatabaseManager
Responsibilities:	Collaborations:
Interfaces with the local database	none

ClassName:	OfflineDatabaseSchema
Responsibilities:	Collaborations:
Contains the schema for the offline database	DocumentTypeEnum

ClassName:	OnlineDatabaseManager
Responsibilities:	Collaborations:
Interfaces with the online database	none

ClassName:	OnlineDatabaseSchema
Responsibilities:	Collaborations:
Contains the schema for the online database	none

ClassName:	OfflineIdentificatorDAO
Responsibilities:	Collaborations:
Execute various operations with identificators stored or to be stored in the offline database	Document DocumentTypeEnum Fingerprint Identificator OfflineDatabaseManager OfflineDatabaseSchema

ClassName:	OfflineInspectorDAO
Responsibilities:	Collaborations:
Execute various operations with inspectors stored or to be stored in the offline database	TicketInspector OfflineDatabaseManager OfflineDatabaseSchema

ClassName:	OfflineTicketDAO
Responsibilities:	Collaborations:

Execute various operations with tickets stored or to be stored in the offline database	Ticket OfflineDatabaseManager OfflineDatabaseSchema

ClassName:	OnlineEventDAO
Responsibilities:	Collaborations:
Execute various operations with events stored or to be stored in the offline database	OnlineDatabaseManager OnlineDatabaseSchema

ClassName:	OnlineIdentificatorDAO
Responsibilities:	Collaborations:
Execute various operations with identificators stored or to be stored in the offline database	OnlineDatabaseManager OnlineDatabaseSchema Document Fingerprint Identificator

ClassName:	OnlineInspectorDAO
Responsibilities:	Collaborations:
Execute various operations with inspectors stored or to be stored in the offline database	OnlineDatabaseManager OnlineDatabaseSchema TicketInspector

ClassName:	OnlineTicketDAO
Responsibilities:	Collaborations:
Execute various operations with tickets stored or to be stored in the offline database	OnlineDatabaseManager OnlineDatabaseSchema Ticket

ClassName:	Document
Responsibilities:	Collaborations:
Holds an inspector's document informations	DocumentType

ClassName:	DocumentTypeEnum
Responsibilities:	Collaborations:
Contains all the availables document types	none

ClassName:	EventTypeEnum
Responsibilities:	Collaborations:
Contains all the availables event types	none

ClassName:	Fingerprint
Responsibilities:	Collaborations:
Holds an inspector's fingerprint informations	none

ClassName:	Identificator
Responsibilities:	Collaborations:
High-level representation of an identificator	none

ClassName:	Ticket
Responsibilities:	Collaborations:
Holds a ticket informations	none

Event manager's platform

ClassName:	SceneManager
Responsibilities:	Collaborations:
Changes the currently displayed scene	none

ClassName:	AuthenticationManager
Responsibilities:	Collaborations:
Allows user to login	ToastMessage SessionManager

ClassName:	OuterviewAdapter
Responsibilities:	Collaborations:
Allows the user to logout Contains EventManagerScreen and AllEventsStatsScreen	SessionManager

ClassName:	EventManagerScreenAdapter
Responsibilities:	Collaborations:
Shows all existing events, allows editing them or deleting them or consulting statistic informations for them and creating a new one	EventManagementControler Event EventTypeEnum

ClassName:	AllEventsStatsScreenAdapter
Responsibilities:	Collaborations:
Shows statistics of all events	AllEventsStatsController ChartCreator

ClassName:	EventIInformationInserterScreenAdapter
Responsibilities:	Collaborations:
Shows a form that allows the user to insert all the basic informatio	EventManagementController

ns for an event	
	SessionManager
	ToastMessage
	EventTypeEnum
	Event

ClassName:	EventLocationDetailsInserterScreenAdapter
Responsibilities:	Collaborations:
Contains a form that allows the user to insert a location's details, and in particular details about its sectors, their capacity and their turnstiles	EventManagementController SessionManager Turnstile Sector

ClassName:	PriceChooserScreenAdapter
Responsibilities:	Collaborations:
Shows a list of all sectors of the selected event's location and allows the user to set a price for each of them	EventManagementController SessionManager SectorPrices

ClassName:	ToastMessage
Responsibilities:	Collaborations:
Shows Android-like toast messages in various styles	none

ClassName:	AllEventsStatsController
Responsibilities:	Collaborations:
Calculates statistic informations relative to all existing events	EventTypeEnum SoldTicketDAO

ClassName:	ChartCreator
Responsibilities:	Collaborations:
Creates and populates various types of charts	none

ClassName:	EventManagementController
Responsibilities:	Collaborations:
Retrieves the active event's list	CustomerDAO
Creates, modifies and deletes events	EventDAO Event
Creates locations	SceneManager

Sets the prices for an event	EventTypeEnum Location LocationDAO Turnstile EmailSender Sector SectorPrices

ClassName:	GoogleMapsAPI
Responsibilities:	Collaborations:
Checks if a string corresponds to an existing location and creates a location	Location

ClassName:	SessionManager
Responsibilities:	Collaborations:
Connects the user to the application Checks if the given user credentials exists in the database	SceneManager UserDAO

ClassName:	SingleEventStatsController
Responsibilities:	Collaborations:
Calculates statistic informations relative to a specific event	EventDAO

ClassName:	MathUtils
Responsibilities:	Collaborations:
Does a little formatting with float numbers	none

ClassName:	EmailSender
Responsibilities:	Collaborations:
Connects to the SMTP server and sends emails	none

ClassName:	ConnectionManager
Responsibilities:	Collaborations:
Ensures that the application runs only if a connection is available	none

ClassName:	EventTypeEnum
Responsibilities:	Collaborations:
Holds informations about all available types for an event	none

ClassName:	Location
Responsibilities:	Collaborations:
Holds informations about a location	none

ClassName:	Sector
Responsibilities:	Collaborations:
Holds informations about a sector of a location	none

ClassName:	SectorPrices
Responsibilities:	Collaborations:
Holds informations about all the prices (reduced and full) for a specific event and for a specific sector	none

ClassName:	Turnstile
Responsibilities:	Collaborations:
Holds informations about a turnstile of a sector of a location	none

ClassName:	User
Responsibilities:	Collaborations:
Holds informations about an user	none

ClassName:	CustomerDAO
Responsibilities:	Collaborations:
Execute various operations with customers stored or to be stored in the offline database	DatabaseManager

ClassName:	DBSchema
Responsibilities:	Collaborations:
Contains the schema for the online database	none

ClassName:	EventDAO
Responsibilities:	Collaborations:
Execute various operations with events stored or to be stored in the offline database	OnlineDatabaseManager OnlineDatabaseSchema

ClassName:	LocationDAO
Responsibilities:	Collaborations:
Execute various operations with locations stored or to be stored in the database	DatabaseManager Location Turnstile Sector

ClassName:	SoldTicketsDAO
Responsibilities:	Collaborations:
Execute various operations with tickets stored or to be stored in the database	DatabaseManager

ClassName:	UserDAO
Responsibilities:	Collaborations:
Execute various operations with users stored or to be stored in the database	DatabaseManager

Implementation documentation

Android application source code

See the attachment file “Android application source code.pdf”

Event manager’s platform source code

See the attachment files “Event manager platform source code.pdf”

Testing documentation

Test plan

This section of the document describes the testing strategy and approach to testing the team used to verify that the system meets the established requirements of the business prior to distribution. All functional and non-functional requirements are specified in this same document.

Testing strategy

Standard

- Valid inputs (tested manually by using the system as the end-user)
- Try as much as possible not allowed user actions
- Try to interrupt or disturb events, for example close the program while an update is in progress or disable the internet connection while a download is in progress

Extended

- All the above
- Black box testing: Strict Equivalence Class Testing (each parameter's domain was divided into many equivalence classes and there is one method for each combination of equivalence classes, i.e. if there are three parameters having respectively 5, 3 and 2 equivalence classes, there are 5x3x2 = 30 test cases)
- White box testing: All edges coverage

Acceptance criteria

In general, a test case is "PASSED" when the real outcome corresponds to the expected outcome, otherwise the test case is marked as "FAILED". Expected outcomes are defined separately in each test case.

- 100% test coverage is required (i.e. all implemented changes need to be tested at least once)
- no critical defects are tolerated
- 100% of failed test cases will be retested

Test environment

Android application

- Android emulator on Genymotion:
  - Samsung Galaxy S8 – API 24 – 1440x2960 – 1 dedicated CPU – 4GB DRAM
  - Google Pixel – API 26 – 1080x1920 – 1 dedicated CPU – 4GB DRAM
  - Emulator running through VirtualBox on a PC with the following characteristics: OS Windows 10, CPU Ryzen 5 1600@3.5GHz, GTX 1060 6GB, 16GB DDR4, 3840x1080
- Lenovo Zuk Z1 smartphone, API 22
- Xiaomi Redmi Note 4 smartphone, API 23
- Asus Zenfone 3 smartphone, API 26

Event manager's platform

- PC with the following characteristics: OS Windows 10, CPU Ryzen 5 1600@3.5GHz, GTX 1060 6GB, 16GB DDR4, 3840x1080
- PC with the following characteristics: OS Windows 10, CPU Intel i7-6500U@2.5GHz, RX 570MB, 6GB DDR3, 1366x768

To test function

Android application

1. Downloading tickets for an event
2. Identifying an inspector
3. Controlling a ticket
4. Uploading the controlled tickets

Event manager's platform

5. Logging in
6. Reading the list of all the events
7. Adding a new event in a new location
8. Adding a new event in an existing location
9. Modifying an event
10. Deleting an event
11. Reading statistics about a single event
12. Reading statistics about all events together
13. Logging out

Tests (standard strategy)

A. Any
    a. A required field is not filled up but the user wants to advance to the next step
        a.i. Expected: an error message saying all fields needs to be filled should pop up
        a.ii. Pass/Fail: PASS

B. When running the event manager's platform
    a. The connection is disabled
        a.i. Expected: an error message saying that an internet connection is needed should pop up; when closed, the program should close
        a.ii. Pass/Fail: PASS

1. Downloading tickets for an event
    a. Download tickets for the eventId "abcdef"
        a.iii. Expected: an error message saying that the event does not exists should pop up
        a.iv. Pass/Fail: PASS
    b. Download tickets for the eventId "ciPRsGLq1dJZ"
        b.i. Expected: after selecting a sector and a turnstile, the tickets should be downloaded
        b.ii. Pass/Fail: PASS
    c. The internet connection is disabled while the download is in progress
        c.i. Expected: no tickets are downloaded
        c.ii. Pass/Fail: PASS
2. Identifying an inspector
    a. All anagraphic fields are filled and a fingerprint is scanned
        a.i. Expected: a new inspector and a new identificator are added to the local database
        a.ii. Pass/Fail: PASS
    b. Invalid user input is disabled by the graphical user interface but is tested below (see JUnit test section)
3. Controlling a ticket
    a. A QR code is scanned and its content is the code of a ticket stored in the local database
        a.i. Expected: the ticket is marked as inspected by the currently active inspector; a success screen is shown
        a.ii. Pass/Fail: PASS
    b. A QR code is scanned and its content is NOT the code of a ticket stored in the local database
        b.i. Expected: an error screen is shown

b.ii. Pass/Fail: PASS

4. Uploading the controlled tickets
    a. The user uploads the controlled tickets, the inspectors and their identificators
        a.i. Expected: all data are uploaded to the database
        a.ii. Pass/Fail: PASS
    b. The internet connection is disabled while the upload is in progress
        b.i. Expected: no data are uploaded to the database and no changes occur in the online database or in the local database

b.ii. Pass/Fail: PASS

5. Logging in
    a. Logging in using valid credentials: username “admin”, password “admin”, without quotes
        a.i. Expected: the user is logged in to the database
        a.ii. Pass/Fail: PASS
    b. Logging in using wrong credentials: username “ciao”, password “hello”, without quotes
        b.i. Expected: an error message is shown
        b.ii. Pass/Fail: PASS
6. Reading the list of all the events
    a. The user opens the main screen
        a.i. Expected: the list of all events should be shown
        a.ii. Pass/Fail: PASS
7. Adding a new event in a new location
    a. The location is not found on Google Maps
        a.i. Expected: an error message should pop up
        a.ii. Pass/Fail: PASS
    b. The location is found on Google Maps
        b.i. Expected: a new location is added to the database; the “Insert location details” screen should be shown
        b.ii. Pass/Fail: PASS
8. Adding a new event in an existing location
    a. The location is not free in the selected time frame
        a.i. Expected: an error message should be shown
        a.ii. Pass/Fail: PASS
    b. The location is free in the selected time frame
        b.i. Expected: the “Insert prices” screen should be shown
9. Modifying an event
    a. There are sold tickets for the event
        a.i. Expected: the price chooser screen is not shown
        a.ii. Pass/Fail: PASS
    b. There are not sold tickets for the event
        b.i. Expected: the price chooser screen is shown
        b.ii. Pass/Fail: PASS
10. Deleting an event
    a. An event deletion is requested
        a.i. Expected: the list of all events should be updated and the event should be deleted from the database
        a.ii. Pass/Fail: PASS
11. Reading statistics about a single event
    a. Statistics about an event are requested
        a.i. Expected: the statistics for the event should be shown
        a.ii. Pass/Fail: PASS
12. Reading statistics about all events together
    a. Statistics about all events are requested
        a.i. Expected: the statistics should be shown
        a.ii. Pass/Fail: PASS

13. Logging out
    a. The logout is requested
        a.i. Expected: the user is logged out and the “Authentication” screen is shown
        a.ii. Pass/Fail: PASS

Tests (extended strategy)

Three methods are tested using the extended strategy. Their code has made it so that when the user input is not acceptable the method throws an exception, and these test cases (made with JUnit) are made to check if the exception is thrown successfully

Checking if an inspector is correctly created

```
Android application →
      ControlTicketsActivity →
            createInspector(name: String, surname: String, dateOfBirth:
            Date, identificatorId: String)
```

See the attachment file “JUnit Test 1.pdf”

Checking if an identificator is correctly created from a document

```
Android application →
      ControlTicketsActivity →
            identifyInspector(documentType: DocumentTypeEnum.DocumentType,
            documentNumber: String, documentExpirationDate: Date)
```

See the attachment file “JUnit Test 2.pdf”

Checking if an identificator is correctly created from a fingerprint

```
Android application →
      ControlTicketsActivity →
            identifyInspector(deviceID: String)
```

See the attachment file “JUnit Test 3.pdf”