Travlendar+

ACCEPTANCE & TEST

DELIVERABLE

Authors:

Edoardo D'Amico
Gabbolini Giovanni
Parroni Federico

Git Repository: https://github.com/keyblade95/DamicoGabboliniParroni

8th December 2017

Indice

Chapter 1

Analized Project

1.1 Name of Authors of the analyzed project

- Michele De Pascalis,

- Gianmarco Dello Preite Castro,

- Amin Mahboubi.

1.2 Link to the repository

https://github.com/Gianmarcodpc/DePascalisDelloPreiteCastroMahboubi

1.3 Main reference document considered

The Main document considered is the ITD.

Chapter 2

Installation Instructions

The software has been installed trough the apk whitin the deliverables folder in the github repository how said from the group in the ITD document. The application has been installed on two different virtual devices, the nexus 5 emulator, with android version 8.0(Oreo) and in the nexsus 5x with the same version of android, that has been done for test how the application can adapt itsel to different display size, 5 for the first one and 5.5 for the second one. There were no inchorency on the installation instruction released on the ITD apart for the presence of the application in the google playstore, in fact the application can't be downloaded from it.

Chapter 3

Test of the implemented functionalities

The tests have been done on the implemented functionalities declared on the ITD document released by the other team, for each of them at least on test has been carried out.

3.1 Insertion of events and free times

The insertion of an event can be done by clicking the button the main page of the application

Figura 3.1: Main page view

The user is redirect to an event creation view with some fields to fill as shown below.
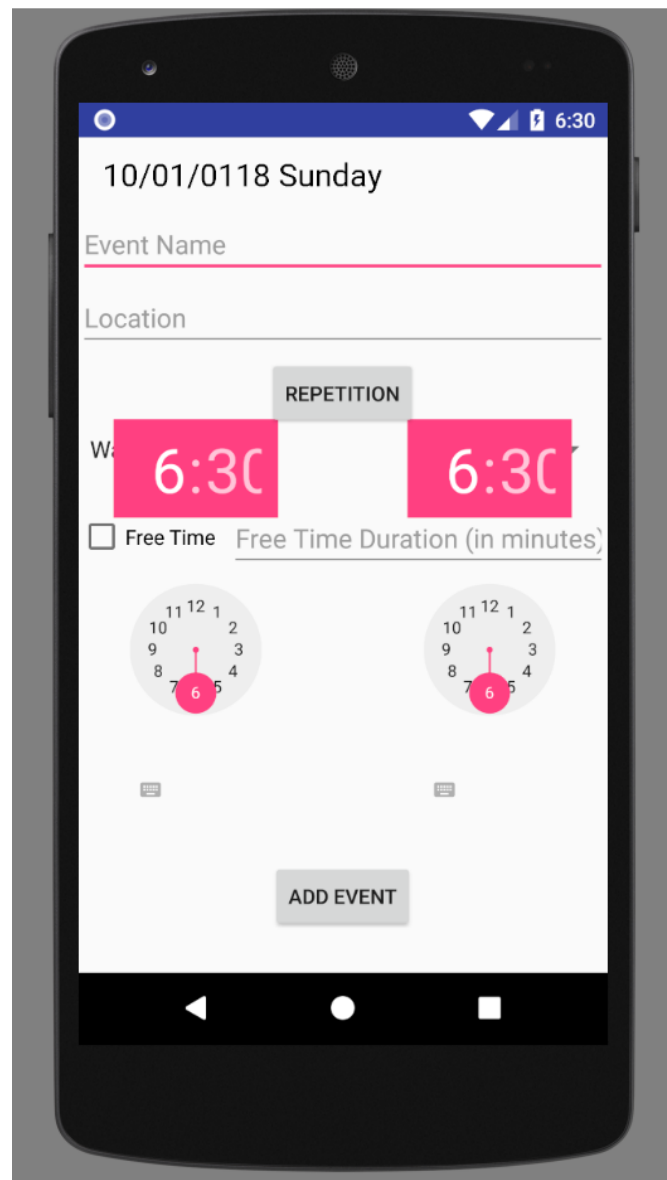
Figura 3.2: Event Creation View

The view has some missing graphic constraints that place the two clock pickers in the middle of the view, hiding the bar that made possible the selection of a travel mean for the event that is currently being created.

Another small issue is that the year of the selected day is wrong (0118 instead of 2018). There isn't any check for the correctness of the location inserted.

3.1.1 Test1

A first test has been carried out with a normal insertion of an event without free time and without repetition. After the click of the add event button the user is redirected to the main page but the appointment has not been added,although with a change of date on the calendar (change date and after rechange on the current date) the appointment is added, as shown in the image below,probably a little problem in refreshing the view after the insertion.
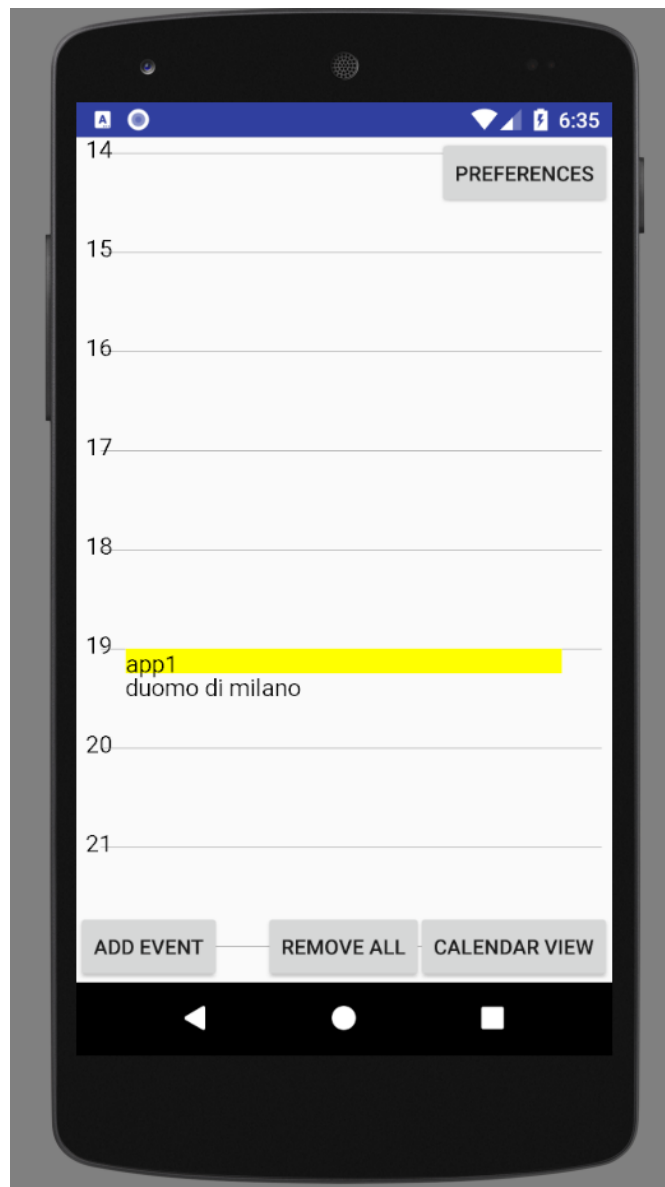

Figura 3.3: Insertion of an event

Another issue is represented by the insertion of an appointment with ending time which is previous than the starting time. In this case the application crashes.

3.1.2   Test2

Here we tried to add an event with repetition. This is possible by pushing the button on the event creation view, named "repetition". The user is redirected to the following view.
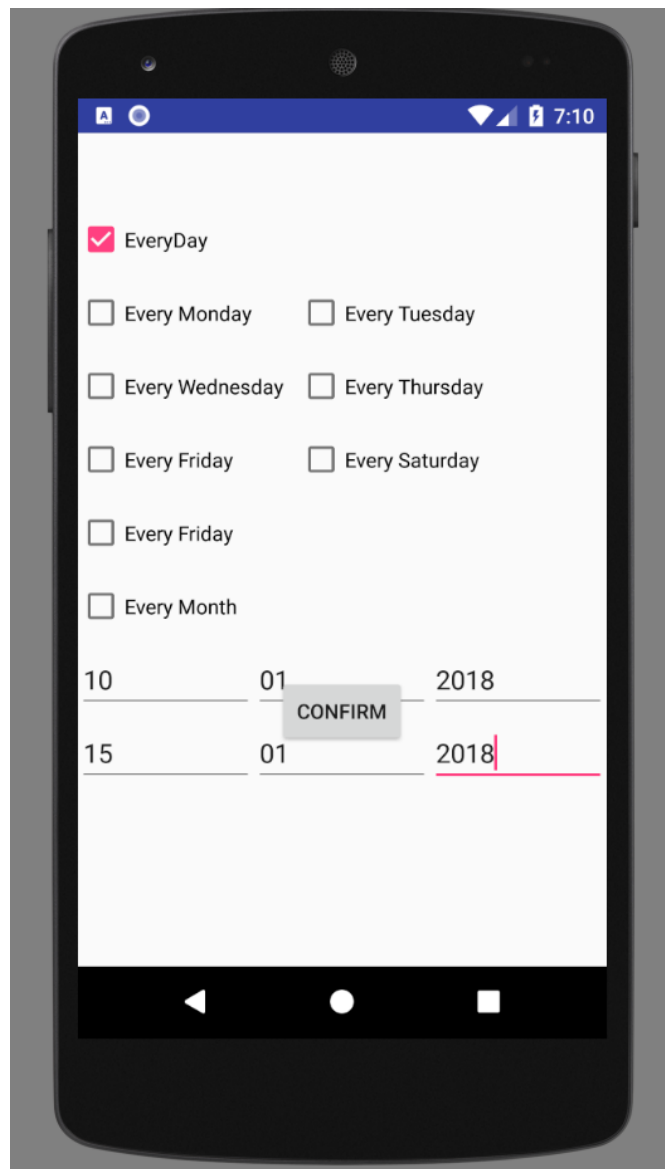

Figura 3.4: Repetition view

The confirm button has some missing constraints, so it stays on the middle of insertion

of the date fields.

After the click of the confirm button the user is redirected to the main page, after the refresh of the view the appointment is visible only on the first date of the slot selected for the repetition, on the other days in which the appointment should be shown nothing has been added.

3.1.3 Test3

Here we tried to add two overlapping appointments, in this case after the insertion of the second one the application crashes.

3.1.4 Test4

Here we tried to add a free time. When the save button is clicked, the application crashes.

3.1.5 Test5

As the other team has reported in their **DD**, after the insertion of any appointment, the temporal feasibility of that appointment should be checked, meaning that the application should calculate the travelling times and therefore check if the user will be able to attend two consequent appointments without being late. This functionality actually is not working: the application lets the user insert two appointments that differ for one minute, even if they are located in very far places, like "Via Golgi 42" and "Duomo di Milano".

Figura 3.5: Temporaly close appointments

3.2   Notification System

This functionality is declared to be implemented on the ITD document, but it isn't working: no notifications are sent to the final user.

3.3   Setting Preferences

By clicking the preferences button on the main page, we are redirected to the preferences page, where it's possible to set some user attributes, such as the ownage of travel means and of a transit travel pass, other than preferences on how many kilometers the user wants to travel with some travel means.

One can argue that this view isn't complete at all: it's not possible to set the distance contraints for all the kind of travel means (for instance, for the public travel means). Moreover, it's possible to set as an optimization criteria just the carbon footprint emitted by the means taken, but not the cost optimization or the time optimization, as stated on the assignment given us.
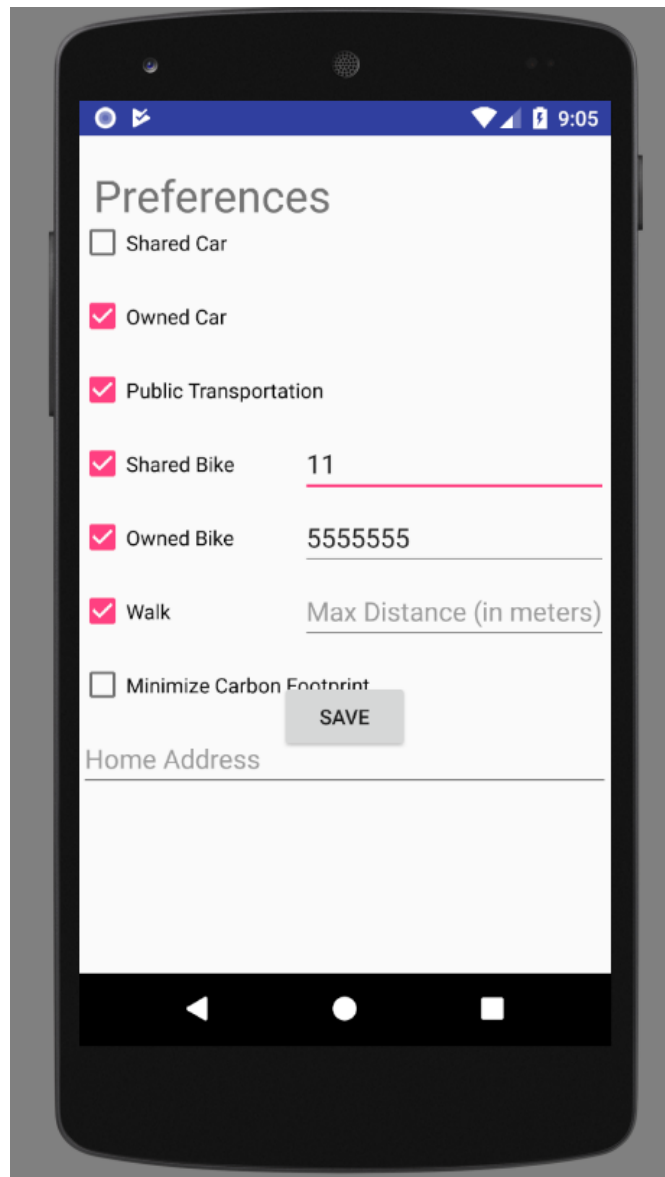
Figura 3.6: Preferences view

However, for the settings considered, the application works properly, letting the user save its characteristics.

3.4   Persistance Database to store events and free times

The user preferences and the user events are declared to be saved inside the device in a persistent database.

3.4.1   Test 1

Here it's tested the persistence of an user event. First, an event (let's say, a Software Engineering II lesson, located in "Via Golgi 42", lasting 2 hours, from 14.15 to 16.15) is added. After having closed the application, when it is opened again we can notice that the event it's still present.
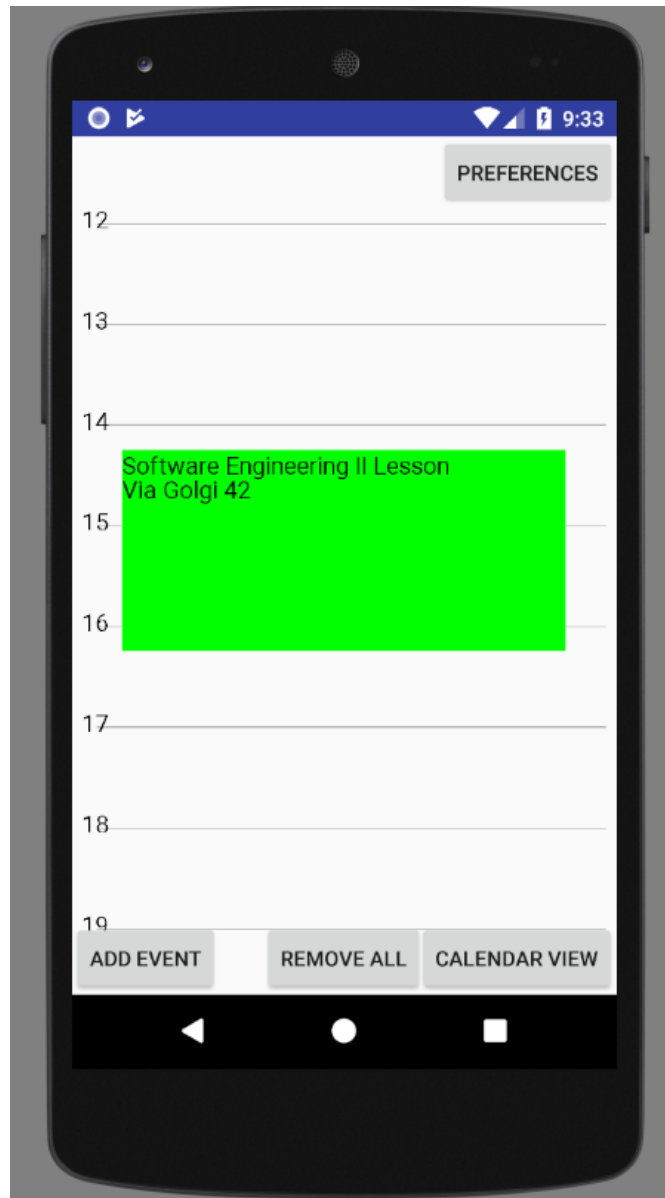

Figura 3.7: Persistence appointment

3.4.2   Test 2

Here it's tested the persistence of an user preference. First, the ownage of a car is set to true. After having closed the application, when it is opened again we can notice that the car ownage checkbox is still checked.
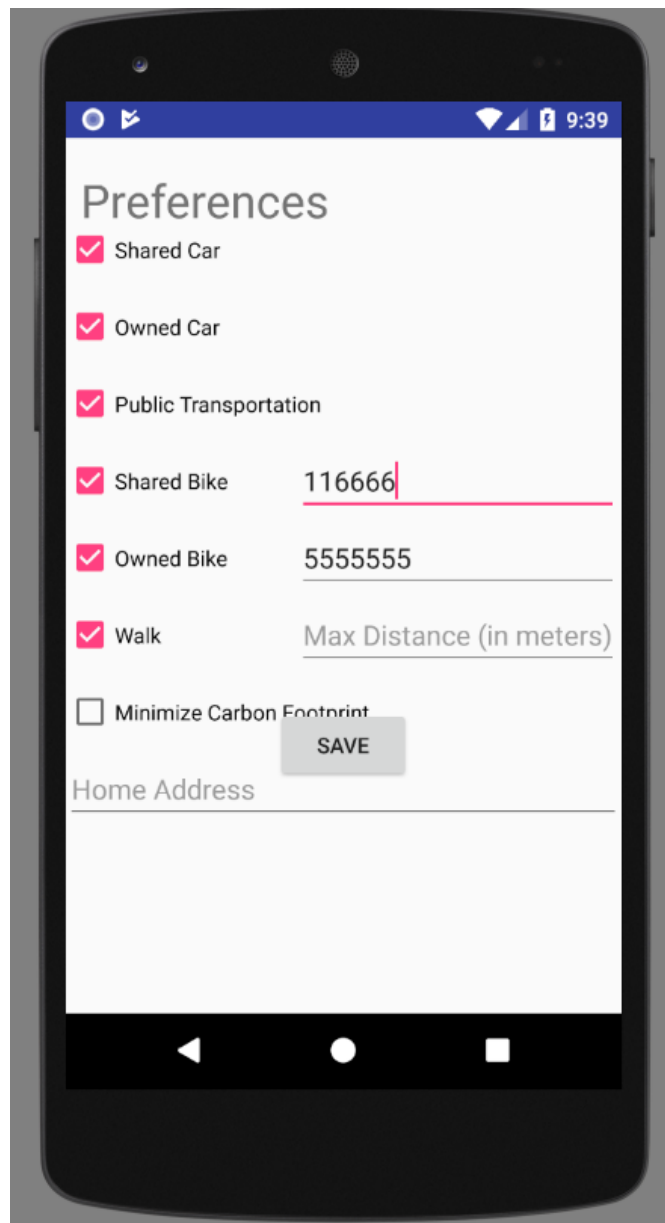

Figura 3.8: Persistence appointment

3.5   BikeMI API

This functionality is declared to be implemented on the ITD document but it isn't working, no response is returned at all during the execution of the application.

3.6   Wheather API

This functionality is declared to be implemented on the ITD document, but it can't be tested because the travel mean for an appointment is selected by the user, so the weather conditions are not checked.

3.7   Preferences Management

As declared, the user preferences are used during the computations to avoid some behaviours or to enhance others.

3.7.1   Test 1

This section is not easy to test, in fact most of the characteristics that it has are encapsulated in the code. However, the coherence between the user's owned means and the means that the application admits to reach an event can be checked: first it's set in the preferences that we don't own a car. Then is set up a new appointment and, as transportation mean, we set our owned car. The application allows the insertion of this appointment, even if this is incoherent with the preferences.
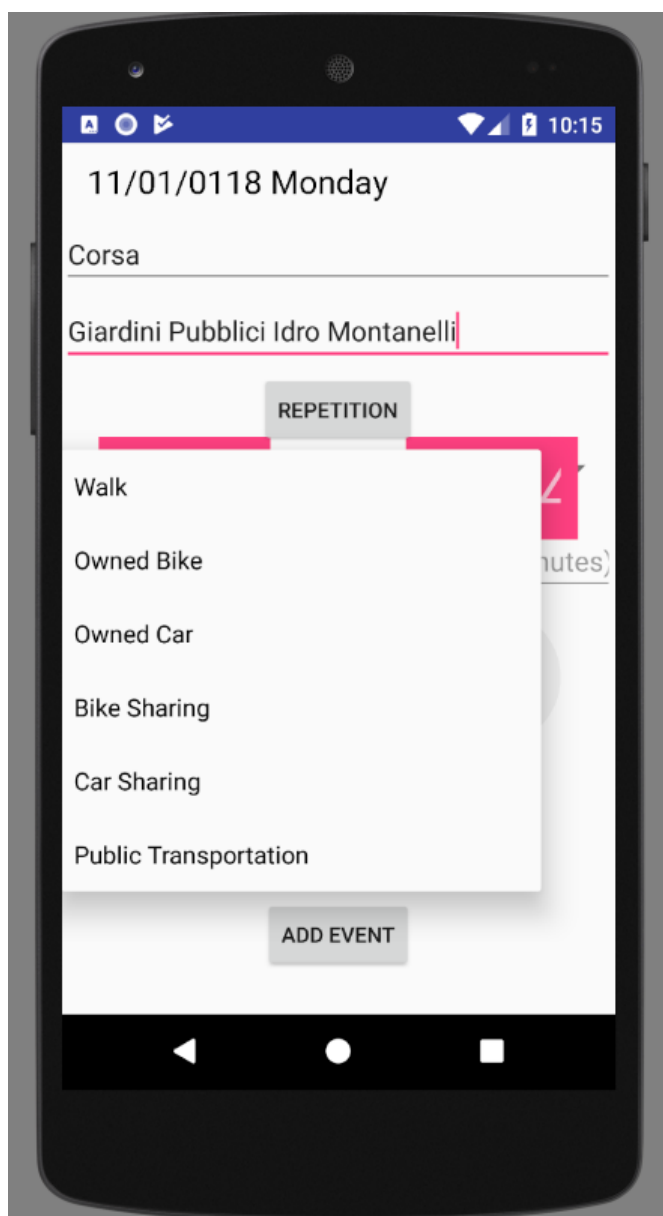
Figura 3.9: Preference Coherence

Chapter 4

Additional Points

The code has been written in a good and readable way, following a clean packages and classes organization, but it is poorly commented and sometimes difficult to understand. In the ITD document, tests are not shown nor described, however they can be find in the test folder on github repository.

Chapter 5

Effort Spent

- Federico Parroni: **2 hours**;

- Edoardo D'Amico: **2 hours**;

- Giovanni Gabbolini: **2 hours**.