Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: Fernando Paulo Correia Morais
Student ID: 12128058
Supervisor: Padraig De Burca

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature:_____
Date:_____

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

FPV Race Manager Application

Project Report

BSHCESD4
Fernando Paulo Correia Morais
x12128058

BSc (Hons) in Computing

National College of Ireland
2016/2017

Table of Contents

Executive Summary

The main objective of my project is to develop one application used to manage FPV racing events.

This application will receive data mainly from lap tracking hardware devices and will be responsible to keep track of that information as well as provide reporting of times.

It is expected to allow to provide several features required to manage such events, like pilots management, class management and to manage the organization of the rounds and.

1 Introduction

1.1 Background

The reason for choosing to develop this application is due to the fact that is a area that is quite new and fast evolving. The current existing solutions are quite simple and fail to provide most of the minimum requirements for its usage.

Companies in the market are focusing more in the development of the hardware that technically is more complex to develop and only provide basic applications to interface that hardware.

So i aim to provide a system that will be able to connect to several types of hardware devices capable of tracking laps and have the option to not only manage the races but also produce reports of the results.

1.2 Aims

The main objective of my project is to develop one application used to manage FPV racing events.

This application will receive data mainly from lap tracking hardware devices and will be responsible to keep track of that information as well as provide reporting of times.

It is expected to allow to provide several features required to manage such events, like pilot's management, class management and to manage the organization of the rounds.

1.3 Technologies

The application ideally will be cross platform, allowing the user to use it independent of the operating system used. Linux, OSX or Windows.

In order to allow a easy to port and convert application that runs on all these operating systems I have researched and found NodeJS as a good solution.

It has a good cross-platform compatibility, special at I/O access that for example with Java is more complicated to achieve.

I will be using more specifically Electron from GitHub, initially my aim was to use NwJS (formerly known a node-webkit) but after experiencing several problems and some difficulties making some modules and code work I went to search for an alternative and found this one.

It is developed by GitHub and is what is used for example for Atom the IDE and several other applications including Visual Studio Code from Microsoft.

It presented more up-to-date product with more support and development, as well as much better documentation.

And the fact that companies like Microsoft has selected it as base for some of their tools gave me the confidence that it would be a better choice.

In regards to data storing and because the application doesn't have much requirement in regards to to I'm aiming at using SQLite. The choice was made purely due to the amount of information found for it.

The application itself could use flat files to store its data as there will be no need to perform any action in it that would require or would bring any benefit of using a more advanced DBMS.

1.4 Special resources needed

Will need to use some hardware for part of the test of the application.

This will be the lap tracking system, not all existing ones will be possible to use and for that software mimicking their behaviour will be used instead.

2 System

2.1 Requirements

2.1.1 Functional requirements

2.1.1.1 Use Case Diagram

Below Fig.1 is the overall use case diagram for the application

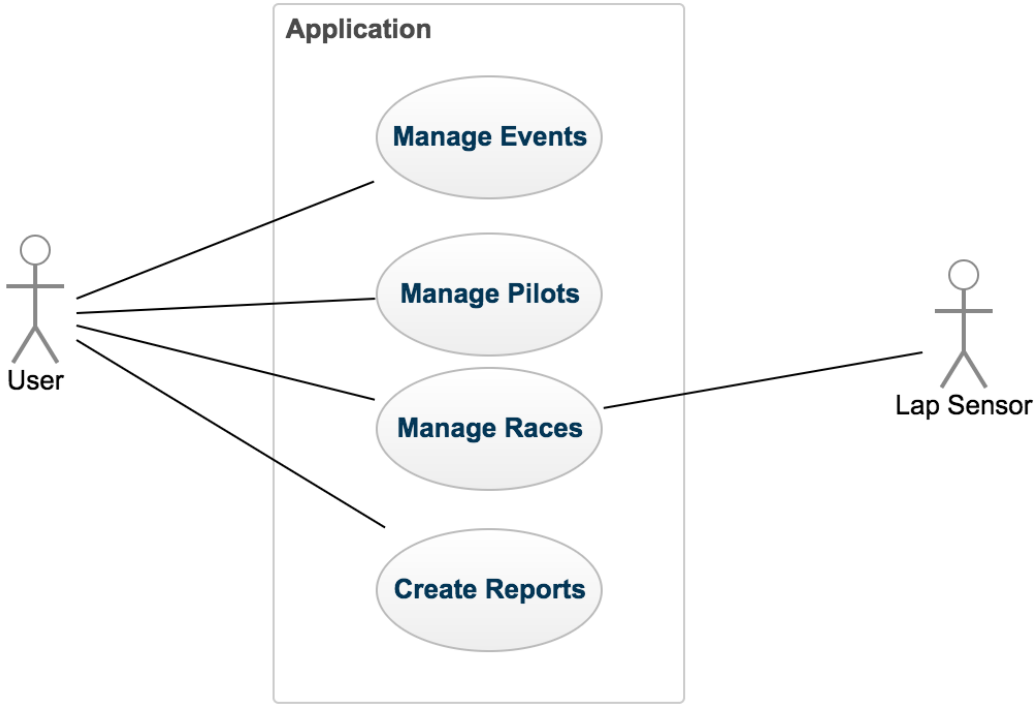


Fig.1 - Use Case Diagram

2.1.1.2 Requirement 1 - Pilot Management

Scope
Creation and editing of pilot details

Description
The application will allow to add, edit and delete pilot entries. That will contain all the details related to the person competing.

Use Case Diagram

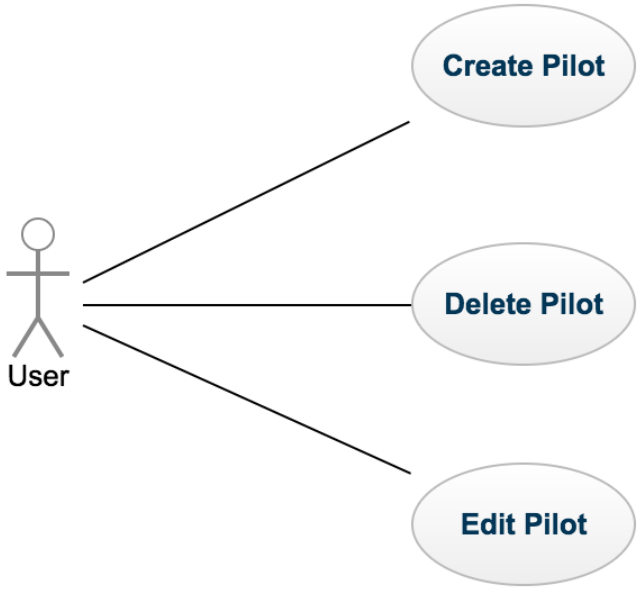


Fig.2 - Pilots Use Case Diagram

2.1.1.3 Requirement 2 - Event Management

Scope
Creation and management of events in the application

Description
The application will provide means for the users to create new events and edit existing events. Allowing to define all details related to an event, such as pilots present and races format.

Use Case Diagram

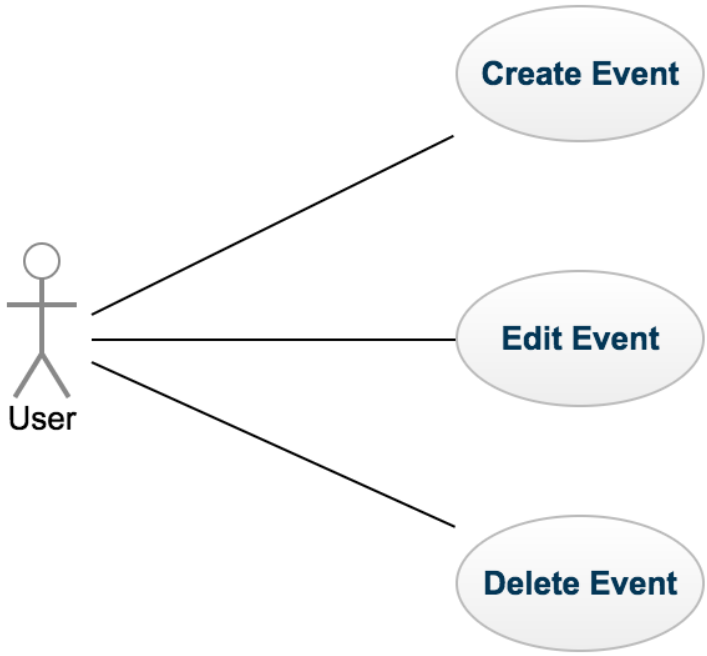


Fig.3 -Events Use Case Diagram

2.1.1.4 Requirement 3 - Race Management

Scope
Creation of races and control them, such as start, stop. See the tracking of laps and check times.

Description
The application will allow to define races within an event and define their format.
Control the execution of the races and pilots present in each race.
The user will be able to visually see the laps, times and pilots in the GUI.

Use Case Diagram

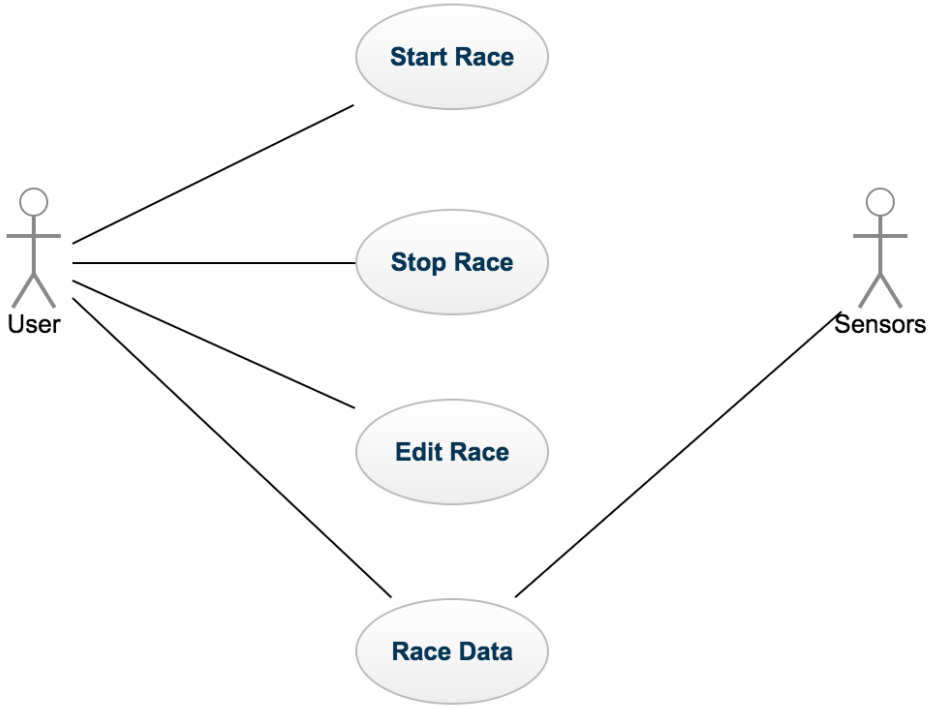


Fig.4 - Race Use Case Diagram

2.1.1.5 Requirement 4 - Reporting

Scope
Generate and visualize the data in reports

Description
The application will provide means for a user to report on different elements of an event, it's races and pilots. Producing reports containing times and count of laps.

Use Case Diagram

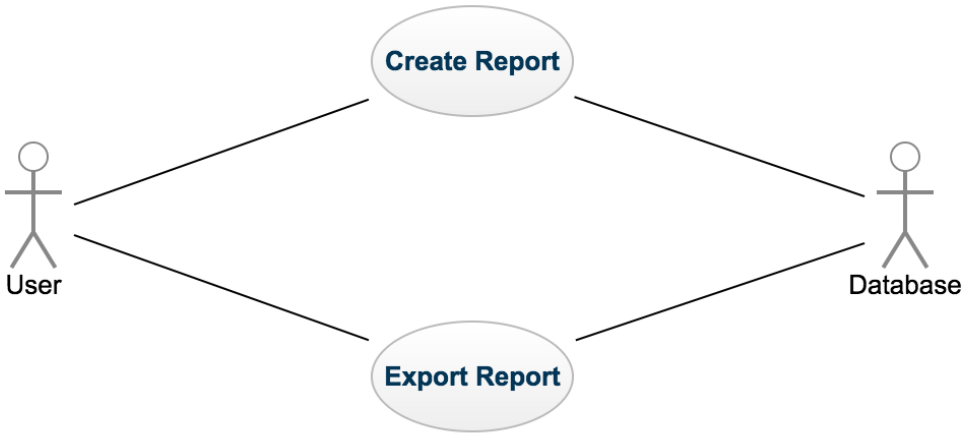


Fig.5 - Race Use Case Diagram

2.1.2 Data requirements

In terms of data storage, the application will not require any specific type of DBMS.

As the bulk of the data to store will be the lap tracking that won't require any type of indexing as they will contain the timestamp. So, it can even be store in flat files.

Still I have opted to use SQLite, is a simple and light DBMS that is supported in multiple platforms and doesn't require any server or service running to operate making it quite simple to deploy as all that is required is to share the .db file.

With this I can easily deploy it cross platform without requiring the user to perform any extra task other than install the application. As well as I can pre-populate the shared database with already some test or example data.

2.1.3 User requirements

The user will be expected to use one of the three different operating systems:

- Linux
- OSX
- Windows

In one of the variants of 32 or 64bits.

The machine should have at least one USB connection port to be able to connect to the hardware devices or a network interface depending on the hardware sensor used.

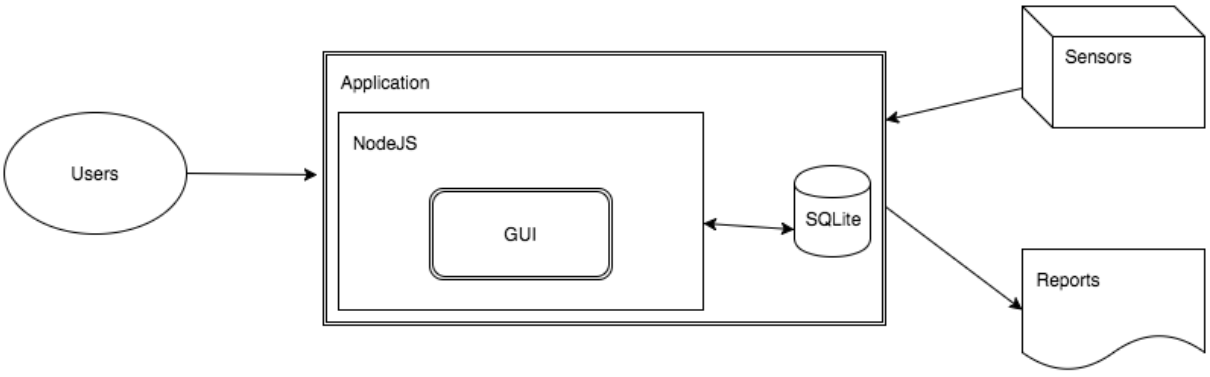
2.1.4 Usability requirements

In terms of usability the application should be guided by two principles:

- **Ease of use**: the application should be ease enough and self-explanatory, meaning that the user should have no need to use documentation in order to understand its core functionalities
- **Consistency**: provide a consisted interface and behaviour, in what refers to colours, errors and messages.

2.2 Design and Architecture

The architecture diagram below defines the overall structure of the application.



It is composed of the main application that will be running using Electron/NodeJS and using a SQLite database to store its data.

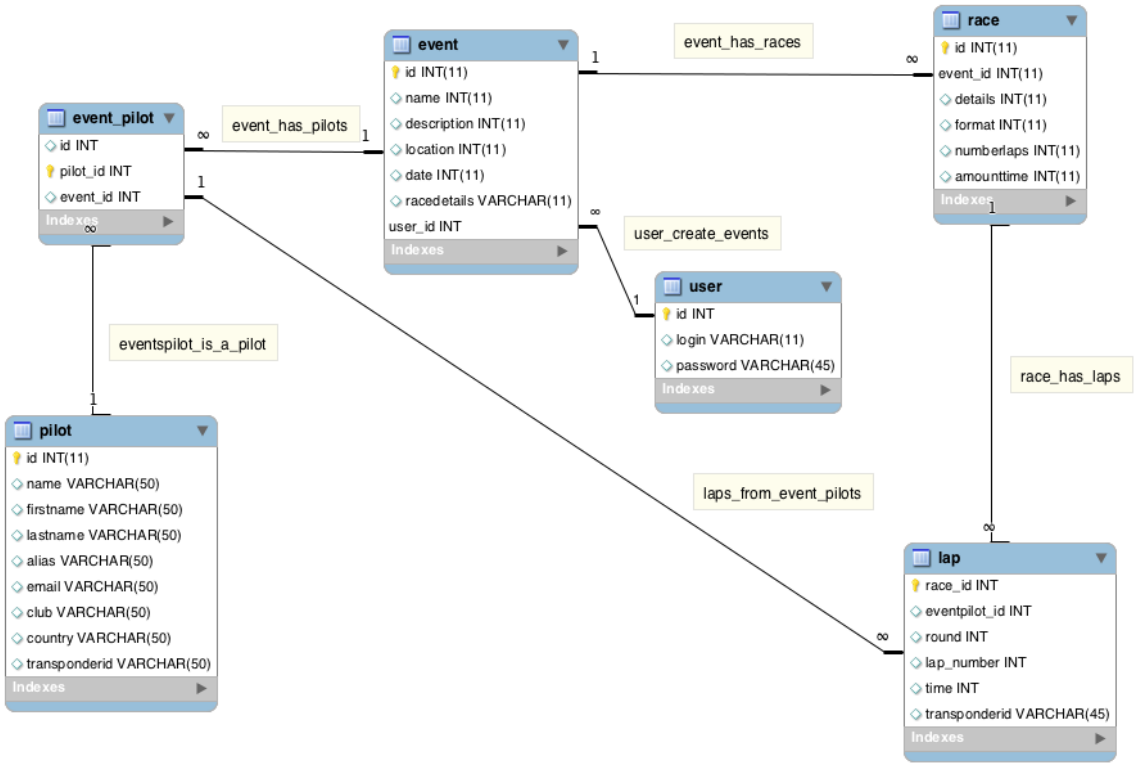
It will be received data from the external sensors and input data form the users and can export the data into reports.

2.2.1 Entity Relationship Diagram

The following diagram represents the entities present in the application and their relationships.

Initially I had opted to create extra entities for rounds and heats but further analysed allowed me to simplify it but combining information into single entities like laps that now contain details such as round making redundant the usage of a external entity.

This proved to greatly simplify the database schema and the queries used to retrieve data from it without loosing capabilities.



2.3 Implementation

This application is mainly developed in JavaScript, the main language supported by the Electron framework. Added to it all the styling is done in the same way as a web application and it uses CSS and or it's structure HTML code.

The initial aim was to find and use a MVC framework for this project.

The first found was BaselJS while functional is still in a very early stage of development and didn't afferent the flexibility and stability expected in order to develop this application. The second one found is EmberJS, this is a more stable MVC framework for JavaScript but is not directly aimed to be used in Electron. For it I found very few documentation or examples of it been used in Electron, so it proved to be a risky solution due to the lack of documentation.

At a later stage of the project I found KnockoutJS that is a MWWM framework but due to the advance state of the project I decided not to refactor the code to use this framework, but it would be something for future development to experiment and see if this could be used in the project.

Due to this the approach was to find and use instead individual frameworks or libraries. The offer is quite vast what ended up making more complicated to find the most appropriate for this project.

After experimenting and running some tests with several libraries, AngularJS, VueJS, ReactJS and others, the choice for a more simple approach combining elements of AngularJS came from the PluralSight courses that I have enrolled in and that provided good insights into this topic.

So solution selected was to use AngularJS and angular addons depending on the task required.

For example Angular UI.Router for the view rendering and navigation.

The examples given allowed for a very simple and light code structure and also reduced the learning curve that would exist in case I had opted for a framework that I had never used before, were the risk of encountering some limitation further down the development path was problematic due to the time constrains for this project.

2.3.1 Frontend

With the lack of a proper MVC framework the structure followed was the recommend by AngularJS.

```
app/
----- controllers/
---------- mainController.js
---------- otherController.js
----- directives/
---------- mainDirective.js
---------- otherDirective.js
----- services/
---------- userService.js
---------- itemService.js
----- js/
---------- bootstrap.js
---------- jquery.js
----- app.js
views/
----- mainView.html
----- otherView.html
----- index.html
```

The code is split between views and controllers and the controller performs the task of the model as it will include all the CRUD operations to be executed against the database.

In the application, there is one controller for all the main sections, for example the pilot controller.

```
(function () {
    'use strict';

    console.log("Loading pilotController...");

    angular.module('myApp')
        .controller('pilotController', ['pilotService', '$q', '$mdDialog',
PilotController]);

    function PilotController(pilotService, $q, $mdDialog) {
        var self = this;

        self.selected = null;
        self.customers = [];
```

```
        self.selectedIndex = 0;
        self.filterText = null;
        self.selectPilot = selectPilot;
        self.deletePilot = deletePilot;
        self.savePilot savePilot;
        self.creatPilotr = createPilot;
        self.filter = filterPilotr;

        // Load initial data
        console.log('Fetching all data...');
        getAllPilots();
```

Within the controller all actions related to it are defined, such as creating new pilot, edit details or delete a pilot.

As views we also have one per area and linked to the controllers.

These views are html files that contain the HTML template code to display the information in the application UI.

```
<div style="width:100%; height:100%" layout="row">
    <sidenav class="site-sidenav sidenav-left whiteframe-z2"
                component-id="left"
                is-locked-open="$mdMedia('gt-sm')">

        <toolbar layout="row" class="whiteframe-z1">
            <h1>Pilots</h1>
        </toolbar>
        <input-container style="margin-bottom:0">
            <label>Pilot Name</label>
            <input required name="pilotName" ng-model="_ctrl.filterText"
ng-change="_ctrl.filter()">
        </input-container>
        <list>
            <list-item ng-repeat="it in _ctrl.pilots">
                <button ng-click="_ctrl.selectPilot(it, $index)" ng-
class="{'selected' : it === _ctrl.selected }">
                    {{it.name}}
                </button>
            </list-item>
        </list>
    </sidenav>
```

For last we have the services, such as the dbService.

These are singleton classes that provide functionality used by the controllers.

```
(function () {
    'use strict';
    var sqlite3 = require('sqlite3').verbose();

    // Create new db connection
    var dbConnection = new sqlite3.Database('appDatabase.db');

    console.log("Loading dbservice...");
```

In the example above it shows the creation of the connection to the database used to store the information from the application.

Or the background async tasks that are executed to receive the data from the serial connection to the sensors.

Electron and NodeJS, despite be a JavaScript platform it allows to have async task running in the background even without eh need to create a visible window.

This allowed me to create services that are contenting running and that can communicate with the UI

This is performed using the electron IPC modules.

```
'use strict';
const { ipcRenderer } = require('electron');
const task = require('../shared/task');

window.onload = function () {
    ipcRenderer.on('background-start', (startTime) => {
        ipcRenderer.send('background-response', {
            result: task(),
            startTime: startTime
        });
    });
};
```

In the module above I start a proceed defined in a shared task.

```
'use strict';

// counter task

module.exports = function task() {
    let result = 0;

    for (let i = 0; i < 100000000; i++) {
        result += i;
    }

    return result;
};
```

In this case is a counter task and whenever there is an update it will send a message back to the main UI.

This allows to have functions like a timer that otherwise done in JavaScript and directly in the UI would block other actions from executing.

2.3.2 Storage

Initially the development of the application I started by using a MySQL database, but due to its lack of portability I decided to look for a different database system.

It ad the advantage that it could be stored online and accessed from anywhere but the requirement for a constant network connection and it case it was local for it to be manually installed didn't provided a suitable solution for this application requirements.

So, after researching on suitable database systems I decided to go for storing its data the application using a SQLIte database. This allowed it to be deployed without need to install the database software as SQLite is a self-contained system.

The single file created is appDatabase.db and electron connects to it in a native way using the node-sqlite NodeJS module. One of the modules that I was required to compiled manually.

Its structure is comprised of several tables.

Pilot

This table stores all the pilot details, such as name, contact and club.

Event_pilot

This table sets the relation between a existing pilot and it's presence in a event.

Event

This table contains all the information related to the event itself, such as dates and location.

User

User table is a sample table used store the login username and the encrypted MD5 password hash**.**

Race

The race table contains all details related to a race in an event.

Lap

The laps table tracks all the lap entries from a given race and pilot. This table contains the times later used to compute the results.

The application itself won't be storing any critical or sensitive data, still due to the presence of personal details such as name and contact email I have added a login feature, this prevents any user from accessing the application.

So it doesn't' restrict access to the data itself but prevents any user with access to the machine where the aplliatin is running form accessing it's details.

The login details are stored in a encrypted way in the database, unfortunately SQLite does not directly support any form of encryption directly, so I can't use like with MySQL built-in content encryption.

The alternatives are to either user a external plugin for it like SQLCipher or handle this encryption at code level.

The first option, SQLCipher is an open source extension to SQLite that provides transparent 256-bit AES encryption of database files

The second approach, the one I selected, it allows to only encrypt selected fields and in case there later the need to perform a backup and restore of the database file that can be performed much easier.

So for the user password I generate a MD5 hash string that is later stored in the database, so the direct password string storing is prevented and in case I decide to move to a more encryption method that would be transparent to the database system as well.

This is handled by a AngularJS module called angular-md5 and in terms of implementation is quite simple.

```
$scope.message = 'Your password Hash is: ' + md5.createHash($scope.password || '');
```

Late for authentication is simply compared the generated hash from the password the user introduced against the hash stored in the database.

If the applications develop to a stage where more sensitive data is stored I would consider moving into a different database system that would provide direct encryption.

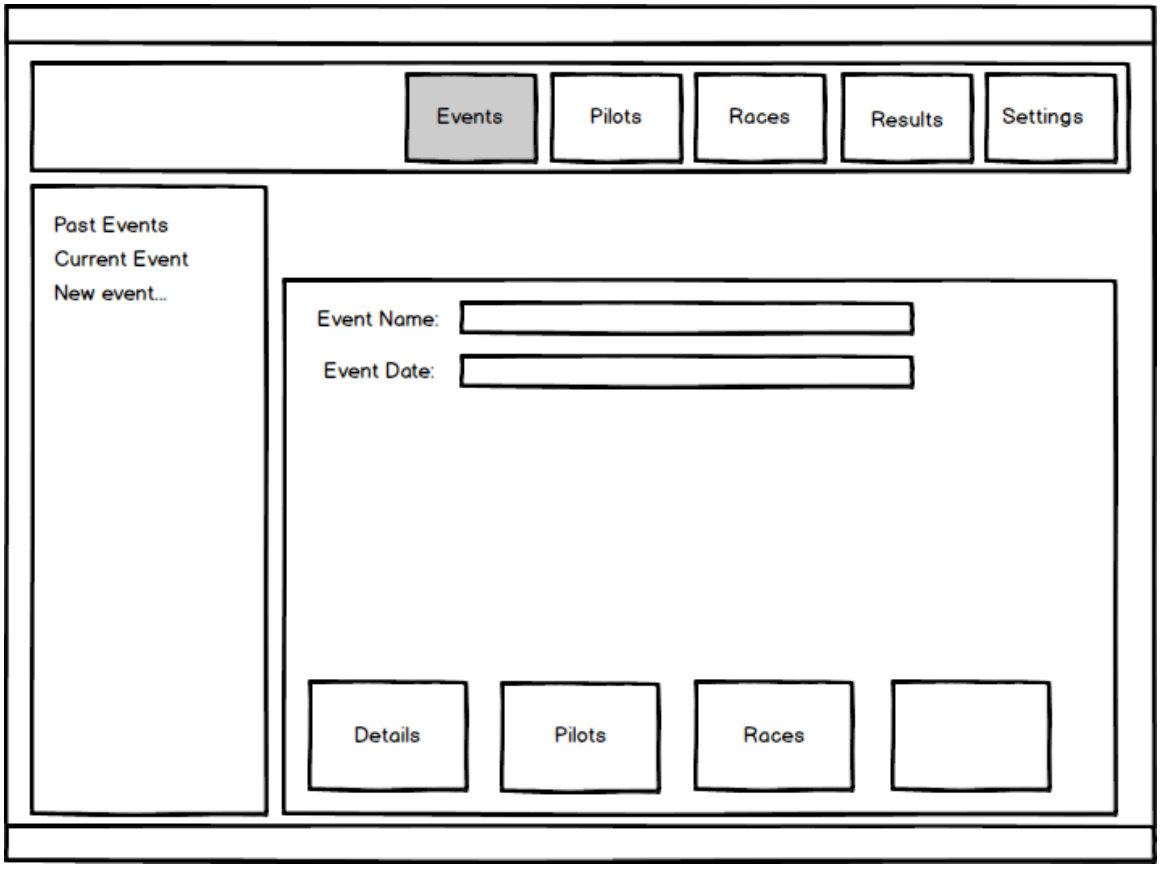
2.4 Graphical User Interface (GUI) Layout

2.4.1 Low-fidelity prototype

Below are some examples of the mocks for the user interface created during the design process.

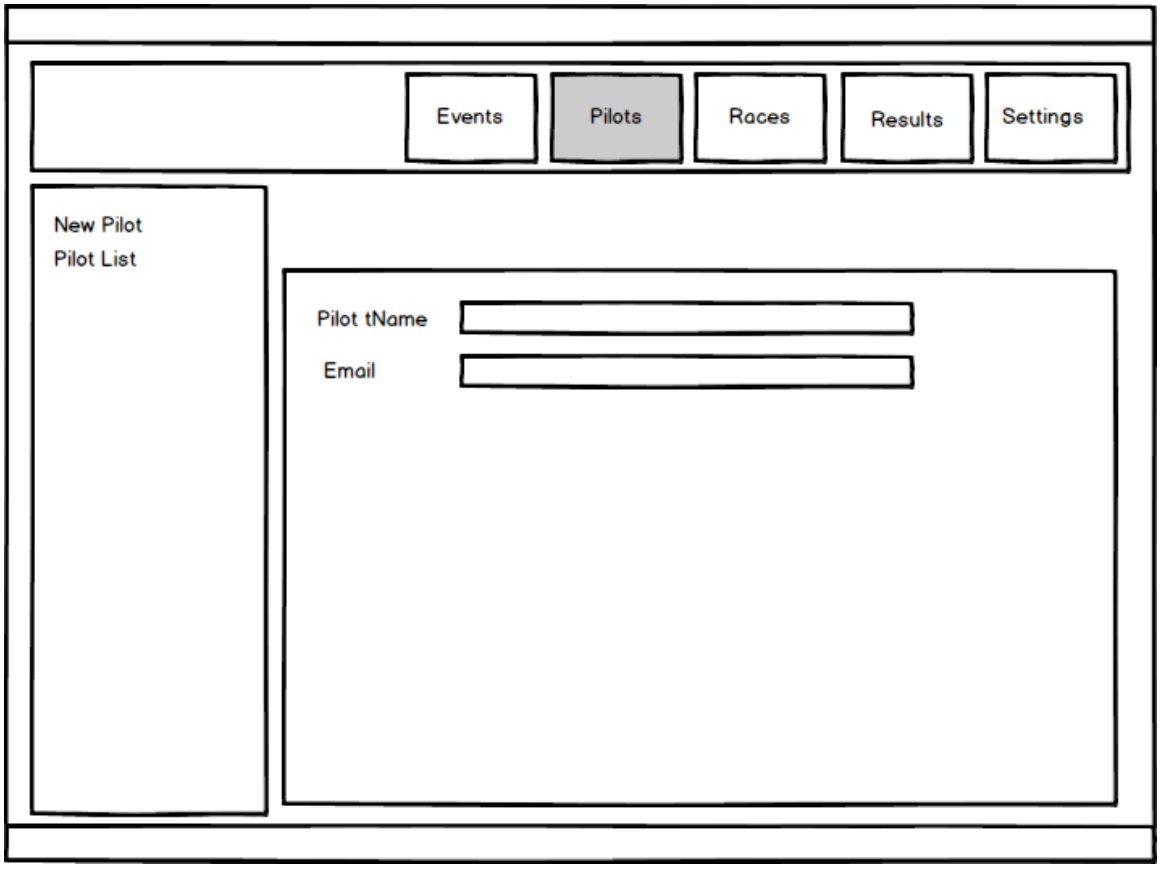
Those were created having in mind usability principles, such as the simplicity by having the initial screen with minimal options available to the user to choose from and principle of structure and reuse where elements with similar functions are always displayed in the same location or with the same visual style so the user can easily identify where they are and their purpose.

Main Event section, where events can be created and existing ones edited.



Notice the simple big buttons easily readable and with clear indication of the content they relate to.

Next an example of a sub-section, the pilot area where pilot details can be edited and new pilot entries created.



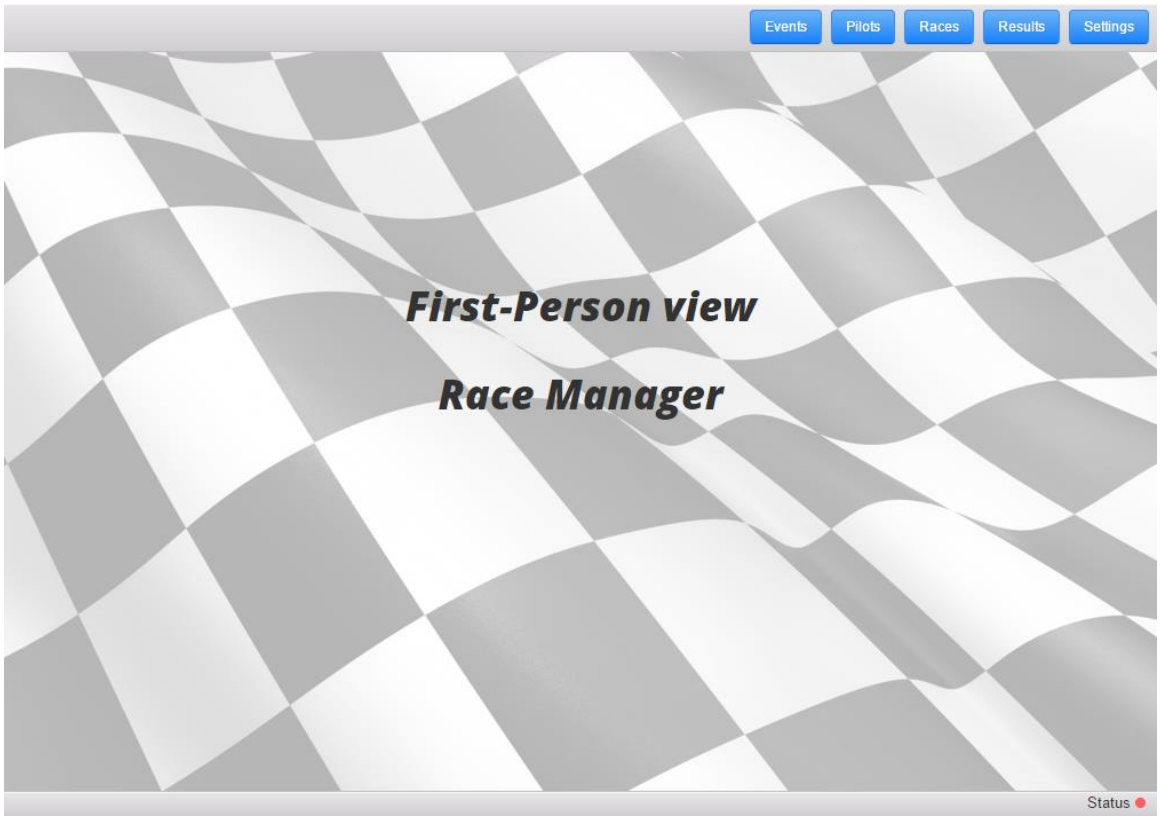
Again the same simple and easy to use layout, using the principles of structure and reuse.

2.4.2 Application Layout

To implement and create the layout seen in the previous mocks I used elements and styling provided by two different frameworks.

The base one is *Bootstrap*, a common web framework that can also be used and included into Electron projects. This framework allows to style the UI elements as well as provide aids to correctly format a UI layout with it's grid system.

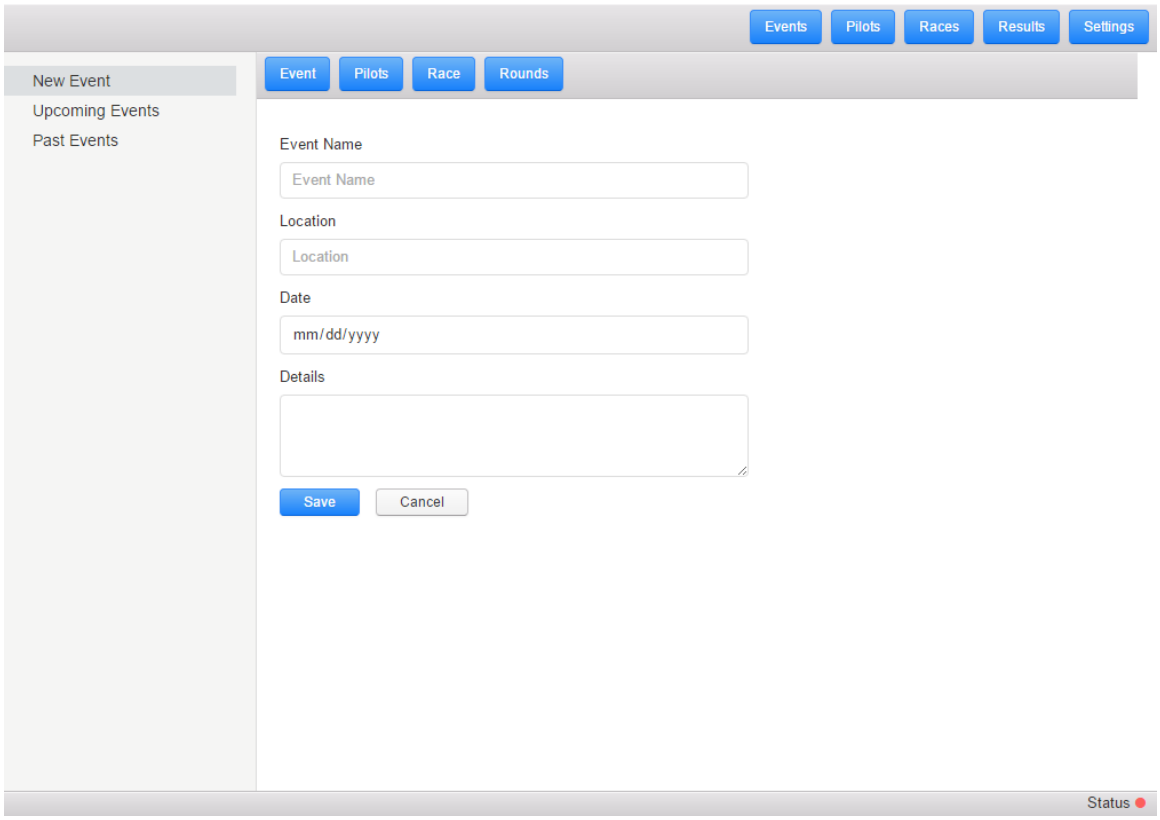
Adding to it I also used parts of *PhotonKit* library that extends Bootstrap and provides elements that give applications a more desktop look, this framework is specifically aimed at electron applications.



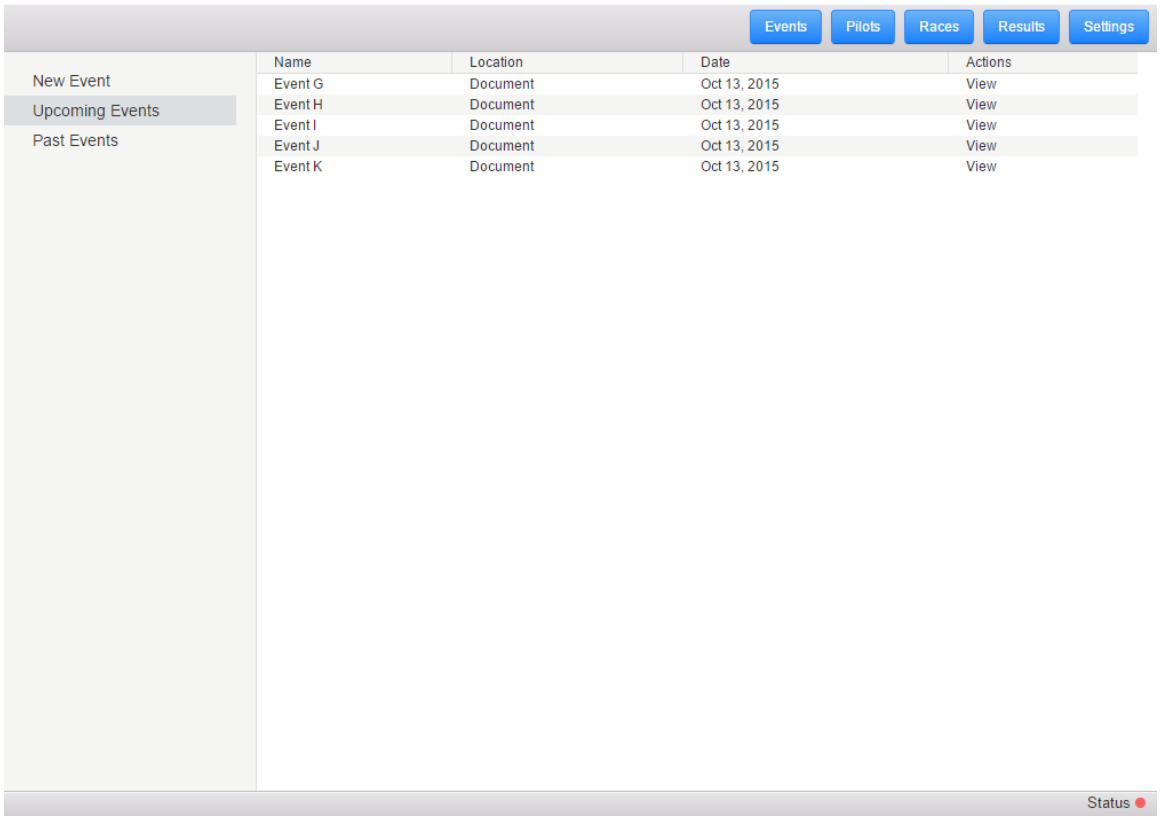
The opening page is quite simple to the user but provides quick and easy access to all options available.

It used a top navigation bar and all the content is displayed in the centre panel.

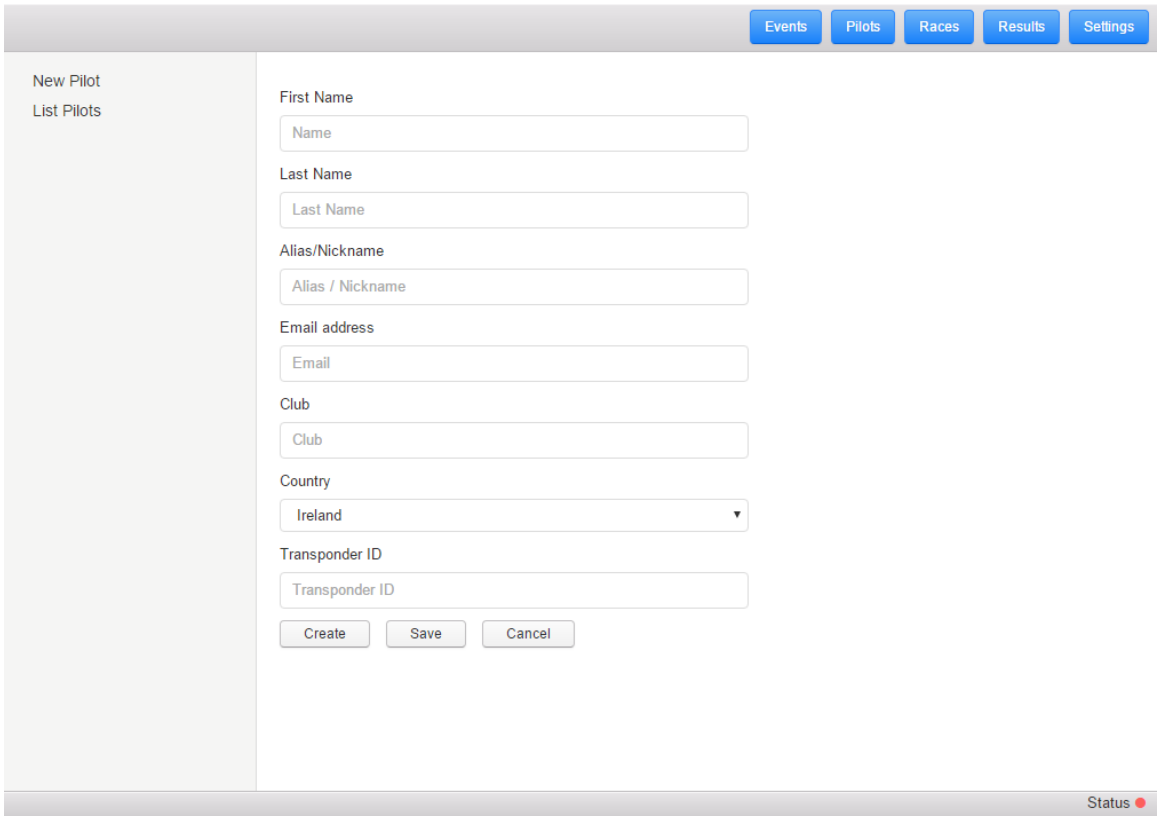
This layout format will be used across all the areas of the application.



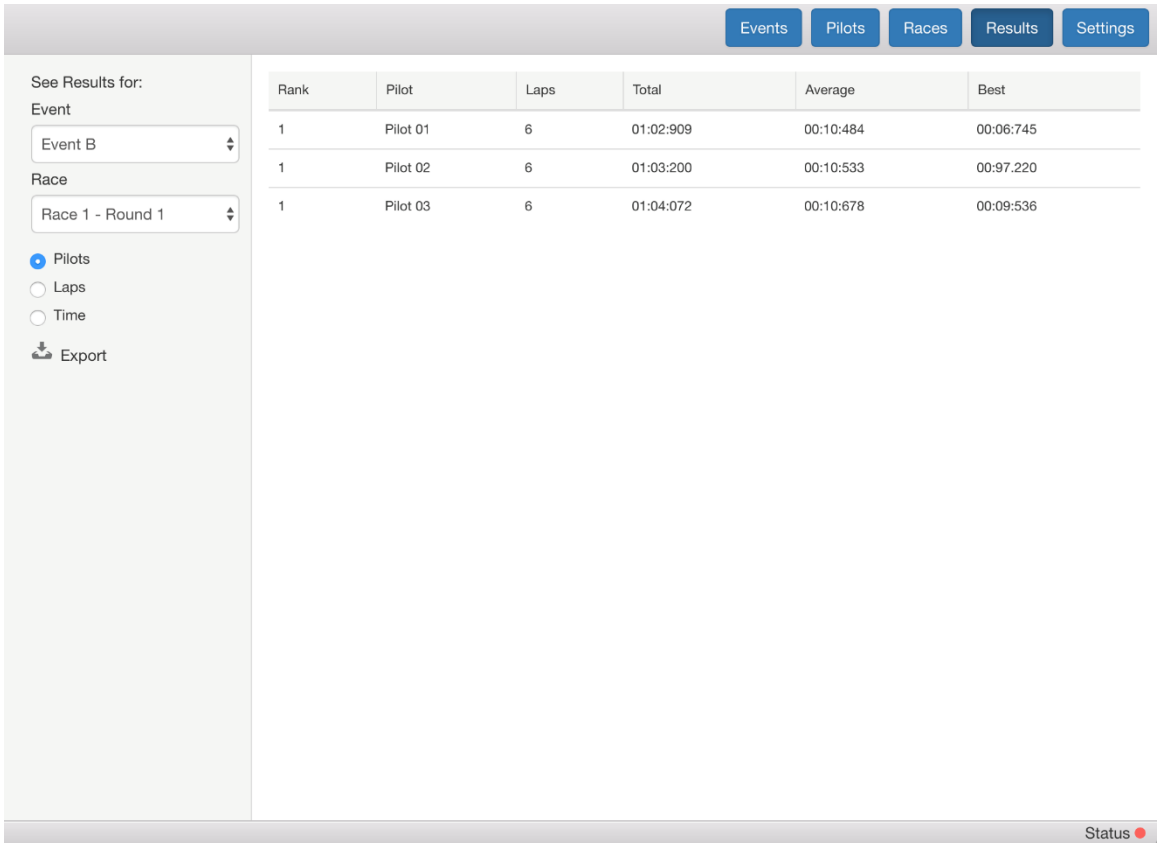
Fundamental part of the application is the creation and editing of existing events, in the screenshot above can be seen the event creation page and the options it provides.



Few other options are available to the users, just as consult the list of all existing upcoming events or view past events.

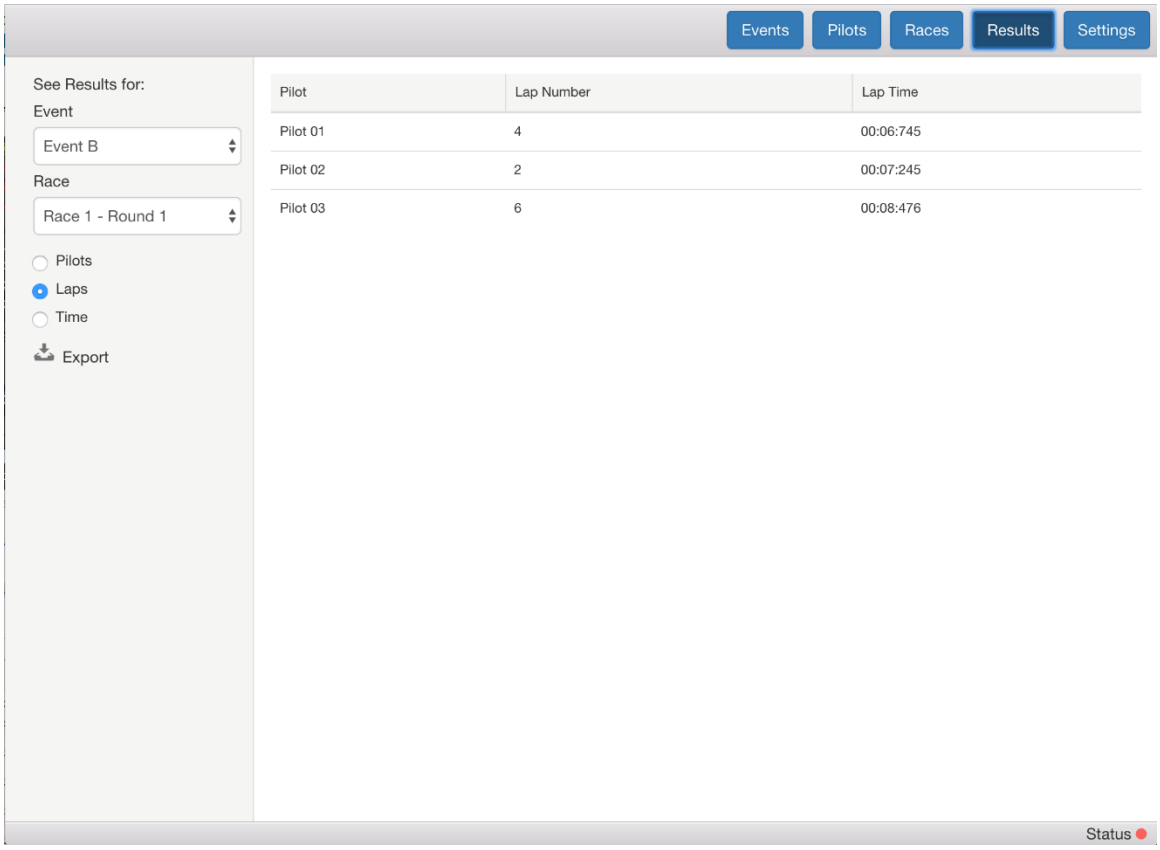


Another key area is the managements of Pilots, where all actions related to that can be found. Here the user can create or update the details of existing pilots.



The final key area is the results area, in this section is possible to list and review the results from all the events recorded in the application.

Information and details such as the top times and lap times per event and race are possible to be identified.



It allows to organize the data over different ways such as by pilot, lap or time and allows the user to export this data if that is required.

2.5 Usability & User Testing

As this application targets a very specific audience and usage the user testing proved to be more difficult than expected as the users that would be testing it would need to be acquainted with the terms and structure of an event an it's components.

Due to this not all testing methodologies could be applied and I selected the following ones to be performed.

2.5.1 Identification of Stakeholder

Here I can only identify primary stakeholders. There will be the staff from a racing event as the application itself is not meant to be used by anyone else.

Primary: Event staff

If instead we look at the data it generates, then we can identify secondary and tertiary stakeholders.

Secondary: Will be the pilots itself and event spectators

Tertiary: Clubs involved or event sponsors

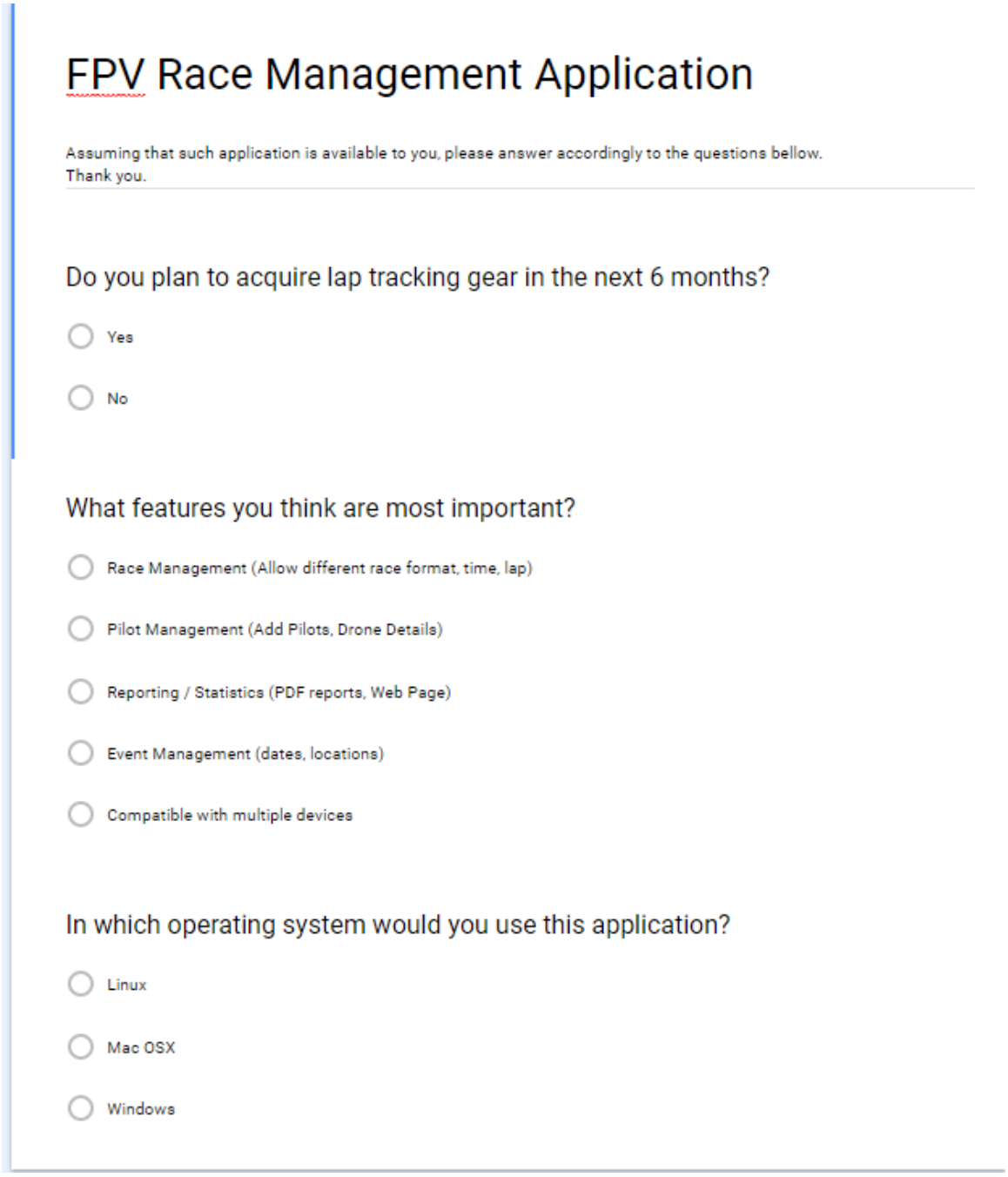
2.5.2 Data Gathering

I have selected the interview/questionnaire method for the data gathering, the selection of candidates didn't use any specific restrictions other than them not be associated to this project and be somehow acquainted with FPV events.

The selected candidates were given and asked to go over some questions done in a online questionnaire.

Questionnaire

Aiming to be simple and concise I have elaborated a few questions that would focus on the key points I address in this application.



FPV Race Management Application

Assuming that such application is available to you, please answer accordingly to the questions bellow.
Thank you.

Do you plan to acquire lap tracking gear in the next 6 months?

- ○ Yes
- ○ No

What features you think are most important?

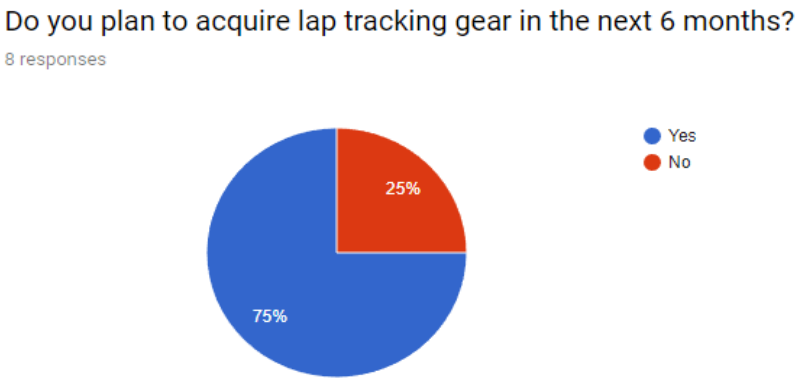
- ○ Race Management (Allow different race format, time, lap)
- ○ Pilot Management (Add Pilots, Drone Details)
- ○ Reporting / Statistics (PDF reports, Web Page)
- ○ Event Management (dates, locations)
- ○ Compatible with multiple devices

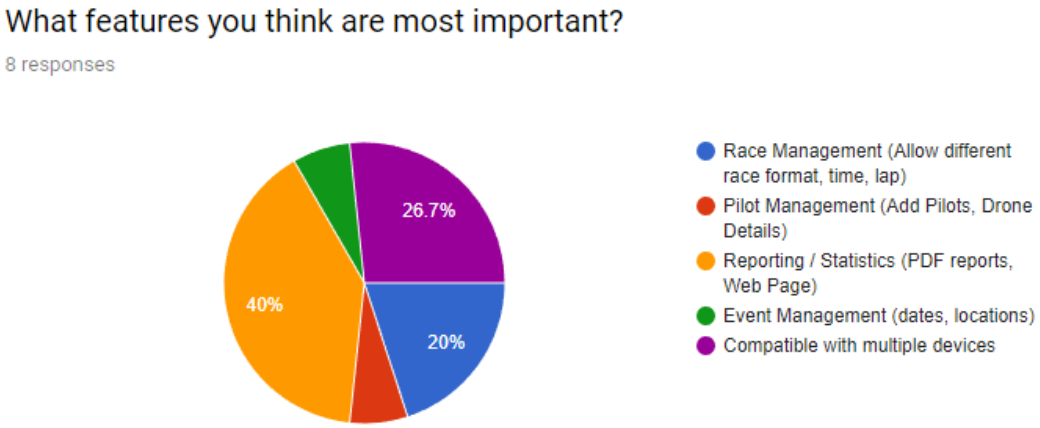
In which operating system would you use this application?

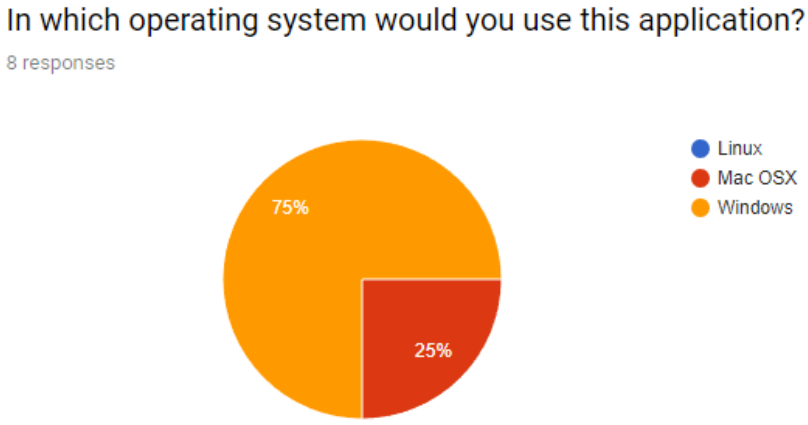
- ○ Linux
- ○ Mac OSX
- ○ Windows

Results

The following charts represent the results from the questionnaire above.







In terms of results the key areas identified are reporting and be compatible with multiple devices, this last one can be further seen when the users are asked in which platform would they run the application, with Windows showing more votes followed by OSX. Linux has no expression in the results but as the code from the application will be also possible to execute in that platform all three will be covered by the result of this project.

2.5.3 Testing Methodologies

Trunk Test

The user is presented with a application or site and is asked to try and quickly identify where are certain areas or where certain tasks can be performed and it's comments are later collected.

Questions asked:
"Where can you edit a pilot name?"
The fact that the pilot button is presented in a central navigation place seems to help a lot in its identification and localization. With all users quickly identifying it.

"Where do you register as a new pilot?"
Again, having the option placed at all times in the top right becomes common to the user that quickly identify where to go.

"How do you see a pilot details?"
Same as before, the element allows for a quick identification and navigation, with all users quickly selecting it.

More similar questions were asked related to other areas such as events and results and all performed in similar way.

Some positive feedback was received in regard to its simplicity and easy to use.

In overall the outcome is that the application is easy and simple to use and users don't experience confusion or doubt when using it.

2.6 Evaluation & Testing

2.6.1 Unit Testing

As the application run into the electron framework the to perform testing at this level I used the Spectron Unit testing framework.

Spectron is a electron unit testing framework develop by the same team that develops Electron so it's well suited for this task and easy to implement.

Still It does not provide unit testing for the code itself that runs in the application it does allows to test the basic electron operations such as the windows live cycle.

This is helpful as allows to quickly identify if a change in the code prevents one of the background windows used for services like the serial port from starting as those are not directly visible.

For testing the code itself it was used a combination of two tools Karma and Jasmine.

The first Karma, is developed by the AngularJS team itself and aims

And Jasmine is a Javascript Unit testing framework that is most commonly used to work with Karma and provides a quite simple descriptive language to definet he tests.

```
  describe('.findPilotById(id)', function() {



    // A simple test to verify the method findPilotById exists
    it('should exist', function() {
      expect(Pilots.findPilotById).toBeDefined();
    });

    // A test to verify that calling findById() with an id, in this case '2',
returns a single user
    it('should return one pilot object if it exists', function() {
      expect(Pilots.findPilotById('3')).toEqual(singlePilot);
    });

    // A test to verify that calling findPilotById() with an id that doesn't
exist, in this case 'AAA', returns undefined
    it('should return undefined if the pilot cannot be found', function() {
      expect(Pilotss.findPilotById('AAA')).not.toBeDefined();



    });
  });
```

2.6.2 Evaluation

The application was evaluated by defining fundamental tasks the application should always be able to perform.

I defined a few different fundamental tasks and performed those every time a big change was performed. The successful completion f each task gave me the assurance that the functionality of the application has not been affected.

Example of tasks:

- Create a new pilot
- Edit pilot details
- Create new event
- Edit event details
- See race results

These tasks aimed primarily to test the basic CRUD database operations as well as its implementation into the UI of the application.

3 Conclusions

While widely used, even by renowned companied like Microsoft, the usage of electron as a cross-platform framework for desktop application seems still very in its initial stages when it comes to be used in all kinds of applications.

IT does excel in certain types of applications, like code and text editors but it's underlying structure does fail to address easily applications with other requirements like the one I have built. More specifically when it needs to interact with elements outside of the application itself or hardware components, in the case of this application the serial ports and the database.

Node-webkit, the electron predecessor, has experienced several changes over the parts year/months, while still widely used it has lost some attention from the developers. So most of the content found only, like tutorials and examples, related to old version of it making it harder to find up to date content that demonstrates how to obtain certain functionalities.

Another issue found was that as it relies in NodeJS and in several other modules, the correct combination of versions causes incompatibilities making some times quite hard to correctly identify what is the problem.

As such this triggered my need to find a suitable solution and thus change to use electron. While with more recent content and information available is still no easy to find precise information on certain topics.

What I can call the quest for a framework to use in the application was quite challenging due the diversity available but in the end this research for a framework gave me very good insights on the variety and diversity of frameworks that exist for web applications, as well as Electron applications and I've gain valuable knowledge from it.

In here the videos tutorials found in Pluralsight were fundamental as they provided a step by step and progressive approach to the platform allowing me to increase my knowledge substantially when compared to either reading the Electron documentation itself or following other sources of information found in the internet. As well as provided insights on how to use simple JavaScript libraries, like AngularJS, and it's addons like UI.Router to achieve similar results as the one I'd get if I was using a UI framework.

One critical point identified is the usage of native modules, while the clear majority of modules that I use will be aimed at the web there is two so far that are native to

the O.S. Those are the serial interface and the database interface, these native modules need to compile against the version of electron that is running and this brings some problems. It means that upgrading to a newer version of electron newer native modules also need to be compiled and that this same upgrade to a newer electron version might not be immediately possible in case the modules were not updated to work with this latest version.

This should not affect the final distribution of the application, because if the two versions match it will not cause problems. Still I had to spend some time installing and configuring build environments for Linux, OSX and Windows to generate and test this connection between an electron version and the native modules used.

4 Further development

In the future, the further development of this system could pass by providing means to publish the data collected to a cloud based storage allowing to share more easily the data with websites or the services interested in reporting on it.

It can also incorporate with such system a leader board system allowing to create ranks of pilots listing their results over several events.

As these events tend to be transmitted online as a video stream, an option could be made to allow data to be presented as an overlay into those video streams allowing the viewers to receive in real-time information about the events, times and pilots, like what is seen currently in other motorsports.

There is still much room expansion in terms of adding features to this application and as users use it definitely more requests will be received and use cases will be identified.

5 References

Electron
https://electron.atom.io/

Spectron

https://electron.atom.io/spectron/

Boostrap
http://getbootstrap.com/

AngularJS — Superheroic JavaScript MVW Framework
https://angularjs.org/

Photon Kit
http://photonkit.com/

6 Appendix

6.1 Project Proposal

Project Proposal

FPV Race Management Application

BSHCESD4
Fernando Paulo Correia Morais
x12128058

BSc (Hons) in Computing

National College of Ireland
Oct 2016

Objectives

The main objective of my project is to develop one application used to manage FPV racing events.

This application will receive data mainly from lap tracking hardware devices and will be responsible to keep track of that information as well as provide reporting of times.

It is expected to allow to provide several features required to manage such events, like pilots management, class management and to manage the organization of the rounds and

Background

The reason for choosing to develop this application is due to the fact that is a area that is quite new and fast evolving.
The current existing solutions are quite simple and fail to provide most of the minimum requirements for it's usage.

Companies in the market are focusing more in the development of the hardware that technically is more complex to develop and only provide basic applications to interface that hardware.

So i aim to provide a system that will be able to connect to several types of hardware devices capable of tracking laps and have the option to not only manage the races but also produce reports of the results.

Technical Approach

The application ideally will be cross platform, allowing the user to use it independent of the operating system used. Linux, OSX or Windows.

In order to allow a easy to port and convert application that runs on all these operatings systems I have researched and found NodeJS as a good solution.

It has a good cross-platform compatibility, special at I/O access that for example with JAva is more complicated to achieve.

Special resources required

Will need to use some hardware for part of the test of the application.

This will be the lap tracking system, not all existing ones will be possible to use and for those software mimicking their behaviour will be used instead.

Technical Details

It will mainly be developed using:

Node.js

NW.js (node-webkit)

SQLite

It is expected that several modules from Node.js will be used, the exact list is not yet know as it will change according to the needs.

Evaluation

Part of the work of the application is to receive data from devices, so in that part I'll create simple software's that mimic their behaviour and the data they send allowing to easily test if the main application is receiving and processing correctly that information.

As for the second part that is the user interface, it will be a mix of direct user testing performed by me as well as automated browser/application testing executed as possible.

Fernando Paulo Correia Morais

_____

Signature of student

6.2 Monthly Journals

Reflective Journal

Student name: Fernando Paulo Correia Morais
Student ID: x12128058
Programme: BSHCESD4 - BSc (Honours) in Computing - Evening
Month: September

Looked at several ideas for the project and what software to use to develop it.

Searched for existing similar applications and determine how have they implemented certain features.

Decided to use Node.js and started reading some tutorials.

Reflective Journal

Student name: Fernando Paulo Correia Morais
Student ID: x12128058
Programme: BSHCESD4 - BSc (Honours) in Computing - Evening
Month: October

Started looking for package maker to facilitate the distribution of the final application.

Run few example tutorials and searched for frameworks to aid the development of the UI part.

Prepared the content to present at the project pitch.

Reflective Journal

Student name: Fernando Paulo Correia Morais
Student ID: x12128058
Programme: BSHCESD4 - BSc (Honours) in Computing - Evening
Month: November

Started working with compiling/generation of native modules, this is needed for the communication with serial ports.
Encountered multiple issues as these modules need to be natively compiled for

each system, so I may need to find a way to package them when distributing the final application files.

Created a test example to list, select and connect to a serial port using the UI.

Reflective Journal

Student name: Fernando Paulo Correia Morais
Student ID: x12128058
Programme: BSHCESD4 - BSc (Honours) in Computing - Evening
Month: December

Working and developed the base UI components to prepare for the presentation.

Added all the base layout elements, menus and areas. Integration with the databases added and pilot create/edit actions now functional.

Prepared and present the midpoint project presentation.

Reflective Journal

Student name: Fernando Paulo Correia Morais
Student ID: x12128058
Programme: BSHCESD4 - BSc (Honours) in Computing - Evening
Month: January

Decided to give one step back and get some better foundations and understating of how Electron and AngularJS work.

For that I found some good tutorial in Pluralsight. “Electron Fundamentals” by Jake Trent.

Following the entire tutorial and done all the test code and examples in it that allowed to get a much better understanding of the framework.

As well as made me realize my code was too messy and could be cleaned.

Reflective Journal

Student name: Fernando Paulo Correia Morais
Student ID: x12128058
Programme: BSHCESD4 - BSc (Honours) in Computing - Evening
Month: February

Continue my study on the electron framework by following the remaining video electron tutorials found in the Pluralsight platform.

These have been giving me very good insights into the framework mechanics as well as means to make clean simple code that works.

Started moving the code to use SQLite rather than MySQL, I want to have a local portable database rather than relying in an online database that was MySQL.

This will require some code changes as the methods used are not the same.

Met with the supervisor and we went over several improvements to be made to my ERD diagram.

Reflective Journal

Student name: Fernando Paulo Correia Morais
Student ID: x12128058
Programme: BSHCESD4 - BSc (Honours) in Computing - Evening
Month: March

Still working in the reporting part of the application.
Moving all database related operation to SQLite and trying to figure out the compilation of the SQLite3 native module for the version of electron I'm using.

Watched a few online videos on how to move electron from the web application feeling and into a more desktop application type that contained really good tips and tricks, like not allowing dragging visual elements something typical of web application code.

Met again with the supervisor that gave some more good insights into content to add to the project.

6.3 Project Plan

Gantt chart below detailed the predicted timeframe and steps for the project.