Trabajo de Fin de Grado

Autocheck for Github Classroom

Autocheck for Github Classroom .

Berkan Reyes Hernández

La Laguna, 2 de septiembre de 2018

D. **Casiano Rodríguez León**, con N.I.F. 42.020.072-S profesor Titular de Universidad adscrito al Departamento de Ingeniería y Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"*Autocheck for Github Classroom.*"

ha sido realizada bajo su dirección por D. **Berkan Reyes Hernández**, con N.I.F. 43.833.923-Vñ

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 2 de septiembre de 2018

Agradecimientos

Como no puede ser de otro modo, los agradecimiento van dirigidos en primer lugar hacia esas personas, cercanas como es obvio, que han puesto toda su confianza en mí: padres, abuelos, tíos, (familia en general) y amigos. Ellos han hecho posible que llegara hasta aquí, sorteando todos los obstáculos que se interponían con el curso de los años. Vivir alejado de los que uno quiere no es fácil y más aún en esta etapa tan importante. Aún así ellos han "estado sin estar" y por esto, y muchos más argumentos los cuales no son necesarios redactar, este grupo tan importante se lleva el agradecimiento principal.

Por otro lado debo agradecer a aquellas personas que de un modo o otro han contribuido en mi formación haciendo que hoy haya llegado hasta aquí. Profesores de la universidad que han realizado su trabajo correctamente, inculcando no solo conocimientos teóricos y prácticos sino también valores. Hacer mención especial a Casiano por su labor como tutor de este TFG, su atención y ayuda durante la realización del mismo han sido inmejorables ofreciendo su mano para cualquier problema.

Hacer referencia también a Moisés García Hernández, jefe de la empresa HGM Networks donde realicé las prácticas externas y donde se me trató de la mejor forma posible haciendo que esta primera experiencia laboral fuese satisfactoria.

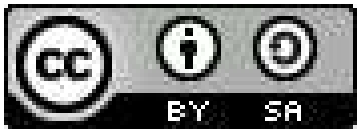
Por último agradecer también a parte del personal de la Universidad de La Laguna, concretamente a algunos que realizan su trabajo en la Residencia Escolar Santa María. Allí estuve durante los dos primeros años de andadura y mi buena relación con ellos hizo que me sintiera cómodo. Allí se forjaron las primeras amistades universitarias y no me arrepiento para nada de mi paso por ella a pesar de abandonarla en el tercer año.

Con esta memoria se pone fin a este recorrido de cinco años que deja atrás malos y muy buenos momentos, amigos, compañeros, etc.

Aquí comienza otro nuevo camino, igual o mejor que el anterior.

Muchas Gracias.

Licencia



Resumen

El objetivo de este Trabajo de Fin de Grado ha sido aplicar todos los conocimientos adquiridos a lo largo de estos años de enseñanza universitaria, en especial aquellos que se centran en el ámbito de las Tecnologías de la Información, itinerario elegido por el alumno.

Dicho TFG se basará en la creación de una aplicación de escritorio que permita y facilite la gestión de las ClassRooms de GitHub[15], convirtiéndose en una herramienta que aúne la plataforma y su herramienta para "classrooms", implementado mejoras que faciliten las labores de revisión de repositorios para una asignación determinada.

En ella se podrán crear asignaciones que muestren todos los repositorios que correspondan con la expresión regular introducida, de forma que para estos repositorios sea visible, mediante un badge, el estado de su integración contínua con travis además de enlaces para facilitar la labor de corrección de las tareas.

En esta memoria se describirán las diferentes fases de este proyecto con una breve introducción al ámbito, la forma en la que se ha trabajado y por último una pequeña guía de uso para que el usuario medio-bajo sepa como funcionan las diversas opciones que ofrece esta aplicación.

Palabras clave: Sistema de Control de Versiones, CVS, Git, GitHub, GitHub Classroom, GitHub Education.

Abstract

The objective of this Final Degree Project has been to apply all the knowledge acquired throughout these years of university education, especially those that focus on the field of Information Technology, itinerary chosen by the student.

This TFG will be based on the creation of a desktop application that allows and facilitates the management of the GitHub Classrooms, becoming a tool that unites Github and its tool for classrooms, implementing improvements that facilitate the revision tasks of repositories for a certain assignment.

In it you can create assignments that show all the repositories that correspond to the regular expression entered, so that for these repositories it is visible, through a badge, the status of its continuous integration with travis as well as links to facilitate the correction work of Tasks.

In this report we will describe the different phases of this project with a brief introduction to the field, the way in which we have worked and finally a small user guide so that the low-intermediate user knows how the various options offered by this application.

Keywords: *Control Version System, CVS, Git, GitHub, GitHub Classroom, GitHub Education.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Antecedentes y estado actual del arte

Con el paso del tiempo, el desarrollo software ha ido "in crescendo" de manera exponencial, pero esto es algo que ya se sabe, pues las tecnologías evolucionan y con ello todo lo que guarde algún tipo de relación con las mismas, y referente a este tema, se hace especial hincapié a la forma que se desarrolla un proyecto informático.

Cada vez más se usan los **CVS**, controladores de versiones, creados para el desarrollo de proyectos de software en los que intervienen numerosos programadores, como ocurre habitualmente en los proyectos de software libre. Un sistema de control de versiones es un software que administra los ficheros de código fuente, permite trabajar simultáneamente a varios usuarios y mantiene un historial de los cambios realizados.

Y si se nombra algún controlador de versiones es necesario hablar de Git[11], un sistema distribuido, en el que todos los nodos manejan la información en su totalidad y por lo tanto pueden actuar de cliente o servidor en cualquier momento, es decir, se elimina el concepto de "centralizado". Esto se lo logra gracias a que cada vez se sincronizan los cambios con el repositorio remoto, Git[11] guarda una copia entera de los datos con toda la estructura y los archivos necesarios. Así ya no es necesario salir a Internet para consultar los cambios históricos sobre un archivo o para ver quién fue la última persona que lo editó, todo se hace directamente sobre tu copia local y luego, cuando lo consideres oportuno, puedes enviar esos cambios hacia el repositorio remoto.

Para su gestión existe, además de muchas otras, la plataforma GitHub[15] donde se pueden alojar los proyectos en repositorios utilizando el conocido sistema de control de versiones. Con la creciente utilización de esta plataforma en el ámbito de la enseñanza, GitHub[15] ha creado herramientas para los desarrollos de los estudiantes y gestión de estos para los profesores.

Figura 1.1: Git + GitHub

El principal problema de esta herramienta a la que nos referimos, **GitHub Classroom**, es que para un profesor, las labores de corrección de tareas es muy lenta ya que éste debe entrar uno a uno a cada repositorio creado por la asignación para el alumno para su revisión de forma que el trabajo se convierte en algo muy costoso. Además, en la plataforma principal no se permite visualizar todos los repositorios para una asignación determinada teniéndose, de igual modo, que entrar a cada repositorio de forma individual.

1.2. Objetivos

En el resumen de esta memoria se menciona el objetivo principal de este proyecto, indicándose que el mismo consiste en aplicar los conocimientos adquiridos en el ámbito de las **Tecnologías de la Información** para la realización de un proyecto que cumpla los requisitos para un **TFG**.

Además, como se concluyó en el apartado anterior y volviendo a hacer hincapié, nace la necesidad de unir el entorno GitHub[15] con su herramienta para las "classrooms" siendo este el objetivo más concreto que identifica el proyecto en cuestión.

Este apartado será más explícito con esto y, en él, se ordenarán a continuación una serie de objetivos que en su totalidad se refieren al principal que debe ser alcanzado. Comenzado por el diseño, transcurriendo por el desarrollo y concluyendo con la finalización del mismo, los objetivos a resaltar son los que se muestran a continuación.

Como se aprecia en la tabla anterior, la mayoría de objetivos se refieren a la etapa de desarrollo ya que lo más importante de este proyecto software se refiere a la funcionalidad sin dejar de lado las buenas prácticas y un diseño bueno, claro e intuitivo.

Ámbito	**Objetivo**
Diseño	Realización del diseño prototipo
Desarrollo	Creación de la app de escritorio con un aspecto visual agradable e intuitivo
Desarrollo	Uso correcto de la API Octokit
Desarrollo	Funcionalidad: listar organizaciones
Desarrollo	Funcionalidad: creación y borrado de asignaciones
Desarrollo	Funcionalidad: visualización de repositorios para una asignación
Desarrollo	Funcionalidad: visualización de repositorios para una asignación
Desarrollo	Funcionalidad: visualización del estado en travis del repositorio
General	Uso de buenas prácticas

Tabla 1.1: Objetivos principales

1.3. Actividades a realizar

Una vez definidos los objetivos, en el apartado anterior, es necesario decir que actividades deben ser realizadas de manera correcta para la consecución del Trabajo de Fin de Grado propuesto por el alumno.

A continuación se muestran estas actividades en orden de ejecución:

- **Elección del nombre de la aplicación**
  - Creación del logo de la aplicación
  - Creación de un prototipo de la aplicación
- **Uso de Git[11] y la plataforma GitHub[15]**
  - Creación de un repositorio para albergar el código
  - Creación de un repositorio para albergar la memoria
- **Desarrollar una aplicación desktop con Electron[4]**
- **Uso de una API de GitHub, en este caso Octokit[8], para la gestión de la plataforma**
  - Autenticación en la plataforma mediante oauth2

  - Visualización de organizaciones para el usuario
  - Creación y eliminación de asignaciones para una organización
  - Visualización de repositorios para una asignación mediante regex
  - Badge indicador del estado del repositorio en Travis[5]
  - Enlaces a integración continua y repositorios
- **Realización de la memoria**
- **Realización de transparencias**
- **Realización de la defensa oral del proyecto**

Cumplir con esto objetivos supone la superación de la asignatura **TFG**, finalizando la misma con la exposición oral del proyecto frente al tribunal de evaluación que ha sido asignado.

Capítulo 2

Desarrollo

En el capítulo anterior se ha definido y explicado de manera simple y sencilla el tema que abordaba este proyecto, comentando y situando el mismo en cuanto a antecedentes y actualidad se refiere y haciendo una análisis de los objetivos y tareas para superar los mismos que el alumno propone, pudiendo ser alguno de estos una línea futura de trabajo para un proyecto de mayor alcance.

A continuación se detallará como se ha procedido en el desarrollo de la aplicación, comentado el método de trabajo utilizado de forma extendida y las tecnologías usadas de manera detallada. Por último se hablará de la plataforma GitHub[15] que tan importante es en esta aplicación.

2.1. Tecnologías y herramientas usadas

La aplicación, de la que se hablará más detenidamente en el capítulo 3 de esta memoria, ha sido desarrollada con la ayuda del el framework Electron[4]. Haciendo así que se trate de una app de escritorio que puede ser instalada, como cualquier otra de uso común como puede ser un navegador, un editor de texto, etc.

Electron[4], la tecnología principal que engloba el resto, se trata de un framework/plataforma usado para desarrollar aplicaciones de escritorio con tecnologías web (HTML[3], CSS[6] y Javascript[13]) creada y mantenida por GitHub[15]. Electron.js funciona creando dos tipos de procesos, el proceso **main** y el proceso **renderer**. El primero es un proceso de Node.js[9], como la propia palabra indica, el proceso principal, lo cual viene a ser aplicación en sí misma. Este proceso tiene acceso a varias API de Electron.js solo para este proceso que nos ayudan a comunicarnos con el SO y realizar distintas acciones o efectos.

El segundo, renderer, es un proceso de **Chromium**, con una diferencia con respecto al original, este tiene un Node.js[9] incorporado y acceso a todos sus módulos y los que instalemos con npm (esto permitiría usar **React.js**, **Angular.js**, **Polymer**, etc. para desarrollar la UI o cualquier otra librería), por

lo que desde el proceso renderer se pueden usar módulos como fs para leer y escribir en el disco, o hacer peticiones a una base de datos directamente.

Además de acceder a los módulos de Node.js[15] y NPM[16], Electron.js nos da acceso a una pocas API en este proceso igual que hace con el proceso main.

Figura 2.1: Logo Electron

Decir además, que esta tecnología es la usada para la creación del conocido editor **Atom**, de GitHub[15].

Como se ha dicho, este framework requiere el uso de **tecnologías web** ya que son las que soporta **Chromium** , así que estarán presente aquí HTML[3], CSS[6]. El primero, el conocido lenguaje de marcado de hipertexto, será el encargado de crear la estructura básica de la aplicación, la cual tomará estilo y diseño gracias a las hojas de estilo CSS que darán presentación y formato a la estructura previamente creada.

Figura 2.2: Logo HTML y CSS

Otra de las tecnologías que incluye Electron es NPM[16], el gestor de paquetes javascript de Node.js[9] por excelencia. Gracias a él, se obtienen las distintas librerías/módulos necesarios para el desarrollo de este proyecto.

Este gestor de paquetes utiliza el archivo **"package.json"**, un fichero de extensión .json que, además de incluir datos del respositorio, almacena en él las dependencias y tareas necesarias para llevar a cabo un proyecto de estas características y será aquí donde se incluya Electron[4] para su uso. Además de este, otros muchos módulos serán incluidos, siendo muchos de estos explicados en el punto tres de la memoria de esta **Memoria de Trabajo Fin de Grado**.

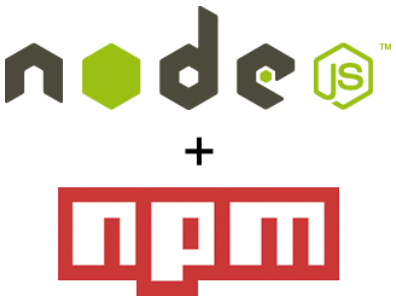


Figura 2.3: Logo NodeJS y NPM

2.2. Metodología de Trabajo

Con metodología de trabajo se refiere a la forma en la que se ha procedido para desarrollar el proyecto, los tiempos de ejecución, las herramientas utilizadas. En conclusión, y dicho claramente: '**Cómo**, **cuándo** y **con qué** se hizo la aplicación'.

De forma general y englobando la metodología en una de las ya definidas, se puede decir que la elegida y utilizada ha sido **XP** , eXtreme Programming[2].

XP es una de las metodologías más utilizadas en cuanto a desarrollo de software se refiere. Esta engloba distintos aspectos que serán descritos a continuación y relacionados con el proyecto en cuestión.

Esta metodología es una de las más destacadas en el marco de los **procesos ágiles de desarrollo** y se diferencia del resto en que pone más énfasis en la adaptabilidad que la previsibilidad, esto es, valorar más la facilidad para afrontar posibles cambios en el rumbo de la aplicación que la fijación de un plan cerrado.

Como cualquier otra, su objetivo principal es el de mejorar la productividad de cualquier proyecto en el cual se haga uso de la misma. Para ello se basa en diversos principios que se relacionarán directamente con el proyecto in situ :

- **Integración Continua**

Las distintas funcionalidades de la aplicación se realizan una tras otra de manera progresiva, reconstruyendo el proyecto varias veces al día si es necesario en vez de crear versiones estables. Por lo tanto se realizan pequeñas entregas tal y como alumno y director de este proyecto han hecho, pues el primero entregaba las funcionalidades una vez hechas para pasar a las siguientes, realizando cambios continuos si así era necesario. Además se ha hecho real hincapié en la refactorización, con revisiones continuas del código para asegurar que este fuese totalmente óptimo.

- **Calidad de trabajo**

Es importante que el programador se encuentre en las condiciones idóneas para realizar el trabajo por lo que es importante dividirlo de manera correcta y no trabajar más del tiempo marcado. Así ha sido durante todo el proyecto, el trabajo se ha realizado poco a poco y cuidadosamente, trabajando a buen ritmo y sin hacer que el desarrollo se volviese cansino.

- **Simplicidad**

Cuanto más simple sea la aplicación, siempre y cuando cumpla las necesidades del cliente, mejor. El proyecto se ha ajustado a lo que pedía el que podría ser un cliente potencial como es el director del mismo, manteniendo los estándares de programación y siendo lo más simple, usable y accesible posible.

- **Retroalimentación**

Como se ha sido anteriormente, el director ha ejercido de cliente, estableciendo las funcionalidades que se requerían y haciendo también de tester para obtener resultados y aplicar cambios a partir de estos. Además ha planificado las tareas y ha recibido contínuo feedback a través de los distintos medios por los que se estableció comunicación.

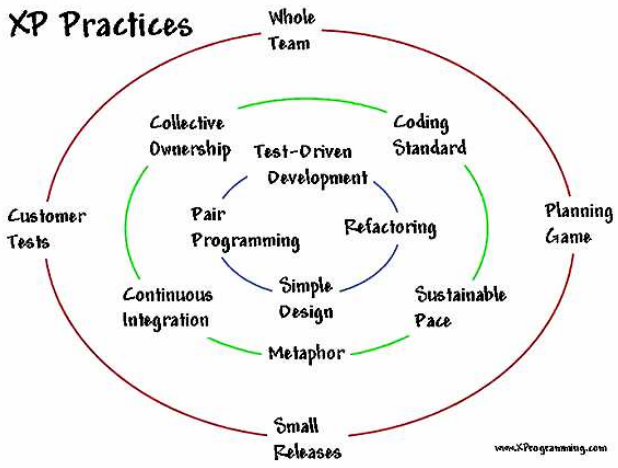


Figura 2.4: Características Extreme Programming

Para aplicar esta metodología, como cualquier otra, son necesarias una serie de herramientas que se describirán a continuación, detallando de manera más profunda una de ella, Github[15].

2.3. Herramientas utilizadas

Entre las herramientas utilizadas destacan los medios físicos, los medios de comunicación, el editor de código y el resto de utensilios necesarios para realización de un proyecto de software siguiendo una metodología de trabajo ágil.

Cuando se habla de medios físicos se refiere a los dispositivos utilizados para el desarrollo así con los lugares de reunión que también son muy importantes ya que debe ser un sitio donde se pueda reunir el equipo en unas buenas condiciones para comentar el trabajo realizado y el porvenir del proyecto.

Figura 2.5: Aplicación Multiplataforma

Por un lado, se han utilizados dos ordenadores, un computador de sobremesa con sistema operativo **Windows 10** y otro, portátil, con **macOS High Sierra**. Esto ha sido necesario para los testeos en diversos sistemas ya que se trata de una aplicación **multiplataforma**, siendo estas dos las que han sido probadas y comprobadas su funcionamiento.

Por otro lado, el lugar donde se han sucedido las reuniones ha sido el despacho del director del proyecto, siendo estas repetidas a lo largo del presente curso y más habituales durante el segundo trimestre del año.

Para fijar estas se ha hecho uso de la aplicación de **correo electrónico** del dispositivo móvil o computadora. A través del correo institucional y cliente utilizado, alumno y director acordaron fecha y hora para verse y comentar aspectos relacionados con el proyecto. En estas reuniones, no tan formales siempre, se trataban todos los puntos de la aplicación los cuales eran anotados para mantener un historial de encuentros.

Por último, en cuanto a herramientas informática se refiere, se ha utilizado el editor de código **Atom**, ya que ofrece sencillez, una buena integración con GitHub[15] y otras herramientas complementarias, además de un buen aspecto y utilidad cuando se refiere a proyectos de no tan largo alcance.

A continuación se dedicará un apartado para la plataforma con control de versiones ya que ha sido uno de los puntos principales a lo largo de todo el **prototipado**, **diseño** y **desarrollo** de la app.

2.4. Github

La plataforma del **"octocat"**, mejor conocida como Github[15], se trata de un sitio web con servicio en la nube que sirve a los desarrolladores un espacio donde almacenar su código y llevar un control y administrar cualquier cambio sobre el mismo gracias al CVS Git[11].

Esta, además de ofrecer almacenamiento también incluye muchas otras herramientas que permiten desarrollar un proyecto software con total **integración**, **feedback**, **cooperación**, **organización**, etc. De ahí que haya sido tan importante para esta aplicación.

Desde un principio se comenzó creando un repositorio, submódulo del principal de esta aplicación, que contiene el código del mismo. En este se llevaba, como se ha dicho, gracias a las "github tools", el seguimiento e integración. En la plataforma, el director, Casiano, notificaría mediante **"issues"** las distintas correcciones que debían hacerse sobre el código o de igual modo, comunicar o sugerir las diversas funcionalidades que ha acabado englobando el proyecto.

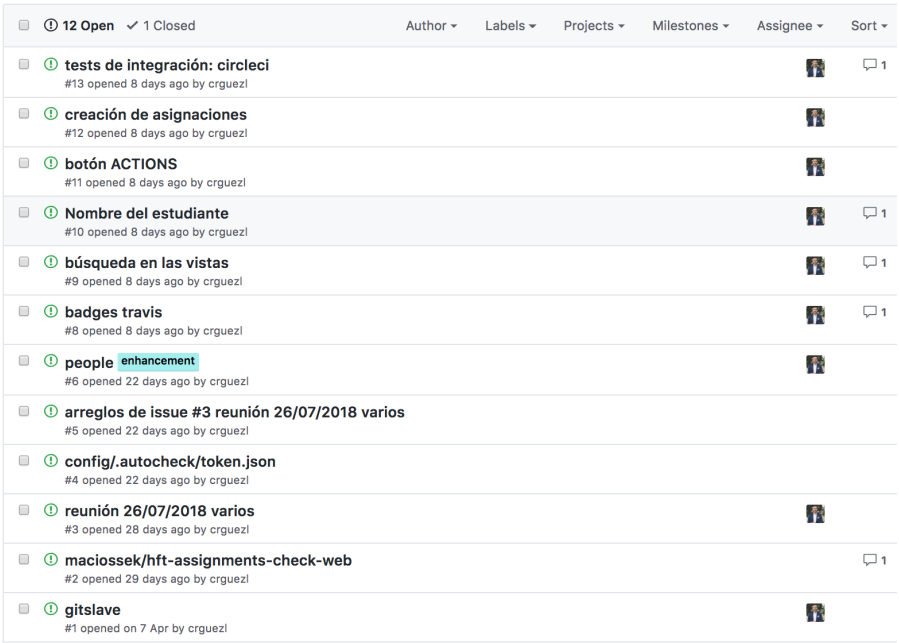


Figura 2.6: Github Issues

Además también se ha hecho uso de los **"projects"** de repositorio de Github[15] para organizar el trabajo aunque en menor grado que la utilidad **"issues"**.

Por otro lado, Github ha servido para mantener un control de versiones del código, de forma que se pudiese volver atrás en cualquier momento para adaptarse de manera inmediata a lo que se pedía. Los **"commits"**, con un número cercano a 80 han seguido un estándar, comenzando estos siempre por verbo en

tercera persona, hechos después de la inclusión de algún cambio notorio y en inglés.

Para concluir con este apartado, decir que la aplicación necesita de esta plataforma para su autenticación mediante **OAuth** y múltiples permisos para tener acceso a las funcionalidades que implementa la app y las cuales necesitan de la **API de Github** Octokit[8] que será la encargada de traer los datos necesarios desde la plataforma.

Capítulo 3

Autocheck

Autocheck es una aplicación de escritorio hecha con Electron[4] que sirve como herramienta complementaria para aquellos que utilizan **GitHub Classroom**. Esta ofrece un entorno para crear y simular las asignaciones de una organización pudiéndose ver los repositorios de cada una de ellas y su integración continua en Travis[5].

Como se ha dicho en reiteradas ocasiones este proyecto ha sido desarrollado con el framework de la "partícula negativa" y se diferencia del resto en que ofrece un diseño muy sencillo, con funcionalidades innovadoras y con un código totalmente **modular**.

3.1. Prototipo

Tras un completo estudio del ámbito y el desarrollo del proyecto, se comenzó con el desarrollo de una aplicación prototipada en la cual se establecieron las diversas pantallas que esta debía contener así como el diseño de forma muy general: navbar en la parte superior, breadcrumbs, paneles de acciones, etc. Además también se optó por una gama de colores y por un primera aproximación del logo de la misma.

Este prototipo finalmente se ha acercado mucho al diseño final, distando solamente en la adición de alguna vista más y una mejora del logo para hacerlo más estético y **elegante**.

3.2. Logo, apariencia y diseño

El logo principal de la aplicación guarda un diseño elegante, sencillo, con una caligrafía clara, en negro, y sólo con una letra distintiva a la cual se la ha dado un diseño moderno y se le ha aplicado también el color verde. Dicha letra es la primera del nombre **"Autocheck"**. Ésta, en un tono verde, oculta el símbolo

de 'correcto' (check), que hace que se forme la vocal de manera clara y limpia, sin llevar a dudas.

La elección de este color es simple, ya que el verde es un color que se le atribuye normalmente al símbolo anteriormente nombrado y encaja perfectamente con el significado del nombre y la funcionalidad de la aplicación. Decir, para finalizar con el logo, que la letra "A" será el logo miniatura y el usado también como icono de aplicación para macOS.

Figura 3.1: Autocheck Logo. Letra "A"

En cuanto al resto de la web, salvo la pantalla de inicio de sesión, el resto es una especie de **dashboard** dividido en dos partes: Una primera parte de color **"negro GitHub"**, para que guarde relación con la plataforma, que contendrá la **barra de navegación de la app**, donde se indicará la vista en la que se encuentra el usuario, **breadcrumbs** para la navegación y un botón para acceder a nuestro perfil y otro para cerrar la sesión.

La segunda parte contendrá la distinta funcionalidad dependiendo de la vista la que se encuentre en el usuario en cada instante, teniendo siempre un color de fondo verde, del mismo todo que tiene la plataforma **GitHub Classroom**, y paneles blancos para que resalte la información que se muestra.

En la guía de uso, último apartado de este proyecto, se verá a través de capturas el aspecto final de las distintas vistas así como una explicación de que se puede hacer en cada de uno de ellas y para qué han sido creadas.

3.3. Funcionalidades

Uno de los apartados más esenciales de la memoria es este que se presenta aquí, pues es en él se explica cómo se han conseguido los objetivos a través de

las distintas funcionalidades, viniendo a decir de qué es capaz Autocheck.

Antes de empezar, decir que la aplicación se ha valido del sistema de vistas EJS[7] para la visualización de las distintas pantallas, siendo esto algo poco común pero que ha sido elegido para una mejor estructuración del código y siendo también necesario la utilización del módulo "electron-view-renderer" para la renderización de la vista y la facilitación de las labores referidas al paso de variables desde el servidor hasta el cliente.

Además se ha seguido una estructura de directorios usual, común en otro tipo de frameworks como Laravel[14] en el caso de PHP[10].

- **main.js**

Este archivo es el principal de la aplicación. Es el **servidor** en sí. Configura Electron[4] y se encarga de servir todas las peticiones al **cliente** y llevar la aplicación tal y como se desea. Incluye los diversos módulos, una clase para el manejo de Octokit[8], la configuración de la ventana y el menú superior y las respuestas a las peticiones del proceso renderer.

- **config/electron/**

En este directorio se encuentra el archivo de configuración OAuth de la aplicación con el **"clientid"** , el **"secret"** y otros parámetros necesarios para que la aplicación funcione con el OAuth de GitHub[15]. Además también se encuentra el archivo **"menu.js"** que incluye la configuración de las opciones del menú superior para la aplicación dependiendo del SO operativo en el que se ejecute.

- **app/**

En este otro directorio se encuentra un archivo/clase javascript con los métodos necesarios para obtener las organizaciones, perfil de usuario, repositorios, etc. para un usuario que ha hecho autenticado.

- **resources/**

En esta ultima carpeta se encuentra todo lo relacionado con el cliente desde las diferentes vistas con los **"layouts"** para el **"footer"**, **"header"**, y la estructura que es identica en la mayoría de pantallas hasta los **"assets"** donde se encuentran las hojas de estilo para el diseño de la aplicación y el directorio **"js/"** donde se encuentran archivos para modificar la funcionalida de la página y hacer las peticiones al proceso "main" de **Autocheck**

Por último, respecto a el código de la aplicación, este cumple con la guía de estilo Standard Javascript[1] por lo que todo se encuentra perfectamente en **orden**. Además cada una de las funcionalidad ha sido comentada en **lengua inglesa** ya que se encuentra más extendida y hay un mayor número de usuarios que entienden este idioma.

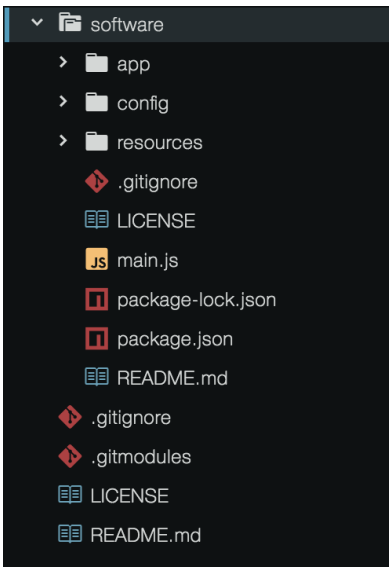
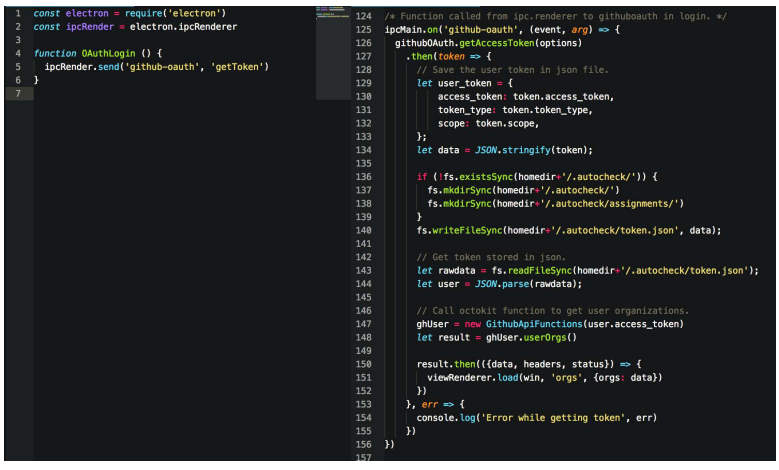


Figura 3.2: Estructura de directorios

3.3.1. Autenticación

El primer aspecto a tratar es la **autenticación**. Esta se realiza nada más ejecutar la aplicación y es totalmente necesaria para poder acceder al resto de funcionalidades.

Dicha autenticación se vale de un **"login"** en la plataforma GitHub[15] gracias al módulo de Node.js[9] para Electron[4] **"electron-oauth2"**[12]. Este módulo implementa un acceso a la **API de GitHub** para realizar la autenticación.



```
const electron = require('electron')
const ipcRender = electron.ipcRenderer

function OAuthLogin () {
  ipcRender.send('github-oauth', 'getToken')
}
```

```
/* Function called from ipc.renderer to githuboauth in login. */
ipcMain.on('github-oauth', (event, arg) => {
  githubOAuth.getAccessToken(options)
    .then(token => {
      // Save the user token in json file.
      let user_token = {
          access_token: token.access_token,
          token_type: token.token_type,
          scope: token.scope,
      };
      let data = JSON.stringify(token);

      if (!fs.existsSync(homedir+'/.autocheck/')) {
        fs.mkdirSync(homedir+'/.autocheck/')
        fs.mkdirSync(homedir+'/.autocheck/assignments/')
      }
      fs.writeFileSync(homedir+'/.autocheck/token.json', data);

      // Get token stored in json.
      let rawdata = fs.readFileSync(homedir+'/.autocheck/token.json');
      let user = JSON.parse(rawdata);

      // Call octokit function to get user organizations.
      ghUser = new GithubApiFunctions(user.access_token)
      let result = ghUser.userOrgs()

      result.then(({data, headers, status}) => {
        viewRenderer.load(win, 'orgs', {orgs: data})
      })
    }, err => {
      console.log('Error while getting token', err)
    })
})
```

Figura 3.3: Autenticación

Para llamar a esta función, desde el **renderer** (cliente), al hacer click sobre botón de inicio de sesión, se llama a una función incluida en uno de los ficheros **".js"** que llama al proceso **main** con un parámetro para obtener el **token**. En

este segundo se realiza el **"sign in"** gracias al módulo anteriormente comentado, y tras obtener el token se crea un archivo ".json" donde se guardará el mismo para futuros inicios de sesión.

Resaltar que para iniciar sesión y disfrutar de las funcionalidades es necesario dar permiso a la aplicación para que puedo ejecutar acciones sobre los repositorios y organizaciones.

Por último, tras almacenar los datos de inicio de sesión, se hará una llamada a la función para obtener las organizaciones del usuario logeado y renderizar la siguiente vista.

3.3.2. Visualización de organizaciones

Esta es la funcionalidad principal de la que parten el resto ya que los parámetros se van pasando de vista en vista simultáneamente, teniéndose siempre acceso al nombre de la organización para la que se ejecutan acciones.

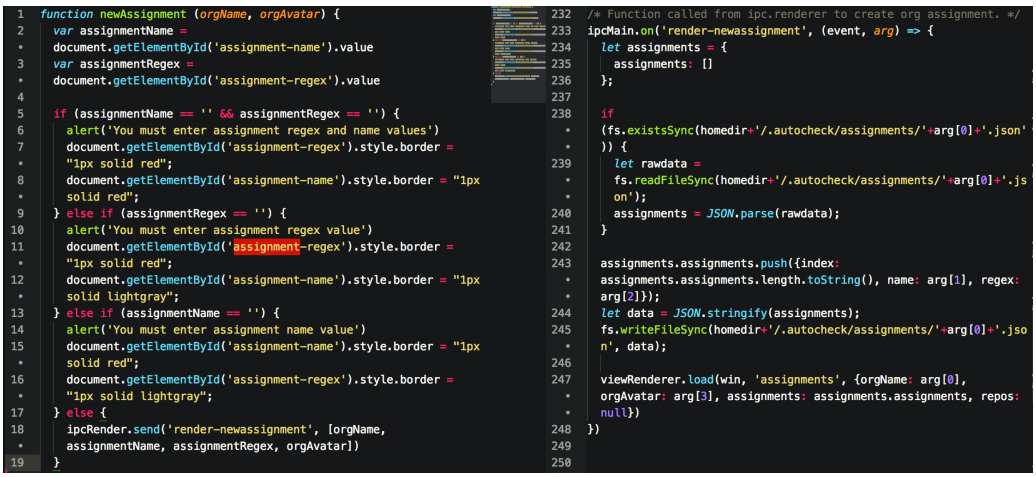
Las **organizaciones** son obtenidas justo en el instante en el que el usuario se autentica en la plataforma. Esto es gracias a una función de Octokit[8], un módulo de Node.js[9], que no es más que una **API oficial de GitHub**, que ofrece multitud de funciones para obtener todo tipo de datos de la plataforma.

Una vez obtenidas las organizaciones se utiliza el módulo comentado anteriormente para renderizar la vista y pasar como parámetro las organizaciones del usuario. Esta vez el cliente, mediante EJS[7] se muestran cada una de estas con la opción de clickear encima de ellas para pasar a la siguiente pantalla.

3.3.3. Creación y eliminación de asignaciones

Esta funcionalidad es simple. Después de llegar a la vista necesaria para realizar la creación o eliminación de **asignaciones**, con la organización para la que se van a crear estas, se debe rellenar una especie de formulario con la expresión regular para filtrar los repositorios y el nombre que se le quiere dar a la asignación.

Estos datos son enviados al **servidor** el cual creará un archivo ".json" en el directorio local del usuario con el nombre de la organización para la que se ha creado la asignación más el nombre indicado anteriormente. Este archivo contendrá todas las asignaciones que se deseen crear, con nombre y expresión regular, para que así cuando se acceda a una organización se puedan listar las assigments que ya han sido creadas anteriormente pudiendo ser así también borradas simplemente clickeando y mandando la orden al servidor para que éste borre la entrada señalada en el JSON.



```
function newAssignment (orgName, orgAvatar) {
  var assignmentName =
  document.getElementById('assignment-name').value
  var assignmentRegex =
  document.getElementById('assignment-regex').value

  if (assignmentName == '' && assignmentRegex == '') {
    alert('You must enter assignment regex and name values')
    document.getElementById('assignment-regex').style.border =
    "1px solid red";
    document.getElementById('assignment-name').style.border = "1px
    solid red";
  } else if (assignmentRegex == '') {
    alert('You must enter assignment regex value')
    document.getElementById('assignment-regex').style.border =
    "1px solid red";
    document.getElementById('assignment-name').style.border = "1px
    solid lightgray";
  } else if (assignmentName == '') {
    alert('You must enter assignment name value')
    document.getElementById('assignment-name').style.border = "1px
    solid red";
    document.getElementById('assignment-regex').style.border =
    "1px solid lightgray";
  } else {
    ipcRender.send('render-newassignment', [orgName,
    assignmentName, assignmentRegex, orgAvatar])
  }
```

```
/* Function called from ipc.renderer to create org assignment. */
ipcMain.on('render-newassignment', (event, arg) => {
  let assignments = {
    assignments: []
  };

  if
  (fs.existsSync(homedir+'/.autocheck/assignments/'+arg[0]+'.json'
  )) {
    let rawdata =
    fs.readFileSync(homedir+'/.autocheck/assignments/'+arg[0]+'.js
    on');
    assignments = JSON.parse(rawdata);
  }

  assignments.assignments.push({index:
  assignments.assignments.length.toString(), name: arg[1], regex:
  arg[2]});
  let data = JSON.stringify(assignments);
  fs.writeFileSync(homedir+'/.autocheck/assignments/'+arg[0]+'.jso
  n', data);

  viewRenderer.load(win, 'assignments', {orgName: arg[0],
  orgAvatar: arg[3], assignments: assignments.assignments, repos:
  null})
})
```

Figura 3.4: Creación de una assignment

3.3.4. Visualización de repositorios de una asignación

La funcionalidad que mayor complejidad ofrece es la siguiente. Tras clickear sobre una asignación, en el servidor se obtienen los datos, desde el "fichero.json" anteriormente creado, de la misma y con el nombre de la organización y la **API de GitHub** Octokit[8] se obtienen los repositorios.

```
/* Function called from ipc.renderer to render assignment repos. */
ipcMain.on('render-assignment-repos', (event, arg) => {
  let rawdata = fs.readFileSync(homedir+'/.autocheck/token.json');
  let user = JSON.parse(rawdata);

  ghUser = new GithubApiFunctions(user.access_token)
  let result = ghUser.paginate(arg[2])

  .then(data => {
    let repos = {
      repos: []
    };

    // Filter repos by RegExp
    let regex = new RegExp(arg[1], 'g');

    for (var i = 0; i < Object.keys(data).length; i++) {
      var orgrepos_filter = data[i].name.match(regex);
      if (orgrepos_filter != null) {
        repos.repos.push(data[i])
      }
    }

    viewRenderer.load(win, 'assignmentrepos', {orgName: arg[2], orgAvatar:
    arg[3], assignmentName: arg[0], assignmentRegex: arg[1], repos: repos.repos})
  })
})
```

Figura 3.5: Renderizando repositorios

Una vez obtenidos los **repositorios**, con la expresión regular de la asignación se filtran el listado obtenido y se crea un objeto con los repositorios filtrados. De este modo se pueden enviar al cliente cuando se renderiza la vista en la que se refleja esta funcionalidad.

3.3.5. Badges con el estado de la integración continua

Esta última funcionalidad es quizás la "más fácil" de implementar. Una vez obtenidos los repositorios, con el nombre de cada uno de ellos y el nombre de la organización dueña de la asignación en la que se encuentra, se construye la url de forma manual poniendo esta como "src" de una imagen para que automáticamente cargue el **badge** de Travis[5] con el estado del repositorio en la plataforma.

Figura 3.6: Travis Badges

Capítulo 4

Conclusiones y Líneas Futuras

Como se dijo en la introducción, los controladores de versiones (CVS) y las plataformas que hacen uso de ello han crecido de manera exponencial debido a las facilidades que ofrecen a los usuarios.

Por esto ha sido necesario el desarrollo de herramientas complementarias que ayuden y hagan aún más sencillo el desarrollo de software. Y con este fin se ha creado **"Autocheck for Github Classroom"**, nombre que ha recibido la aplicación tras diferentes pruebas y análisis. Este nombre resume para lo que ha sido creada esta app que aúna la plataforma Github, con las Classrooms y con la integración contínua. Volviendo una vez más a su nombre, la aplicación intentará "automatizar" de manera visual el estado de las pruebas puestas para los distintos repositorios de una asignación siendo así más fácil su corrección en vez de acceder uno a uno.

En conclusión, con el desarrollo de esta aplicación, que podría ser usada por profesores que evalúen las tareas de sus alumnos mediantes repositorios e integración continua, se pretende que sea una herramienta de utilidad, por ello es que se ha cuidado con mimo su **interfaz**, **funcionalidad** y **usabilidad** aunque esté dirigida a un usuario con conocimientos medio-alto.

En términos generales se han conseguido los objetivos principales después de mucho esfuerzo y dedicación al estudio pero, como es obvio, se han planteado muchas ideas que hubiesen sido muy útiles para aquellas personas que harán uso de **Autocheck** no siendo posible su realización por falta de tiempo y una mejor documentación de apoyo en el desarrollo. Por ello se ha decidido optar por una versión estable que ofrezca funcionalidades básicas con un diseño **robusto** y con posibilidad de **escalar** y añadir nuevas funcionalidades de manera poco costosa.

En el futuro existe la intención de continuar ampliando esta aplicación con nuevas e innovadoras funcionalidades ya que, como se ha comenzado diciendo,

las herramientas para Github no paran de crecer y ésta podría ser una buena forma de adentrarse en el mundo del desarrollo **full stack** por la puerta grande.

Figura 4.1: Autocheck Logo

Capítulo 5

Summary and Conclusions

As we started by saying in the introduction, the version controller systems and the platforms that make use of it have grown exponentially due to the facilities they offer to the users.

This is why it has been necessary to develop complementary tools that help and make the development of software even easier. And for this purpose .ªutocheck for Github Classrooms"has been created, name that the application has received after different tests and analysis. This name summarizes for what this app has been created that unites the Github platform, GH Classrooms and the continuous integration. Going back once more to its name, the application will try to .ªutomate"in a visual way the status of the tests put for the different repositories of an assignment, thus being easier to correct them instead of accessing them one by one.

In conclusion, with the development of this application, which could be used by professors who evaluate the tasks of their students through repositories and continuous integration, it is intended to be a useful tool, which is why we have taken care of its interface, functionality and usability even if it is aimed at a medium-high user.

In general terms, the main objectives have been achieved after a lot of effort and dedication to the study, but, as is obvious, many ideas have been raised that would have been very useful for those people who will use Autocheck, not being possible to achieve due to lack of time and better support documentation in the development. For this reason it has been decided to opt for a stable version that offers basic functionalities with a robust design and with the possibility of scaling and adding new functionalities in an inexpensive way.

In the future, there is an intention to continue expanding this application with new and innovative functionalities since, as has been said, the tools for Github do not stop growing and this could be a good way to enter the world of full stack development making a grand entrance.

Capítulo 6

Presupuesto

En este capítulo se hará un desglose del presupuesto necesario para la realización del proyecto **"Autocheck for Github Classroom"** . De este modo se conocerán bien los activos necesarios así como lo que podría suponer el desarrollo de un proyecto de este tipo.

Muchos de los activos aquí citados ya han sido explicados en otros puntos de la memoria y se conoce su uso en el proyecto por lo que será más fácil el entendimiento de su valor en el presupuesto. Decir que esta es una buena aproximación tanto a nivel material como económico, de lo que supone un desarrollo fullstack de una aplicación de escritorio.

Los dispositivos hardware son valores reales sacados de la web oficial de los proveedores de un ordenador con sistema operativo **Windows** y otro **macOS**, mientras que para el software muchas veces no es necesario ningún tipo de inversión, o por lo menos en este caso no lo ha sido, pero en la mayoría de proyectos suele ser necesario el alojamiento del servidor, módulos más complejos, etc. y esto si que requiere un coste.

Activo	**Cantidad**	**Precio Unitario**	**Precio Total**
Hardware			
Ordenador: SO Windows10	1	659,99€	659,99€
Ordenador: SO macOS High Sierra	1	1105,59€	1105,59€
Herramientas Software			
Cliente de correo	1	0,00€	0,00€
Atom	1	0,00€	0,00€
Github	1	0,00€	0,00€
Módulo NodeJS	8	0,00€	0,00€
Mano de obra			
Estudio de mercado	48 horas	15,00€	720,00€
Diseño del prototipo	48 horas	20,00€	960,00€
Creación definitiva del diseño	72 horas	20,00€	1440,00€
Desarrollo Software	185 horas	26,00€	4810,00€
PRESUPUESTO TOTAL - 9585,58€			

Tabla 6.1: Presupuesto Autocheck

Capítulo 7

Guía de uso

Para finalizar la memoria, en este último punto se mostrarán las diversas pantallas de las que consta la aplicación, explicando las distintas funcionalidades visibles haciendo una pequeña guía de usuario a través de **"Autocheck"**.

A través de esta guía se pretende que cualquier usuario pueda utilizar la app teniendo una idea clara de lo que puede hacer, dónde y accediendo a qué lugar, facilitándole una vez más la comprensión de la misma.

7.1. Pantalla Sign In

La primera pantalla con la que se encuentra al iniciar la aplicación es la vista de **"Login"**. En esta pantalla se cargará de forma dinámica el logo de la aplicación de manera que va aumentando su tamaño viniendo hacia el frente. Incluye, como se ha dicho, simplemente el logo y un botón de color negro para iniciar sesión en la aplicación.

Al hacer "click" sobre este botón vendrá al centro una ventana que nos indicará que se debe iniciar sesión en con el usuario de GitHub[15] y dar acceso a la aplicación para poder ejecutar acciones sobre las **organizaciones** y los **repositorios**. Tras esto el usuario se encuentra en disposición de poder acceder a las distintas funcionalidades que **"Autocheck"** ofrece.

7.2. Pantalla Profile

Esta pantalla es meramente visual y muestra información de la plataforma GitHub[15] del usuario que se encuentra autenticado.

En ella se puede ver una foto en circular del usuario, el correo electrónico, su localidad . Igual que el resto de las vistas que se comentarán a continuación, esta muestra el aspecto general con **menú de navegación** negro y **dashboard** verde con **paneles** blancos.

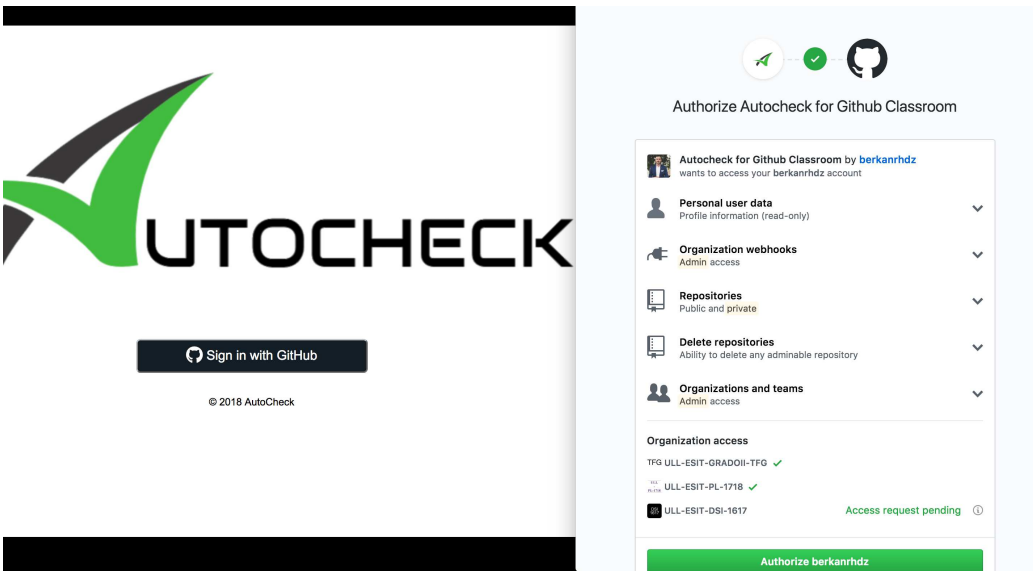


Figura 7.1: Autenticación Autocheck

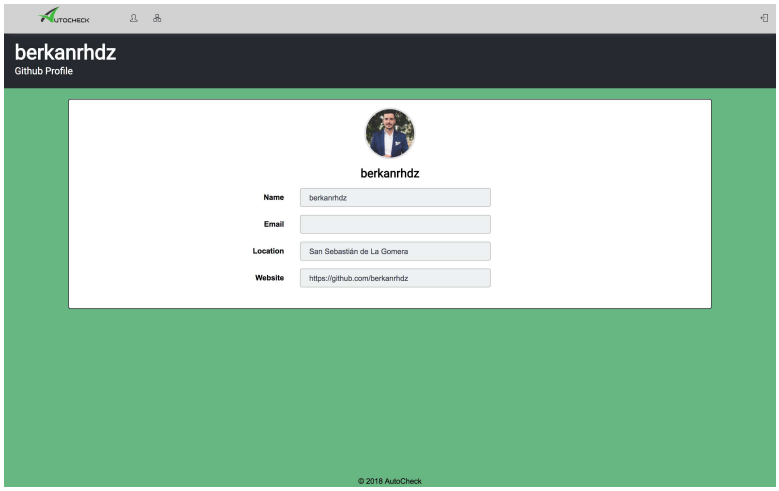


Figura 7.2: Perfil de Usuario

Decir que el botón que aparece siempre en el **"topbar"** con el icono de un usuario es el que permite acceder a esta vista. Justo al lado aparece el botón que da acceso a la ventana de organizaciones, y más a la derecha un botón para hacer **"logout"** (cerrar sesión).

7.3. Pantalla Organizations

La pantalla **Organizations** es la principal, renderizada justo después de inciar sesión y a la cual se puede acceder, como se dijo en el anterior apartado, con el botón situado en la parte superior. Esta, además, se cargará de forma automática cuando el usuario ya se haya autenticado alguna vez, de forma que

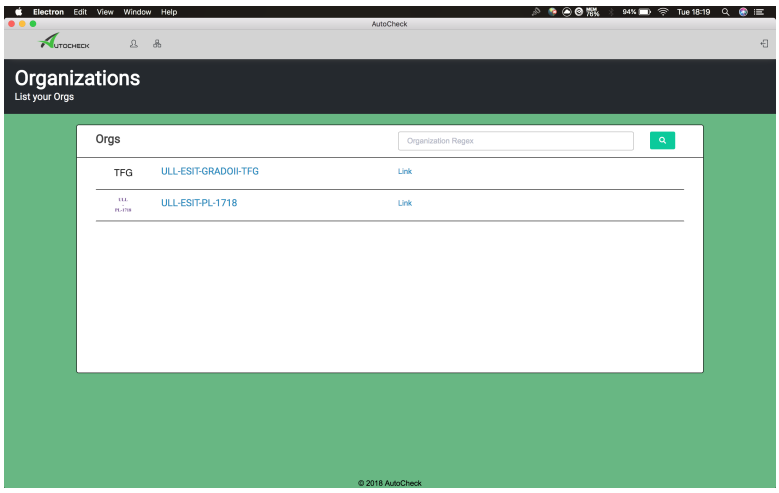


Figura 7.3: Pantalla Organizations

este no tenga que iniciar sesión repetidamente cada vez que ejecuta la aplicación.

En todas las vistas se mantiene la estructura, cambiando el título de la vista en la barra de navegación y el contenido de los paneles blancos que contienen la funcionalidad de la pantalla.

En este caso, el panel contiene una especie de tabla con una única columna, donde cada fila corresponde a una organización de GitHub[15] del usuario, con su imagen y un link que lleva directamente al sitio web de esta.

Sobre el nombre de las organizaciones se puede hacer "click" y de este modo se pasaría a la siguiente vista.

7.4. Pantalla Assignments

El panel de esta vista, con mayor contenido que el anterior, está compuesto en una primera instancia de un **"formulario"** para crear asignaciones, donde se debe introducir la expresión regular y el nombre que se desea dar para esta asignación. De igual modo, introduciendo la expresión regular y seleccionando el botón de búsqueda, se puede ver una vista previa de los repositorios, de la organización en la que se encuentra el usuario, que concuerdan con la **regex**.

En esta segunda parte se divide el panel en dos para dejar a la izquierda los resultados de la búsqueda y a la derecha un listado de las **asignaciones** que el usuario ya ha creado y que puede acceder o bien eliminar.

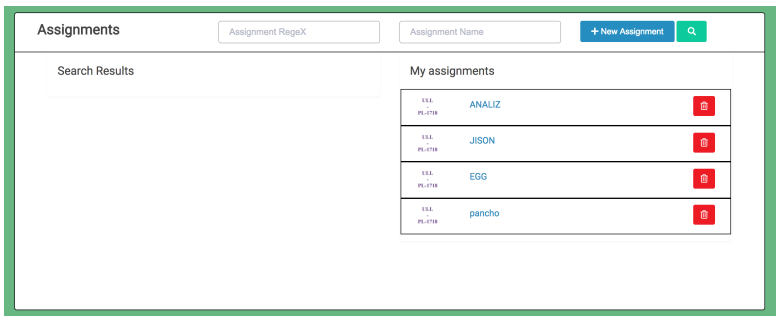


Figura 7.4: Panel Asignaciones

7.5. Pantalla Assignment repositories

Esta última pantalla a la que se accede después de seleccionar una asignación muestra en el panel una lista de repositorios que corresponden con la **expresión regular** introducida para la assignment en la que se encuentra.

Además de los repositorios, a los que se pueden acceder a su contenido en la plataforma del "octocat" pulsando sobre su nombre, aparecen los badges de Travis-CI[5], los cuales muestran el estado de los mismos pudiendo también clickear sobre las imágenes para que se abra el navegador con las pruebas del repositorio en esta plataforma.

Figura 7.5: Repositorios de una asignación

Bibliografía

[1] Feross Aboukhadijeh. Standard javascript. https://standardjs.com/.

[2] Kent Beck. extreme programming. http://www.extremeprogramming.org/, 1999.

[3] Tim Berners-Lee. Html. https://www.npmjs.com/, 1993.

[4] Github Cheng Zhao. Electronjs. https://electronjs.org/, 2013.

[5] Travis CI community. Travis ci. https://travis-ci.org/, 2011.

[6] World Wide Web Consortium. Css. https://www.w3.org/Style/CSS/, 1996.

[7] Matthew Eernisse. Ejs. http://ejs.co/, 2012.

[8] GitHub. Octokit. https://octokit.github.io/rest.js/.

[9] Node.js Developers Joyent. Nodejs. https://nodejs.org/es/, 2009.

[10] Rasmus Lerdorf. Php. http://php.net/docs.php/, 1994.

[11] Junio Hamano y Software Freedom Conservancy Linus Torvalds. Git. https://git-scm.com/, 2005.

[12] mawie81. electron-oauth2. https://github.com/mawie81/electron-oauth2, 2017.

[13] Mozilla Fundation Netscape Communications Corp. Javascript. https://www.javascript.com/, 1995.

[14] Taylor Otwell. Laravel. https://laravel.com/, 2011.

[15] Tom Preston-Werner. Github. https://github.com/, 2008.

[16] Collin Winter. Npm. https://www.npmjs.com/, 2014.