Trabajo de Fin de Grado

CodeLab

A Tool to automate repository creation and access control, giving support to distribute starter code, collect assignments and evaluate the students work on GitHub.

Samuel Ramos Barroso

La Laguna, 24 de mayo de 2018

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

C E R T I F I C A

Que la presente memoria titulada:

"*CodeLab.*"

ha sido realizada bajo su dirección por D. **Samuel Ramos Barroso**, con N.I.F. 51.165-611-H.

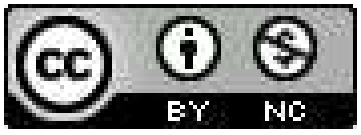
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 24 de mayo de 2018

Agradecimientos

A mi familia, por el apoyo que me han dado durante estos años.

A Casiano, por enseñarme el mundo de las tecnologías web y transmitirme sus conocimientos sobre JavaScript e iniciarme en el mundo fullstack.

Licencia



Resumen

Este proyecto trata sobre la creación plataforma web destinada al apoyo del profesorado para la realización de prácticas intentando resolver las limitaciones que tienen otras plataformas para la gestión de prácticas. El profesor podrá crear tareas que serán asignadas a sus alumnos de forma individual o grupal. Cada alumno realizará su trabajo en un repositorio git, el cual una vez finalizado, podrá ser revisado por el profesor.

Palabras clave: Git, Github, Tareas, Clases, Prácticas de Programación.

Abstract

This project deals with the creation of a web platform designed to give support to teachers for the realization of programming labs. It tries to solve the limitations of other platforms with the same goal. The teachers can create tasks that will be assigned to their students individually or in groups. Each student will perform their work in a git repository, which once it is completed, can be reviewed by the teacher.

Keywords: *Git, Github, Tasks, Classes, Programming Labs.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Herramientas para el apoyo del profesorado de informática

En los últimos años se han desarrollado las nuevas tecnologías, lo que ha permitido que lleguen nuevas herramientas de aprendizaje y de apoyo a la docencia. Es el caso de la exitosa plataforma Moodle, un LCMS, sigla de Learning Content Management System.

El Moodle, como otros LCMS, se utiliza para crear y manejar el contenido de un programa educativo, como por ejemplo un curso. Normalmente se crean partes de contenido en forma de módulos que se pueden personalizar.

Existen otras herramientas como el LMS sigla de Learning Management System. El LMS es un software instalado en un servidor que se emplea para administrar, distribuir y controlar las actividades de formación de una institución. Un LCMS suele estar integrado en un LMS.

Las principales funciones del LMS son: gestionar usuarios, recursos, materiales y actividades de formación, administrar el acceso, controlar y hacer seguimiento del proceso de aprendizaje, realizar evaluaciones, generar informes, gestionar servicios de comunicación como foros de discusión, videoconferencias y correo electrónico entre otros.

Como ejemplos reales de las definiciones mencionadas, se encuentra el servicio LCMS de Google, Google Classroom que está basado en Moodle, con la diferencia de que usa la Suite de Google Drive para el apoyo de almacenamiento y gestión de tareas.

En el caso del apoyo al profesorado de informática encontramos los VPL sigla de Virtual Programing Lab, [19] que es un módulo para la plataforma Moodle. Los VPL permiten editar y ejecutar el código fuente de las tareas de

programación en el navegador, además permite que el profesor pueda corregir sin necesidad de descargar el código fuente para ejecutarlo, también se permite añadir restricciones como pegar código externo y permite comparar la similitud entre códigos de los alumnos, entre otras cosas.

Por último, la única herramienta para la gestión de tareas está desarrollada por Github, se trata de Github Classroom, una plataforma aprovecha las organizaciones y repositorios de Github como estructura para las aulas y tareas, haciendo uso de la herramienta de control de versiones Git. En el siguiente punto hablaremos en profundidad de Github Education y Classroom.

1.2. Github Education

Github Education [4] es una comunidad desarrollada por Github para el mundo de la enseñanza y que contiene varias herramientas y servicios para apoyar a los profesores en su labor:

- GitHub Classroom [3]
- Classroom Desktop
- Teachers Pet
- Education Comunity [4]
- Student Pack

1.2.1. Classroom

Classroom es una herramienta destinada a profesores para gestionar el uso educativo de GitHub. Simplifica la asignación de tareas, automatizando la creación de repositorios git usando las organizaciones de Github como aulas y los repositorios como asignaciones.

Es una herramienta útil y sencilla de usar, tanto para profesores como para alumnos, pero tiene ciertos defectos.

- Por ejemplo, no se puede crear un repositorio de evaluación que contenga las tareas de todos los alumnos, para eso se debe usar Classroom Desktop, es decir, se requiere de otra herramienta para una única funcionalidad.
- Tampoco se puede acceder a los enlaces de travis, si la práctica requiere su uso.

- Por último, tiene un sistema para asociar información a cada alumno que es bastante complejo de usar, ya que requiere introducir a mano cada usuario y la información que se desea asociar.

1.2.2. Teachers Pet

Teachers Pet es un CLI destinado a la administración de Education vía terminal, pero dejaron de desarrollarla porque para realizar ciertas tareas requería demasiadas opciones y los comandos eran demasiado largos.

1.2.3. Education community

Education Comunity es un foro para el intercambio de información entre alumnos, profesores, investigadores y desarrolladores.

1.2.4. Student Pack

Student Pack [6] es un paquete que ofrece Github a los alumnos para mejorar su experiencia haciendo uso de Github para el desarrrollo de tareas.

Capítulo 2

Plan de trabajo

2.1. Objetivos

Los objetivos a desarrollar en este trabajo se definen a continuación:

- **A1.** Analizar otras plataformas web existentes para la gestión del código de las prácticas de informática y su metodología de trabajo.
- **A2.** Estudiar el funcionamiento de otras plataformas web existentes para la gestión del código de las prácticas de informática.
- **A3.** Estudiar las tecnologías a usar y enfocar el diseño de la plataforma web.
- **A4.** Estudiar las funcionalidades que se van a incluir en la plataforma web.
- **A5.** Crear una aplicación web básica que permita al usuario iniciar sesión con su cuenta de Github.
- **A6.** Continuar con el desarrollo de la aplicación incluyendo las funcionalidades que solucionen las dificultades de otras plataformas web existentes para la gestión del código de las prácticas de informática.
- **A7.** Diseñar y desarrollar los estilos de las vistas.

2.2. Plan de trabajo

Objetivo	**Fecha**
A1	30 Enero - 3 Febrero
A2	4 - 9 Febrero
A3	10 - 15 Febrero
A4	16 - 21 Febrero
A5	22 Febrero - 22 Marzo
A6	23 Marzo - 23 Mayo
A7	25 Abril - 20 Junio

Tabla 2.1: Tabla resumen del plan de trabajo

Capítulo 3

Desarrollo del proyecto

3.1. Tecnologías

3.1.1. Tecnologías populares

Django - Python Django es un framework para aplicaciones web gratuito y open source, escrito en Python. Un conjunto de componentes que te ayudan a desarrollar sitios web más fácil y rápidamente siguiendo el patrón Modelo Vista Controlador.

Ruby on Rails - Ruby Ruby on Rails, también conocido como RoR o Rails, es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el patrón Modelo Vista Controlador (MVC).

Express.js - Javascript Espress.js [2] es un framework de desarrollo de aplicaciones web minimalista y flexible para Node.js [13]. Está inspirado en Sinatra, además es robusto, rápido, flexible y muy simple, además, es compatible con el patrón Modelo Vista Controlador.

Aquí tenemos una pequeña comparación de la popularidad que han tenido estas tecnologías en el último año. Véase la figura 3.1.2.

3.1.2. Tecnologías Escogidas

Finalmente se escogió Express.js, Node.js y Javascript como tecnologías principales a usar en este proyecto por las siguientes razones:

En primer lugar, el uso de Node.js como tecnología de servidor aporta una gran ventaja ya que Javascript se usa en ambos lados, en el backend y en el frontend, reduciendo complejidad y tiempo de desarrollo ya que Express es bastante fácil de aprender y usar. Además, JavaScript es el lenguaje de

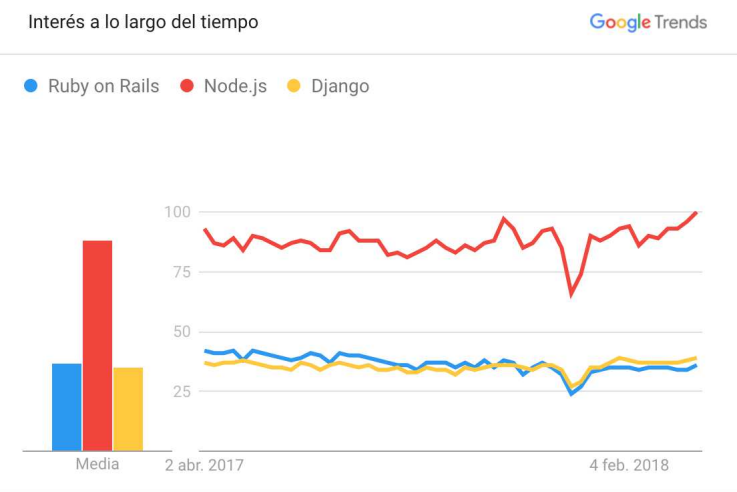


Figura 3.1: Comparativa de Tecnologias usadas

programación más popular en el desarrollo web.

Además de eso, la comunidad de Node.js está en constante crecimiento: la cantidad de preguntas de StackOverflow aumenta constantemente, por lo que la base de conocimiento para la tecnología es amplia. También hay que destacar el hecho de que Node.js sea de código abierto y gratuito.

Finalmente, Node nos ofrece NPM, un gestor de paquetes que dispone de una gran cantidad de paquetes que crece rápidamente y aporta una forma sencilla de gestionar los paquetes que necesitemos en nuestro proyecto.

NPM

NPM [14] es un administrador de paquetes para Node.js con cientos de miles de paquetes. Aunque crea parte de la estructura del directorio del proyecto, este no es el objetivo principal.

El objetivo principal es la automatización de dependencias y gestión de paquetes de cada proyecto. Esto significa que puede especificar todas las dependencias de su proyecto dentro del fichero `package.json` usando el comando `npm i -s nombrepaquete`, de forma que cada vez que un usuario clone el proyecto en su máquina simplemente tenga ejecutar `npm install` e inmediatamente se descargan e instalan todas las dependencias necesarias para el funcionamiento del proyecto en el directorio `node modules/`. Además, también sirve para especificar en qué versión está su proyecto.

También es posible descargar manualmente los paquetes, copiarlo en el directorio `nodemodules/` y usarlo de esa manera. Sin embargo, a medida que crezca el proyecto y la lista de dependencias, llevará mucho tiempo instalar los paquetes necesarios y será una tarea engorrosa. También hace que colaborar y compartir tu proyecto sea mucho más difícil

Express.js

Express es un framework ''minimalista" que permite crear una infraestructura web simple sobre Node.js. Express permite crear una API REST, un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP:

- GET: Para consultar y leer recursos
- POST: Para crear recursos
- PUT: Para editar recursos
- DELETE: Para eliminar recursos
- PATCH: Para editar partes concretas de un recurso

Express también permite crear una aplicación web siguiendo el patrón de diseño Modelo Vista Controlador, que separa los datos y la lógica de de una aplicación de su representación.

Github API

Para poder usar Github como estructura necesitaremos hacer uso de su Github REST API v3 [5]. La API nos permitirá acceder a los servicios que ofrece Github de forma externa, es decir, podremos crear repositorios, usar organizaciones como clases y manejar otros servicios, como consulta de información sobre el usuario. Para poder acceder a la clave de cada usuario, debemos usar un Token generado por Github, es decir, una cadena que nos permitirá identificarnos para realizar transacciones usando la API, en este caso, el token será generado usando las aplicaciones OAuth que nos ofrece Github para que el usuario haga login con su cuenta de usuario de Github, sin necesidad de poner su usuario y contraseña de Github en la plataforma Codelab. La Github REST API v3 tiene además varias librerías de terceros y tres librerías oficiales escritas en Ruby, Javascript y .Net.

MongoDB

MongoDB [11] es un sistema gestor de base de datos NoSQL, es decir, MongoDB es un sistema de bases de datos no relacional, en lugar de almacenar los datos en tablas, se almacenan colecciones o documentos, que está formado por objetos, y estos a su vez por claves y cada clave tiene un valor asociado. Es decir, sería algo parecido a un documento JSON, aunque en MongoDB usan una distribución llamada BSON. Véase la figura 3.2.

Mongoose

Mongoose [12] es un una herramienta para el diseño de objetos de MongoDB. Mongoose provee una serie de métodos y funciones para manejar la base de datos en MongoDB.

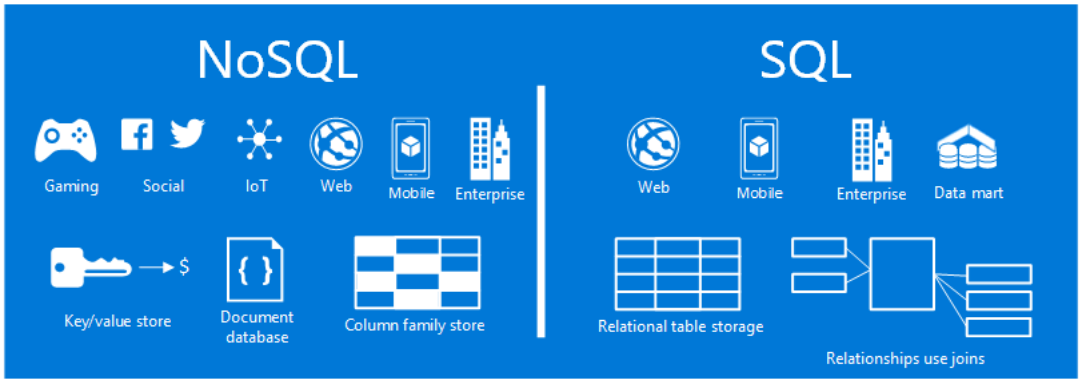


Figura 3.2: NoSQL vs SQL

Motor de vistas

Como motor de vistas para el proyecto se ha elegido Pug [17], antes conocido como Jade, un motor de vistas implementado en Javascript para su uso en NodeJS y que dispone de una fácil integración con Express.JS.

Automatización de tareas

Para la automatización de tareas como generar el fichero de variables de entorno o poner en marcha un servidor nodemon se ha usado Gulp [7], una herramienta que permite automatizar tareas en Node.js, es simple y bastante fácil de usar.

Framework de CSS

Como framework para los estilos de CSS se ha usado Materialize CSS [10], basado en material design [9]. Ofrece varias opciones a la hora de diseñar la plataforma, permitiendo un diseño más minimalista o un diseño más cargado y vistoso.

3.2. Github API

Para poder usar Github como estructura para la creación de aulas y tareas de código usaremos la Github REST API V3. La API permite acceder a las funcionalidades de Github, A continuación se detallará que funcionalidades de la Github API se han usado

- Oauth
- Organizaciones, Repositorios y Equipos

3.2.1. OAuth

La funcionalidad más básica que se usa es OAuth [15], que permite loguear al usuario con su cuenta de Github en Codelab.

OAuth, sigla de Open Authorization es un estándar de código abierto que permite a un usuario compartir su información de un proveedor de servicios, como Google, Twitter, Facebook o en este caso de Github, con un consumidor, en nuestro caso Codelab.

Desde el punto de vista de Codelab, OAuth proporciona un método de acceso a los datos de Github a la vez que se protegen los credenciales de la cuenta.

Desde Codelab, solicitamos al usuario una serie de accesos y permisos en sus datos, para poder operar con las organizaciones y repositorios de Github.

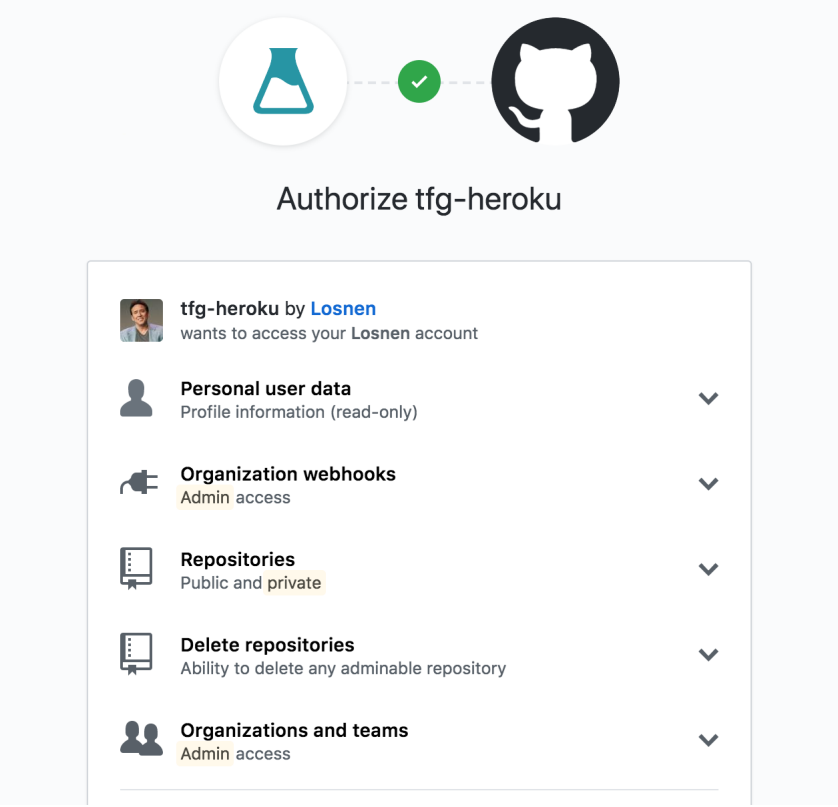


Figura 3.3: Vista de autorización de la OAuth App

Para tener acceso a OAuth primero debemos crear ua aplicación de OAuth en Github, simplemente tenemos que facilitarle a Github el nombre de nuestra aplicación, el dominio y un Callback URI, que es donde el servicio de Github redirigirá al usuario después de autorizar o denegar su aplicación, Github devolverá un API Key y un Secret API Key para que lo introduzcamos en la plataforma.

Para simplificar el uso de OAuth en la plataforma, se usa un framework

llamado Passport.js diseñado para simplificar el Sign up y Log in para Express.js y que permite el uso de Github OAuth, Passport nos solicita las claves de la API, un callback url y los scopes o permisos que el usuario debe aprobar, para devolver los datos solicitados al usuario en forma de token para poder acceder a las demás funcionalidades de la Github API.

3.2.2. Organizaciones, repositorios y equipos

El acceso a las organizaciones, repositorios y equipos se realiza mediante una librería oficial de Github para Node.js llamada Octokit, que permite casi todas las funcionalidades de la Github REST API, además ofrece varias formas de llamar a sus funciones, mediante el uso de async/await, promise y callback.

Para abordar la gestión, se utiliza una clase llamada Github en la que el constructor inicializa la API con el token del usuario y cada método accede a una funcionalidad de la API.

```
class Gh {
  constructor (token) {
    octokit.authenticate({ type: 'oauth', token: token })
  }

  userOrgs () {
    return new Promise((resolve, reject) => {
      octokit.users.getOrgs({ }, (error, result) => {
        if (error) reject(error)

        resolve(result)
      })
    })
  }
}
```

Figura 3.4: Clase Github

Las funcionalidades que aporta la API a Codelab son las siguientes:

- Oauth
- Obtener las organizaciones del usuario para usar .
- Añadir usuarios a la organización.
- Crear un repositorio en las organizaciones.
- Añadir colaboradores al repositorio.
- Crear equipos.
- Añadir un equipo a un repositorio.

- Comprobar si un usuario es miembro de un equipo.
- Obtener los repositorios de una organización.
- Crear un fichero en un repositorio.

3.3. Modelo-Vista-Controlador

3.3.1. El MVC

El modelo vista controlador es una arquitectura de software que separa la lógica de la aplicación de la interfaz de usuario. Lo hace separando la aplicación en tres partes: el modelo, la vista y el controlador.

El modelo maneja los datos fundamentales, responde a las solicitudes de información, solicitudes para insertar, actualizar o eliminar la información. Esto podría ser una base de datos o cualquier sistema de estructura de datos. En resumen, es la gestión de datos y datos de la aplicación.

La vista proporciona la interfaz de usuario de la aplicación. Transformará los datos del modelo para mostrárselos al usuario de forma que pueda entender y manejar la información de forma sencilla e intuitiva.

El controlador recibe los datos y acciones del usuario y realiza llamadas al modelo para manejar los datos y a la vista para mostrar el resultado de dichas acciones. Véase la figura 3.5.

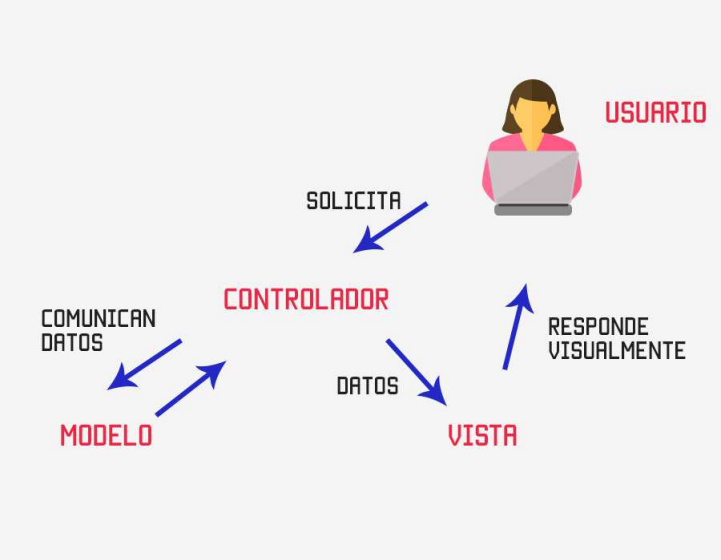


Figura 3.5: MVC

3.3.2. MVC en Codelab

En la plataforma Codelab se ha implementado el Modelo Vista Controlador, usando las funciones que nos proporciona Express.JS.

En el caso de las vistas Express nos permite establecer un motor de vistas. Véase la figura 3.6.

```
app.set('views', path.join(__dirname, '/app/views/'))
app.set('view engine', 'pug')
```

Figura 3.6: Motor de vistas

Con la primera línea de código le indicamos que la carpeta de vistas estará en el directorio app/views y con la segunda línea le indicamos que motor de vistas usaremos: Pug JS.

En el código de abajo se puede ver la vista de error, que muestra al usuario el error y se le muestra el mensaje de error según el mensaje que se le pase desde el controlador. Véase la figura 3.7.

```
doctype html
- var user = usuario
- var mensaje = msg

html
  include ../layouts/headParams2.pug
  body
    include ../layouts/nav.pug
    div(class="container")
      div(class="row")
        div(class="s12", id="space")
          p(align="center")
            i(class="fas fa-exclamation-triangle erroricons")
          p(align="center", class="errortxt") Error: #{mensaje}

    include ../layouts/footerParams2.pug
```

Figura 3.7: Ejemplo de vista en Pug

Pug es un motor de vistas bastante potente. Se pueden modular las vistas para reutilizar código y evitar repetirlo. Por ejemplo, se puede dividir el código en Head donde se definen los distintos enlaces al CSS, JS y se pone el titulo, Nav que contiene la barra de navegación, que es igual para todos, el body que es el contenido de la vista y por último el footer. También se pueden hacer bucles para mostrar las tareas o las aulas o mostrar un valor u otro dependiendo de si se tiene rol de alumno o de profesor.

En el caso de los modelos se usa Mongoose y estarán ubicados en la carpeta de app/models. Los modelos le proporcionan la información a los controladores usando las funciones de mongoose, find, update, delete y save entre otras. Véase la figura 3.8.

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

const UserSchema = new Schema({
  login: { type: String, unique: true, lowecase: true },
  id: String,
  token: String,
  signupDate: { type: Date, default: Date.now() },
  lastLogin: Date
})

module.exports = mongoose.model('User', UserSchema)
```

Figura 3.8: Ejemplo de modelo

Y por último, en el caso de los controladores, se definen funciones con las tareas que tienen que hacer de forma que todo esta modulado. El controlador le pasa la información a las vistas usando el comando render, que renderiza la vista de Pug a HTML. Véase la figura 3.9.

```
// Controller for get the group invi page.
function groupInvi (req, res) {
  let tarea = req.params.idassign
  let aula = req.params.idclass
  let titulo = 'Tarea ' + req.params.idassign

  Team.find({ org: aula }, (err, teams) => {
    if (err) console.log(err)

    res.render('assignments/groupInvi', { titulo: titulo })
  })
}
```

Figura 3.9: Ejemplo de Controlador

Router se usa para crear rutas modulares. Una instancia Router es un sistema de middleware y direccionamiento completo que a menudo se denomina pequeña aplicación o mini aplicación.

Cabe destacar que para las rutas genéricas como aula o asignación en la que se puede representar los datos de cualquier aula o asignación se usan rutas que dependen de parámetros, es decir, rutas que muestran unos datos dependiendo de los parámetros que se le pasen. Véase la figura 3.10.

```
api.get('/', appControllers.home)
```

Figura 3.10: Ejemplo de ruta

3.4. Diseño de la base de datos

Se usará una base de datos NoSQL, en concreto MongoDB como sistema gestor y Mongoose como ODM, por lo que nos referimos las tablas como colecciones, columnas como claves y filas como objeto, todas las claves se escribirán usando el estilo camel case.

La base de datos es una parte muy importante de la plataforma, ya que en ella recae toda la responsabilidad de simular toda la estructura de aulas y tareas.

3.4.1. Colección de usuarios

Para almacenar los datos de los usuarios, existe una colección llamada user, donde se almacenarán datos como el id de github o el token:

- login: Nombre de usuario de github, de tipo String y clave primaria.
- id: ID de github, de tipo string.
- token: Token para autorizar a la api de github, de tipo string.
- signupDate: Fecha en la que el usuario se registro, de tipo fecha
- lastLogin: Fecha en la que usuario se logueo por última vez, de tipo fecha.

3.4.2. Colección de organizaciones

Para almacenar los datos de las organizaciones existe una colección llamada orgs, que simula las aulas. Cada aula tendrá los siguientes datos:

- login: Nombre de la organización en github, de tipo String y clave primaria.
- id: ID de github de la organización, de tipo string.
- avatar: Enlace a la foto de la organización, de tipo string.
- ownerID: ID del usuario dueño de la organización en github, de tipo string.
- ownerLogin: Nombre del usuario dueño de la organización en github, de tipo string.
- isActive: Booleano para identificar si las invitaciones a la organización están activas, de tipo boolean.
- createDate: Fecha en la que se registro la organización, de tipo fecha.

3.4.3. Colección de asignaciones

Para almacenar los datos de las tareas existe una colección llamada assigns, que simula las tareas. Cada tarea contendrá los siguientes datos:

- title: Nombre de la tarea, de tipo String y clave primaria.
- ownerLogin: Nombre del usuario dueño de la tarea de tipo string.
- assignType: Tipo de aisngación, grupal o individual, de tipo string.
- repoType: Tipo de repositorio, privado o público, de tipo string.
- orgLogin: Nombre de la organización de github (aula) a la que pertenece la asignación, de tipo string.
- isActive: Booleano para identificar si las invitaciones a la tarea están activas, de tipo boolean.
- createDate: Fecha en la que se registro la tarea, de tipo fecha.

3.4.4. Colección de asignaciones individuales

Para almacenar los datos de las tareas individuales existe una colección llamada repos, que simula las tareas individuales. Cada tarea contendrá los siguientes datos:

- name: Nombre del repositorio en github, de tipo String y clave primaria.
- assignName: Nombre de la tarea, de tipo string.
- StudentLogin: Nombre de github del alumno que realizará la tarea, de tipo string.
- ownerLogin: Nombre de github del dueño del aula, de tipo string
- orgLogin: Nombre de la organización de github (aula) a la que pertenece el repo, de tipo string.
- createDate: Fecha en la que se registro la tarea, de tipo fecha.

3.4.5. Colección de asignaciones grupales

Para almacenar los datos de las tareas grupales existe una colección llamada groups, que simula las tareas grupales. Cada tarea grupal contendrá los siguientes datos:

- name: Nombre del repositorio en github, de tipo String y clave primaria.
- team: Nombre del equipo que realizará la tarea, de tipo string.
- idTeam: ID del equipo que realizará la tarea, de tipo string.
- assignName: Nombre de la tarea, de tipo string.
- StudentLogin: Nombre de github del alumno que realizará la tarea, de tipo string.
- ownerLogin: Nombre de github del dueño del aula, de tipo string
- orgLogin: Nombre de la organización de github (aula) a la que pertenece el repo, de tipo string.
- createDate: Fecha en la que se registro la tarea grupal, de tipo fecha.

3.4.6. Colección de equipos

Para almacenar los datos de los equipos existe una colección llamada teams. Cada equipo contendrá los siguientes datos:

- name: Nombre del equipo, de tipo string.
- id: ID del equipo que realizará la tarea, de tipo string.
- members: Miembros que forman parte del equipo, de tipo array de string.
- orgLogin: Nombre de la organización de github (aula) a la que pertenece el equipo, de tipo string.
- createDate: Fecha en la que se registro el equipo, de tipo fecha.

3.4.7. Colección de alumnos

Para almacenar los alumnos de un aula existe una colección llamada students. Cada alumno tendrá los siguientes datos:

- name: Nombre del alumno, de tipo string.
- surname: Apellido del alumno, de tipo string.
- email: Email del alumno, de tipo string.
- idGithub: Nombre de usuario del alumno en github, de tipo string
- createDate: Fecha en la que se registro el alumno de tipo fecha.

3.5. Descripción general de la plataforma web

Codelab es el nombre de la plataforma web que está destinada a simular el funcionamiento de un aula que contiene tareas grupales o individuales sobre Github, es decir, como ya se ha mencionado anteriormente las organizaciones simularán las aulas y los repositorios las tareas.

3.5.1. Diseño

El diseño se realizó en Materialize CSS, un framework basando en material dessign. Como se puede apreciar, el diseño es bastante minimalista e intuitivo, aunque esté dirigido a un sector que maneja las nuevas tecnologías, se pretende que la plataforma sea lo más simple posible de manejar, tanto para profesores como para alumnos. El usuario no debe registrarse, simplemente debe tener una cuenta de Github y loguearse con ella aceptando los permisos para poder ofrecerle los servicios de la plataforma, por lo que no es necesario rellenar ningun formulario de Log in o Sign Up. Véase la figura 3.11.

Figura 3.11: Landing Page

3.5.2. Navegación

La navegación dentro de la aplicación es bastante simple e intuitiva, existen dos barras de navegación:

La primera es la barra superior, que contiene los botones de Sign In si no se está logueado y los botones de Perfil, Aulas y Sign Out en caso de que se esté logueado. Véase la figura 3.12.

Figura 3.12: Barra de navegación

La segunda está dentro de cada vista, si estamos en la vista de aulas veremos dos pestañas: Aulas y nueva Aula.

En cambio si estamos en la vista de aula veremos tareas, enlace de invitación, nueva asignación, opciones, alumnos y subir fichero de alumnos.

Por último si nos encontramos en la vista de asignación se verá: repositorios, enlace de invitación, repositorio de evaluación y opciones. Véase la figura 3.13.

Figura 3.13: Barra de navegación de aula

3.6. Funcionalidades

El proyecto se divide en tres paquetes de funcionalidades, Cada paquete incluye funcionalidades para cada rol:

- Funcionalidades básicas
- Funcionalidades para profesores.
- Funcionalidades para el alumno

3.6.1. Funcionalidades básicas

Las funcionalidades básicas son comunes a todos los roles que participan en la plataforma, tanto alumnos como profesores,

Todos los usuarios de la aplicación pueden hacer Log in, Log out y consultar un perfil.

En el perfil del usuario se puede consultar la información básica del usuario en Github, foto de perfil, nombre de usuario, correo y en el caso de que sea un alumno puede consultar las tareas que tiene en grupo o individualmente. Véase la figura 3.14.

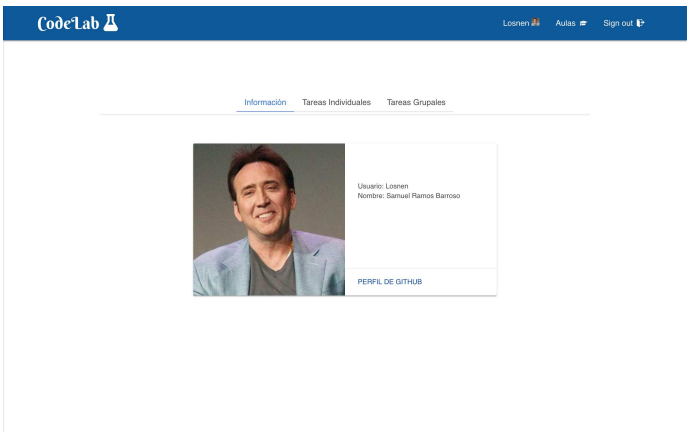


Figura 3.14: Perfil de usuario

3.6.2. Funcionalidades para profesores

Los profesores tendrán el grupo de funcionalidades más completo, ya que ellos son los protagonistas de la app, los profesores podrán desempeñar las siguientes tareas:

- Añadir una organización como aula.
- Invitar alumnos al aula.
- Crear una tarea.
- Añadir un fichero de alumnos asociado al aula.
- Editar las opciones del aula.
- Invitar alumnos a la tarea.
- Editar las opciones de la tarea.
- Crear un repositorio de evaluación de cada tarea.

Añadir una organización

Añadir una organización como aula es la primera funcionalidad que puede desempeñar un profesor, en esta funcionalidad el profesor podrá elegir una organización de las que tenga creadas y a las que haya autorizado el acceso de terceros, es decir, el profesor tendrá que crear una organización y darle acceso de terceros antes de poder asignarla como aula en Codelab, una vez creado el aula, se le redirigirá al profesor a la vista donde puede ver todas sus aulas. Véase la figura 3.15.

Figura 3.15: Aula

Invitar alumnos al aula

Para tenerlo todo bien organizado, será necesario tener a todos los alumnos incluidos de la organización de Github y que los alumnos tengan asignados el rol de miembro en lugar de colaboradores externos. Aunque las diferencias entre los colaboradores externos y los miembros son mínimas, es importante que los alumnos estén en la organización como miembros, ya que los colaboradores externos no tendrán acceso a todos los repositorios de la organización, sólo a aquellos en los que el profesor los incluya y además no podrán crear equipos, ver los miembros de la organización, ser un maintainer de un equipo o mencionar a otros equipos visibles.

Por eso, el profesor podrá invitar a los alumnos a la organización mediante un enlace de invitación, que el alumno deberá aceptar para formar parte de la organización, así, el profesor podrá automatizar el proceso de admisión en el aula, sin tener que añadir alumnos uno a uno buscando por el identificador de Github.

Desde las opciones del aula el profesor podrá activar o desactivar las invitaciones para que sus alumnos puedan unirse. Véase la figura 3.16.

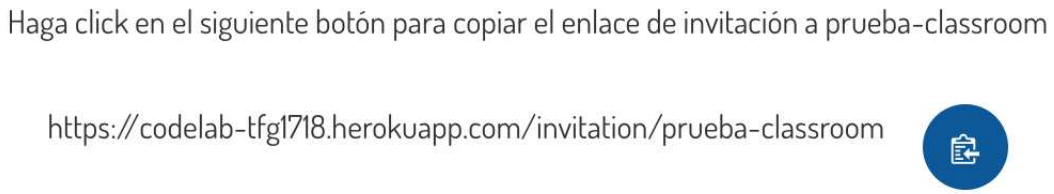


Figura 3.16: Invitación

Crear una nueva tarea

Esta funcionalidad destaca entre todas, crear la tarea es la funcionalidad más importante de la plataforma, ya que esta funcionalidad permite la automatización para la creación de los repositorios. Si el profesor quiere crear una tarea es tan fácil como asignarle un título, elegir si el repositorio es público o privado y por último definir si la tarea es individual o grupal.

Después de crear la tarea, se redirigirá a la vista de la tarea, donde el profesor podrá gestionar las tareas de cada alumno. se podrá ver una breve información del usuario si el profesor añadió un fichero xslx o csv con la información de los alumnos.

En esa vista aparecerá un pequeño menú que tiene la pestaña descrita en el

párrafo anterior, una pestaña para el enlace de invitación, una pestaña para crear el repositorio de evaluación una última pestaña de opciones, todas estas, son las opciones disponibles para operar con las tareas.

Desde las opciones de la tarea el profesor podrá activar o desactivar las invitaciones para que sus alumnos puedan unirse. Véase la figura 3.17 y 3.18

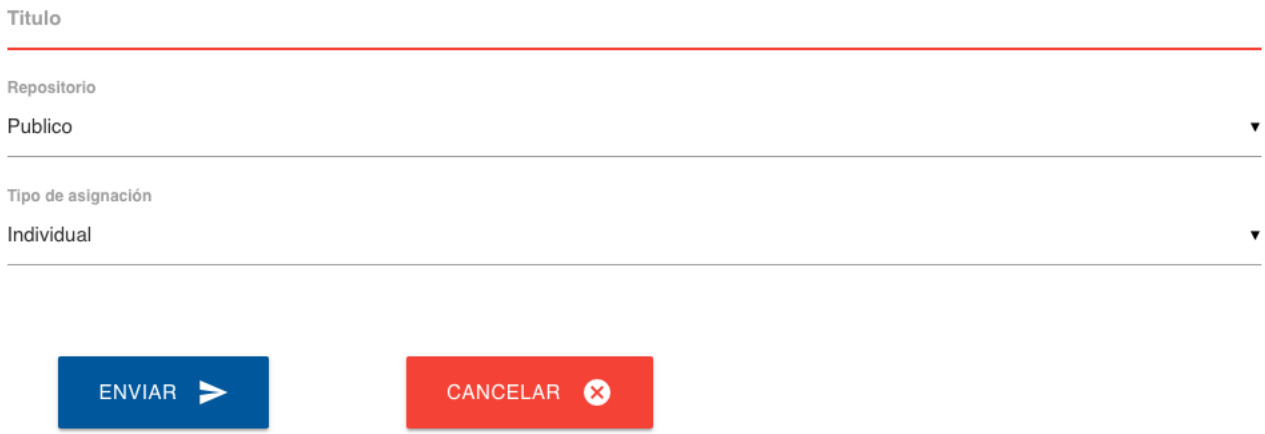


Figura 3.17: Nueva tarea

Figura 3.18: Gestión de tareas

Añadir un fichero de alumnos asociado al aula

El mayor problema que tiene asociar Github con la automatización de tareas es asociarle información a cada alumno, ya que pueden crearse una cuenta en Github con el nombre, foto o ID de github que les apetezca por lo que identificar a un alumno puede ser complicado.

Por este motivo, la plataforma tiene un apartado para subir un fichero xslx o csv con información asociada a los alumnos de un aula en concreto, la información que debe ser asociada es la siguiente:

- Nombre.
- Apellido.
- Dirección de correo.
- ID de Github.
- Nombre de la organización de Github que será usada como aula.

La ventaja es que el campus virtual genera un archivo en formato xslx con la mayoría de datos necesarios, salvo el ID de github de cada alumno y el nombre de la organización, es decir, el profesor tendría que solicitar al alumno su ID de github y añadir la organización de github que se usará como aula. Por tanto, el profesor no debe escribir un fichero xslx desde cero.

Crear un repositorio de evaluación de cada tarea

Otro gran problema de automatizar la creación de tareas es la corrección de las mismas. Desde el punto de vista de cada profesor se vuelve engorroso clonar cada repositorio, y ejecutar todos los comandos para que la práctica del alumno esté lista para corregir.

Pongamos de ejemplo la asignatura de Procesadores de Lenguajes que tiene 31 alumnos, es decir, cada tarea podría tener hasta 31 repositorios, que debería clonar uno a uno y ejecutar dentro de cada repositorio local los comandos para que funcione cada proyecto, por ejemplo `npm install`, para instalar las dependencias NodeJS.

La solución a este problema se propone haciendo uso del comando `git` combinado con la opción submodule. `git` submodule nos proporciona un repositorio incrustado dentro de otro, es decir, en nuestro caso, sería un repositorio de alumno incrustado dentro de un repositorio de evaluación que contiene todos los repositorios de esa tarea, esto lo consigue mediante la creación del fichero

`.gitmodules`, que contiene todos los repositorios submodulos. `git` submodule tiene varias opciones:

- `git submodule init repo` Inicializa el fichero `.gitmodules` con el repositorio que se le pase.
- `git submodule add repo` Nos permite añadir un repositorio como submódulo del repositorio actual, crea el fichero `.gitmodules` si no existe.
- `git submodule update repo` Actualiza los submodulos.
- `git submodule foreach repo` Permite ejecutar un comando en cada submódulo de forma paralela.
- `git submodule sync repo` Sincroniza los submodulos con el remoto.

Una vez explicado que es `git` submodule y vista su utilidad veremos como se integra con Codelab y la automatización de corrección.

Codelab crea un repositorio de corrección en Github con el nombre eval-nombre-tarea, en un principio la idea fue crear un repositorio que contuviese como submódulos todos los repositorios de la tarea en cuestión, pero la api de Github no permite esta funcionalidad, como segunda opción se pensó crear un fichero que contuviese directamente el fichero `.gitmodules`, pero no funciona si el repositorio no está clonado y como última opción se crea un repositorio que contiene un pequeño script en bash que contiene una línea con el comando `git` submodule add ¡repo¿por cada repositorio de alumno, de forma que al ejecutar el comando se añaden todos los repositorios de forma masiva como submódulos.

Una vez creado el repositorio de evaluación, el profesor debe clonar dicho repositorio y ejecutar el script `eval.sh`, añadiendo todos los repos de alumnos como submodules de forma masiva como se explicó anteriormente. Una vez hecho esto, el profesor podrá ejecutar cómodamente comandos en todos los repositorios de forma paralela usando el comando `git` submodule foreach repo, como por ejemplo `git` submodule foreach `npm i`, que instala todas las dependencias de NodeJS en cada práctica.

3.6.3. Funcionalidades para alumnos

Perfil

Como alumno el usuario puede visitar el perfil, donde encontrará información básica de Github y dos pestañas, en las que el alumno tiene un historial de las tareas que ha realizado de forma grupal e individual. Véase la figura 3.19.

Figura 3.19: Perfil

Capítulo 4

Caso de uso

4.1. Introducción

Con vistas a probar y testear que todo funcionaba de forma correcta, Casiano me sugirió probar la plataforma para la realización de algunas prácticas individuales y grupales. Por ello, decidimos realizar algunas tareas para la asignatura de Procesadores de Lenguajes en CodeLab.

4.2. Detección de errores

Al poner en uso la plataforma, los alumnos de Procesadores de Lenguajes nos ayudaron a encontrar algunos errores:

- Imposibilidad de crear repositorios privados
- No se añadía el usuario como colaborador al repositorio.

Los alumnos utilizaron la funcionalidad de los issues para notificar el error añadiéndolos en el repositorio.

Todas los errores fueron solucionados lo más rápido posible para que los alumnos de Procesadores de Lenguajes pudiesen usar la plataforma de forma correcta y realizar sus tareas en su repositorio correspondiente. Por tanto, CodeLab se comenzó a usar en producción en un aula de 31 alumnos, demostrando que es una herramienta totalmente funcional y capaz de gestionar las tareas de código de una asignatura. Véase la figura 4.1.

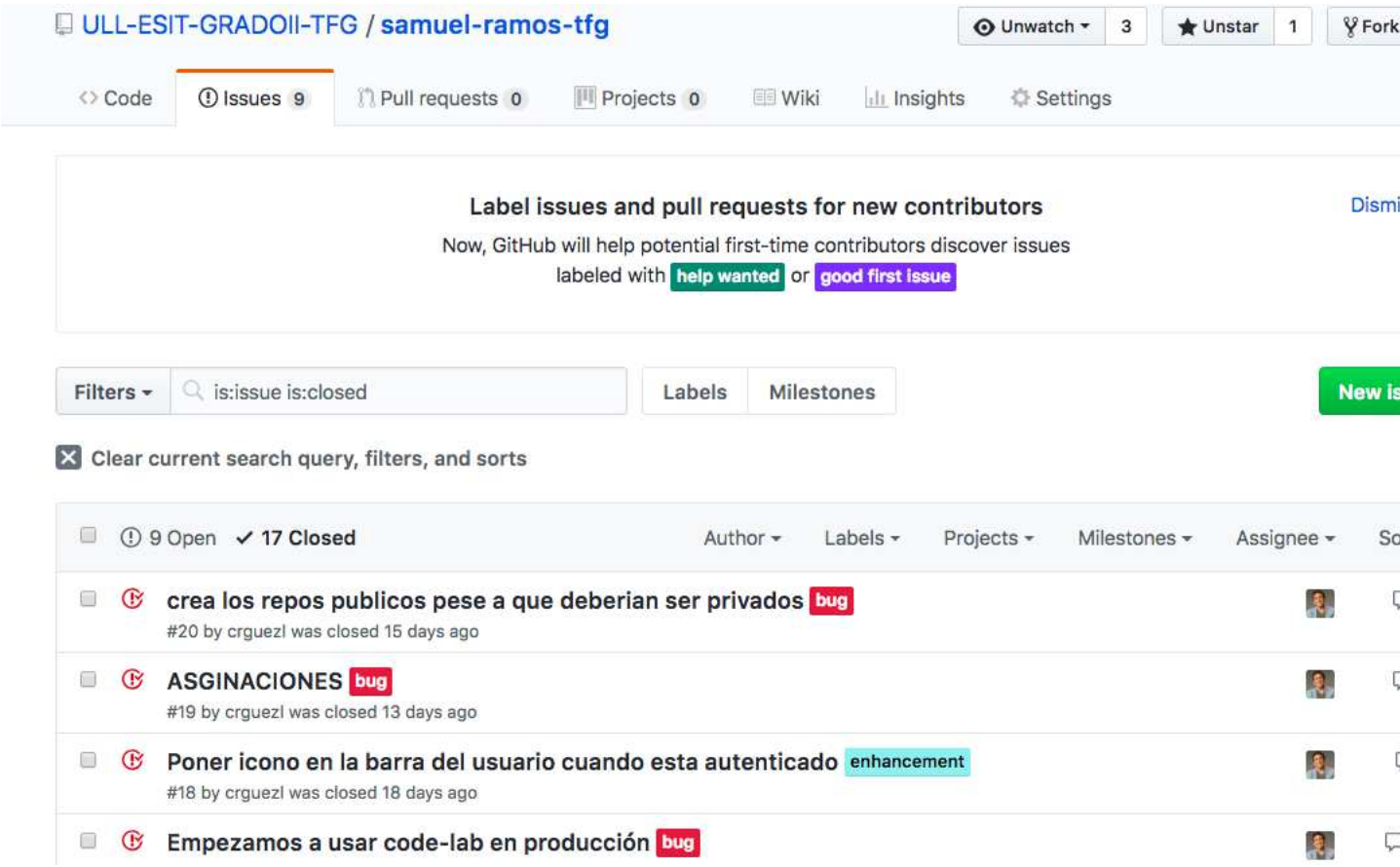


Figura 4.1: Incidencias Cerradas

Aparte de errores, Casiano encontró varias mejoras que podrían ser útiles para la plataforma:

- Foto y nombre del usuario en la barra de navegación.
- Ubicación del profesor, es decir, especificar si se encuentra en un aula o en una tarea.
- Elección de permisos del alumno dentro del repositorio de Github.
- Algunos fallos visuales como sombreados que impedían que el texto se viese de forma correcta.

Las mejoras fueron valoradas y realizadas para mejorar la interfaz gráfica y pulir ciertos defectos básicos. Sin el test de un profesor estos defectos y mejoras hubiesen sido más complicados de encontrar y resolver.

Capítulo 5

Conclusiones y líneas futuras

CodeLab nace como una herramienta que pretende extender las funcionalidades de otras herramientas como Github Classroom, incorporando respuestas a las necesidades específicas del profesorado para apoyar la gestión de los cursos y también la corrección de prácticas.

El uso de la plataforma se ha diseñado pensando en su facilidad de uso para aquellos que no estén tan familiarizados con los conceptos de GitHub.

Por otro lado, se ha tenido especial cuidado en cumplir lo máximo posible con los estándares del Software Libre, facilitando el desarrollo colaborativo y facilitando la extensión de nuevas funcionalidades por parte de otros programadores.

Creemos que las facilidades que proporciona CodeLab son útiles para aquellos profesores que tengan grupo grandes de alumnos incluso si no están familiarizados con el entorno de Github. CodeLab automatiza la creación de repositorios y el manejo de los permisos de acceso de cada alumno a sus repositorios y a los repositorios de los compañeros

Además, el control de versiones ofrece muchas ventajas a los desarrolladores. En la actualidad, todas las empresas de desarrollo usan control de versiones git, por lo que, es fundamental que los alumnos aprendan a manejar git de forma correcta y aprendan técnicas de trabajo en equipo así como todas las herramientas que proporciona Github como los issues o los projects.

En el futuro, me planteo seguir desarrollando CodeLab, mejorando la herramienta y añadiéndole nuevas funcionalidades. Una de las primeras mejoras que se plantean es la utilización de una librería de front-end como Vue o React para mejorar la calidad visual de la plataforma web. Otra nueva funcionalidad que me gustaría añadir es la posibilidad de más de un profesor por aula.

Capítulo 6

Summary and Conclusions

CodeLab was born as a tool that aims to extend the functionality of other tools such as Github Classroom, adding specific functions for the teachers to support the management of courses and the correction of programming labs.

The platform has been designed with ease of use in mind for those who are not familiar with GitHub. On the other hand, emphasis has been placed on fulfilling the Free Software standards as much as possible, facilitating collaborative development and the inclusion of new features by other programmers.

We believe that the facilities provided by CodeLab are useful for lecturers who have large groups of students or who are not familiar with the Github environment.

Version control offers many advantages to developers. Currently, most development companies use the git version control system, and consequently it is essential that students learn to handle git correctly and learn teamwork techniques as well as all the tools that Github provides such as issues or projects.

In the future, I would like to continue developing CodeLab, improving it and adding new functionalities. One of the first improvements that is proposed is the use of a front-end library such as Vue or React to improve the visual quality of the web platform. A new functionality that I would like to add is the possibility of more than one teacher per classroom.

Capítulo 7

Presupuesto

En este apartado se describirá el presupuesto del proyecto.

7.1. Coste de servicios

Para el despliegue de la aplicación se han utilizado dos servidores, un servidor en Heroku que se ha usado de forma gratuita, pero que si se pagase hubiesen sido aproximadamente 30 euros en estos 4 meses que ha durado el desarrollo.

Por otro lado, el STIC ULL nos cedió una servidor virtualizado en OVirt con sistema operativo Ubuntu y una dirección IP pública para el despliegue de la aplicación web, si hubiésemos acudido a una empresa como Digital Ocean, que nos proporcionase un servidor de las mismas características nos hubiese salido 70 euros en total.

7.2. Coste humano

El proyecto ha tenido una duración de 4 meses, si fuese remunerado implicaría un coste de 1.500 euros por mes según el convenio, es decir, un total de 6.400 euros por el proyecto.

7.3. Coste total

El presupuesto total para realizar un proyecto de estas características sería de un total de 6.500 euros, teniendo en cuenta el coste humano y el coste de servicios.

Apéndice A

Repositorios

A.1. Repositorio del proyecto en Github

https://github.com/ULL-ESIT-GRADOII-TFG/samuel-ramos-tfg

A.2. Repositorio de la memoria en Github

https://github.com/Losnen/memoria-tfg

A.3. Enlace a la plataforma

https://codelab-tfg1718.herokuapp.com/

Bibliografía

[1] Documentación de Octokit - Rest JS. https://octokit.github.io/rest.js/.

[2] Express JS. http://expressjs.com/es/.

[3] GitHub Classroom. https://classroom.github.com/.

[4] GitHub Education. https://education.github.com/.

[5] Github REST API V3. https://developer.github.com/v3/.

[6] GitHub Student Pack. https://education.github.com/pack.

[7] Gulp JS. https://gulpjs.com/.

[8] Heroku. https://devcenter.heroku.com/.

[9] Material Design. https://material.io/design/.

[10] Materialize CSS. https://materializecss.com/.

[11] Mongo DB. http://mongoosejs.com/.

[12] Mongoose JS. http://mongoosejs.com/.

[13] Node JS. https://nodejs.org/es/docs/.

[14] NPM. https://www.npmjs.com/.

[15] OAuth. https://oauth.net/.

[16] Passport JS. http://www.passportjs.org/docs/.

[17] Pug JS. https://pugjs.org/api/getting-started.html.

[18] Stack Overflow. https://es.stackoverflow.com/.

[19] Virtual Programming Lab. http://vpl.dis.ulpgc.es/.