**Proyecto Sistema De Gestión De Exámenes**

Memoria de la Aplicación

**Centro Educativo:** Escuela Virgen de Guadalupe

**Ciclo Formativo:** Desarrollo de Aplicaciones Web

**Curso:** 2020-21

**Alumno:** Ignacio Lorenzo Vélez

**Tutor:** Miguel Jaque Barbero

**ÍNDICE:**

1.-Introducción.

2.-Guia de estilo.

3.-Estudio de aplicaciones similares.

4.-Libro de estilo

4.1.-Normas Generales.

4.2.-Normas para Ficheros.

4.3.-Normas para SQL.

4.4.-Normas para HTML.

4.5.-Normas para CSS.

4.6.-Normas para Javascript.

4.7.-Normas para PHP.

5.-Arquitectura tecnológica

6.-Actores y objetivos

7.-Modelo de datos

7.1.-E/R

7.2.-Base de Datos

7.3.-Diccionario de Datos

8.-Mapa de Navegación

9.-Diagrama de clases

10.-Bocetos de Páginas

11.-Planes de mejora

12.-Tecnología usada.

13.-Problemas encontrados.

14.-Sprints.

15.-Listado de Requerimientos

16.-Explicación del Modelo/Vista/Controlador.

# 1.-Introducción:

Una de las actividades de formación que más tiempo consume a los profesores es el planteamiento y la corrección de exámenes. Elegir las preguntas más adecuadas para la evaluación del alumno, su tipo (será tipo test, abierta, texto corto…), confeccionar el examen y sobre todo corregirlo consume tiempo que de otra forma podría emplearse en la preparación de clases, formación del profesorado o incluso vida familiar y ocio.

El mismo tiempo, los alumnos ven por primera vez el examen el mismo día en que se examinan, lo que implica un "factor sorpresa" que puede mermar sus resultados. Muchos buscan exámenes de años anteriores o de otras instituciones para conocer de antemano el tipo de preguntas que les van a plantear y el nivel de dificultad.

# 2.-Justificación del proyecto:

El objetivo de este proyecto es desarrollar un sistema capaz de poder registrar, listar, modificar y eliminar exámenes.

Con este sistema el profesor ahorrará tiempo a la hora de confeccionar los exámenes, ya que lo tendrá registrados en la aplicación y los podrá modificar y eliminar según crea conveniente.

# 3.-Actores y objetivos:

* + **1- Profesorado del Guadalupe**

El objetivo principal de este proyecto es programar una aplicación para el profesorado que facilite el hacer exámenes para los alumnos. El sistema recogerá las preguntas que el profesor preparará para el examen y se guardarán en la aplicación, así cuando tenga que poner otro examen solo tiene que elegir las preguntas que ya ha guardado la aplicación, ahorrándole tiempo y esfuerzo.

* + **2-Fundación Loyola**

A la Fundación Loyola le interesa que sus profesores tengan menos carga de trabajo y puedan rendir mejor en su trabajo, lo cual favorecerá el cumplimiento de los objetivos de la fundación con sus alumnos para sacar los profesionales mejor cualificados.

* + **3-Tribunal del proyecto.**

Este proyecto también debe convencer al tribunal al que vamos a presentar el proyecto. Todo lo que se exponga en día de la presentación debe estar ensayado para que la presentación sea amena y concisa para el tribunal, que no tiene por qué saber del tema que estamos exponiendo.

* + **4-Miguel Jaque**

Nuestro primer objetivo será crear un proyecto que le guste al tutor del proyecto, que en este caso es Miguel Jaque. El proyecto debe darle una muy buena primera impresión como siempre ha recalcado en sus clases y debe ser lo más simple posible.

# 4.-Plan de proyecto:

**Sprint 1 -> 06/04/2020- 12/04/2020**

1. **Reunión de Lanzamiento** del proyecto con el tutor.
2. Puesta en marcha del **Diario del Proyecto**.
3. Elaboración del documento “**Listado de Actores y Objetivos**”.
4. Registro de requerimientos (Funcionales, Normativos, Ergonómicos y de Seguridad) en el documento “**Listado de Requerimientos**”.

* Entregables
  + Listado de actores y objetivos
  + Listado de requerimientos

**Sprint 2-> 20/04/2020- 26/04/2020**

1. Reunión de Seguimiento con el tutor.
2. Desarrollo del **Diario del Proyecto**.(en proceso)
3. **Estudio de Aplicaciones Similares**.(casi completado)
4. Elaboración de la **Guía de Estilo** (condicionada por el Listado de Actores y Objetivos).(en proceso)
5. Elaboración del **Plan de Proyecto** (condicionada por el Listado de Requerimientos).
6. Registro de Casos de Uso (procesos) en el “**Listado de Requerimientos**”.

* Entregables:
  + Guía de estilos
  + Plan de proyecto

**Sprint 3-> 27/04/2020- 03/05/2020**

1. Reunión de Seguimiento con el tutor.
2. Desarrollo del **Diario del Proyecto**.
3. Elaboración del **Esquema de Navegación** (condicionado por los **Casos de Uso**).
4. Elaboración de la **Arquitectura Tecnoloǵica** (Módulos, frameworks, SGBD, patrones…)
5. Creación del Entorno de Programación(instalación mongo, codeigniter).
6. Creación del Entorno de Despliegue.

* Entregables
  + Esquema navegación
  + Arquitectura tecnológica
  + Libro de estilos

**Sprint 4-> 04/05/2020- 10/05/2020**

Elegir los Requerimientos que se incluirán en el Sprint. Para ellos:

1. Desarrollar el **Modelo de Datos** (condicionado por Casos de Uso y Bocetos de Páginas)

Casos de uso:

* Desarrollo de **CU2 - Login Google**
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N). (Aprobación)
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 5-> 11/05/2020- 17/05/2020**

Casos de uso:

* Desarrollo de **CU3.1- Alta examen sencillo y CU3.2-1-Mostrar listado exámenes**
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N). (Aprobación)
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 6-> 18/05/2020- 24/05/2020**

Casos de uso:

* Desarrollo de  **Creación estructura alta preguntas para estos casos de uso->CU3.4.2.1.1-CU3.4.2.1.1 , CU3.3-Eliminación exámenes ,CU8-Modificación datos generales examen**
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 7->01/06/2020-07/06/2020**

* Desarrollo de **CU3.4.2.1.1-CU3.4.2.1.2 - Alta preguntas de tipo texto corto y Alta de preguntas de tipo texto largo**
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 8-> 14/09/2020- 20/09/2020**

* **Retomado el proyecto**
* Desarrollo de **CU3.2.1-CU3-2.4 - Alta preguntas de tipo texto corto y Alta de preguntas de tipo texto largo mandando el examen(json)**
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 9-> 21/09/2020- 27/09/2020**

* Desarrollo de **CU3.4.2.1.3.-CU3.4.2.1. - Alta preguntas de tipo test y respuesta múltiple**
* Debug del CU2-Inicio sesión
* Objeto examen creado y gestionado correctamente
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 10-> 28/09/2020- 04/10/2020**

* Desarrollo de **CU3.2.1-CU3-2.4 - Alta preguntas de tipo test y Alta de preguntas de tipo respuesta múltiple ,**
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 11-> 05/10/2020- 11/10/2020**

* Desarrollo de **F3.4.2 completa**
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 12-> 12/10/2020- 18/10/2020**

* Búsqueda y solución de bugs en la aplicación , mejorado diseño de la aplicación
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 13-> 19/10/2020- 25/10/2020**

* Mejora estilos aplicación
* Investigación PHPDoc y JSDoc
* Listado de requerimientos mejorado y  casi completado
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 14-> 26/10/2020- 1/11/2020**

* **Desarrollo CU1-Proceso instalación**
* Depuración de código en la aplicación y uso de plantillas para ahorrar código
* Comprobada accesibilidad de las páginas
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 15-> 2/11/2020- 8/11/2020**

* Uso de PHPDoc y JSDoc en el código del programa
* Manual del programador empezado
* Corrección bugs
  + Elaborar **Boceto de las páginas, solo las del caso de uso de la semana.**
  + Elaborar **Modelo de Datos.**
  + Añadimos los tests al **Listado de Requerimientos.**
  + Codificamos el Modelo Físico de la BBDD (json).
  + Insertamos datos de ejemplo en la BBDD.
  + Codificar las páginas del CU en HTML (hacer el Mock en B/N).
  + Codificamos la lógica del CU (selenium).
  + Codificamos los tests.
  + Probamos los tests.
  + Añadir estilo (Guía de Estilos), el CSS.
  + Commit en el git (src(html + js + css, json (MongoDB), test(selenium)) .

**Sprint 16-> 9/11/2020- 15/11/2020**

* Mejorado el feedback del programa
* Cambios sobre el **CU3.4.1**
* Cambios en la iconografía de la página
* Acabados manual del programador e instalación
* Cambio de diseño del login
* Mejora listado requerimientos
* Plan pruebas completado

**Sprint 17-> 16/11/2020- 22/11/2020**

* Fixes rutas absolutas aplicación
* Completados test selenium

**Sprint 18-> 23/11/2020- 29/11/2020**

* Memoria aplicación

# 5.-Documentación de la aplicación:

## 5.1.-Listado de Requerimientos

****

## 5.2-Arquitectura tecnológica:

Arquitectura Cliente/Servidor

El proyecto utilizará arquitectura Cliente/Servidor en tres niveles: cliente, servidor y base de datos.

Nivel de Cliente

Navegador: HTML5, CSS3, ECMA6

Modelo: Rich Client (one page)

POO: Para asegurar la modularidad del proyecto, usaremos programación orientada a objetos

Librerías: Bootstrap,JQuery

Comunicación-Servidor: AJAX (POST/GET),JSON

Nivel de Servidor

PHP versión 7.3.9

POO: Para asegurar la modularidad del proyecto, usaremos programación orientada a objetos

Framework: Codeigniter

Librerías: libmongoc y libbson (librerias de mongo sobre PHP).

Comunicación-BBDD: TCP/IP (local)

**Nivel de Base de Datos**

mongoDB 4.2

Entorno desarrollo:selenium

**Diagrama de Despliegue**

Cliente

Navegadores compatibles:

* Microsoft Edge
* Firefox
* Google Chrome

**SO:**

* Windows

**Resoluciones:**

* 1920 x 1080
* 1024 x 728
* 375  x 667

Servidor (pendiente)

Sistema Operativo

Servidor Web

Base de Datos

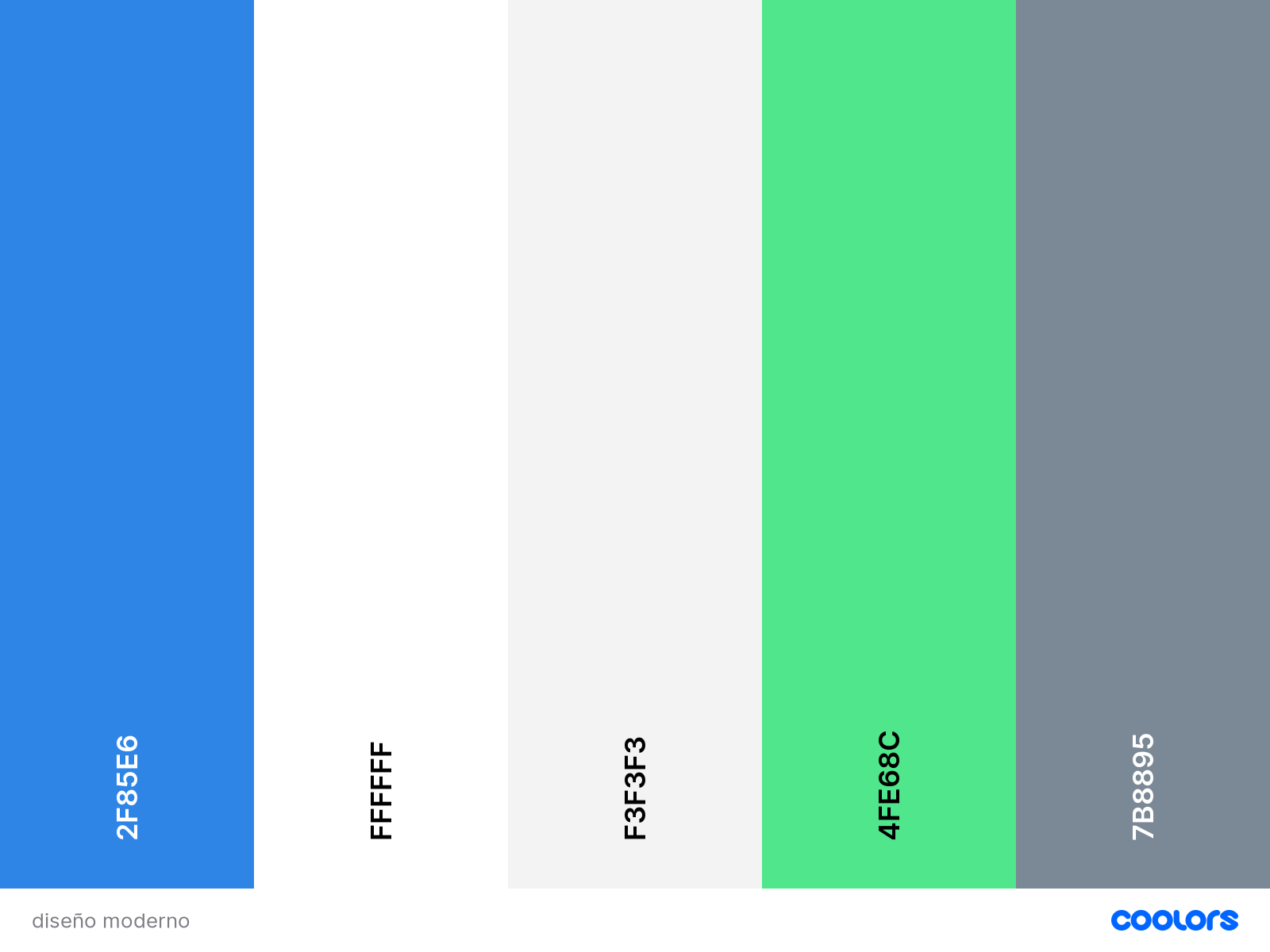
Entorno Desarrollo

Editor código:PhpStorm

Usaremos Selenium para la automatización de los test

## 5.3.-Guia de estilo:

Paleta de colores



Fuentes Utilizadas

Fuente para texto plano:**Monserrat.**



Iconografía

BORRAR EXAMEN

EDITAR EXAMEN 

EXAMENES 

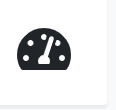
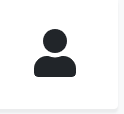
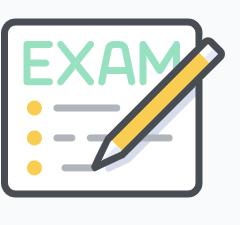
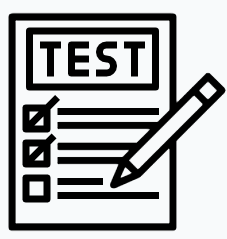
DASHBOARD 

IMAGEN POR DEFECTO 

CREAR EXAMEN 

LISTA EXAMENES 

## 5.4-Estudio de aplicaciones similares:

CLASSMAKER

Idiomas:

Por defecto, CLASSMAKER está en inglés, y aunque se le puede cambiar el idioma a la interfaz nuestro programa estará por defecto en español ya que nuestros profesores no tienen por qué saber inglés ni perder tiempo cambiando esta opción.

Aplicación freemium:

Muchas de las opciones interesantes de la aplicación son de pago, como subir audio, video, imágenes y varias opciones de seguridad importantes. En nuestra aplicación no omitiremos ninguna de estas opciones y se podrán usar todas las opciones desde el principio.

Difícil acceso a la aplicación:

El acceso a la aplicación es mediante un registro de usuario con el que se tardan varios minutos. En nuestra aplicación tendremos un acceso por cuenta de Google con el que nuestros profesores no tardarán en acceder.

EASY LMS

Aplicación de pago:

EASY LMS es una aplicación de pago en la que ni siquiera puedes pedir una prueba de unos días y te exige elegir un tipo de financiación con distintos tipos de ventajas dependiendo del plan que elijas.

GOOGLE FORMS

GOOGLE no permite barajar las preguntas:

A la hora de crear un examen, Google Forms no te permite mezclar el orden de las preguntas, por lo que si el examen tiene 20 preguntas a los alumnos les saldrán 20 preguntas en el mismo orden, lo cual puede facilitar que los alumnos compartan las preguntas entre si y se pueda copiar más fácilmente.

No se puede elegir el número de preguntas:

Al contrario que las dos aplicaciones anteriores, Google Forms no te deja seleccionar un número de preguntas concreto de tu banco de preguntas, y debes ir seleccionándolas una a una.

## 5.5.-Libro de estilo:

### 5.5.1.-Normas Generales

1. **Se evitará mezclar distintos lenguajes en un mismos fichero**. Por ejemplo, se evitará incluir código CSS o JavaScript en páginas HTML y también se separará el código PHP del HTML y del SQL.
2. **Los nombres utilizados para ficheros, variables, clases, etc, serán en castellano y descriptivos evitando el uso de acentos, ñ, diéresis y espacios**. En general, se utilizarán caracteres de ASCII básico. Se admite el uso de abreviaturas habituales como num, cont, etc.
3. **Todo el código estará adecuadamente comentado**. Se buscará la legibilidad y claridad del código para reducir la necesidad de comentarios.
4. **Todo el código estará indentado**. Se utilizará tabuladores (TAB) a 8 espacios,  sin omitir las llaves opcionales y líneas en blanco. Funciones y clases abrirán con la llave debajo de la declaración , las condiciones abrirán en la misma linea y en el propio cierre de la condición se escribirá la siguiente.

Por ejemplo:

<?php

function foo()

{

    if ($x > 5) {

        echo "bar";

    } else{

        return “hola”;

    }

}

if () {

} else {

    if () {

    } else {

    }

}.

### 5.5.2.-Normas para Ficheros

1. **La cabecera de cada fichero tendrá información para identificar el proyecto y el autor**.
2. **Los nombres de ficheros se escribirán en minúsculas**.
3. **Los ficheros se codificarán en UTF-8**.

### 5.5.3.-Normas para SQL

1. **Las palabras reservadas de SQL se escribirán en mayúsculas** (SELECT, FROM, CREATE TABLE…).
2. **Los nombres de las tablas tendrán todo el nombre en minuscula y serán en singular**. Si el nombre se formará por varias palabras se utilizará notación C (con subrayado bajo). Por ejemplo usuario o usuario\_acceso.
3. **Los nombres de los campos serán en minúsculas**. Si el nombre se formara por varias palabras se utilizará notación C.
4. **Las claves sintéticas de las tablas tendrán el nombre “id”**.
5. **Las claves externas de las tablas que hagan referencia a claves sintéticas se formarán como id\_Nombre\_Tabla**.
6. **Las tablas de relación se llamarán Tabla1\_Tabla2**, ordenando las entidades alfabéticamente: Ejemplo: Alumno\_Curso
7. **Se utilizarán restricciones de claves externas** para asegurar la integridad referencial.

### 5.5.4.-Normas para HTML

1. **Las marcas de html se escribirán en minúsculas**.
2. **El código cumplirá las reglas de validación de XML** (etiquetas correctamentamente cerradas).
3. **Las etiquetas que requieran varias líneas se iniciarán y terminarán con la etiqueta sola en la línea al mismo nivel de tabulación**. Por ejemplo:

<p>

Texto muy largo…

</p>

1. **Los ficheros de estilo se cargarán en la cabecera del documento** (<head>)**.**
2. **Los ficheros de JavaScript se cargarán en al final del cuerpo del documento** (antes de </body>).

### 5.5.5.-Normas para CSS

1. Se simplificará agrupar los estilos a los elementos que mantengan la misma propiedad.

### 5.5.6.-Normas para JavaScript

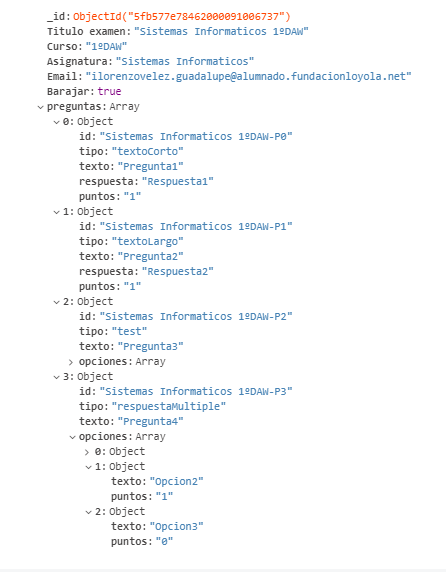
1. **Se utilizará notación camelCase**.
2. **Las clases se nombrarán con la primera letra mayúscula**. Por ejemplo: Usuario, UsuarioAdministracion.
3. **Las variables y objetos se nombrarán con minúsculas**. Por ejemplo: contador, fechaApertura.
4. **Las constantes se escribirán íntegramente en mayúsculas y en notación C**. Por ejemplo: FICHERO, FICHERO\_USUARIOS

### 5.5.7.-Normas para PHP

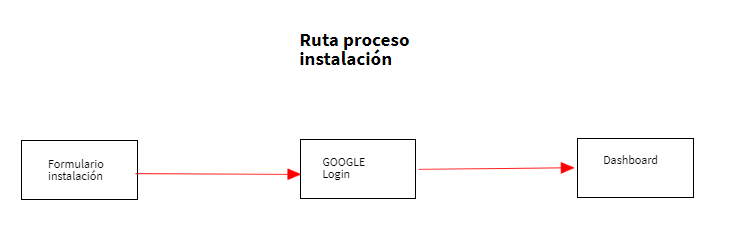
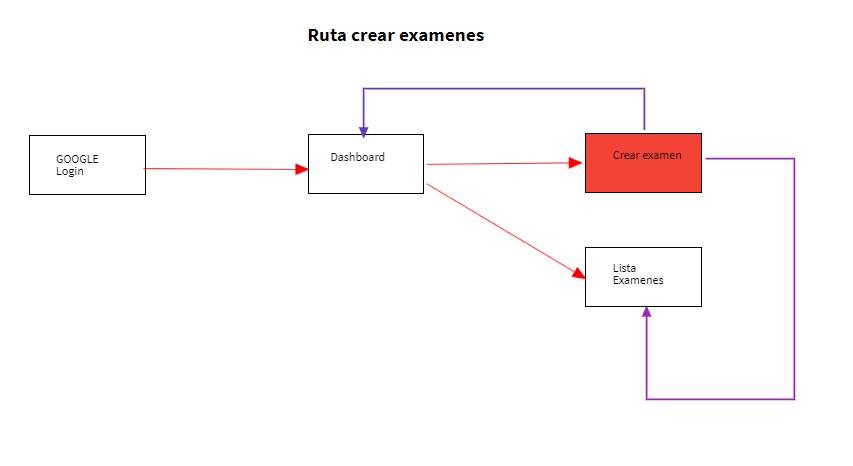
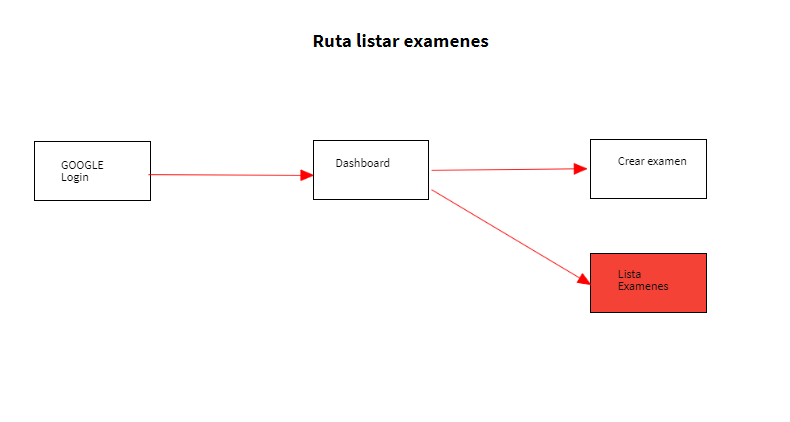
1. **Se utilizará notación C** (con subrayado bajo).
2. **Las clases se nombrarán con la primera letra mayúscula**. Por ejemplo: Usuario, Usuario\_Administracion.
3. **Las variables y objetos se nombrarán con minúsculas**. Por ejemplo: $contador, $fecha\_apertura.
4. **Las constantes se escribirán íntegramente en minusculas**. Por ejemplo: fichero,ficherousuarios.

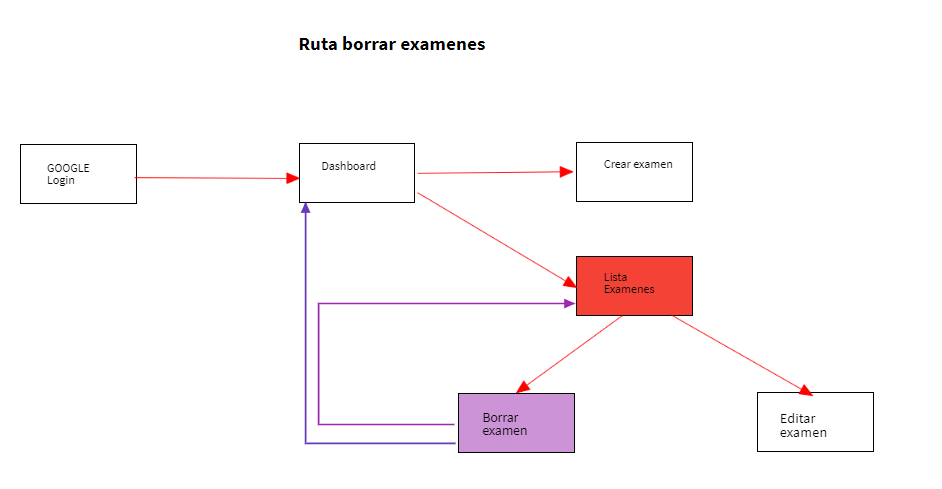
## 5.6.-Modelo de datos:

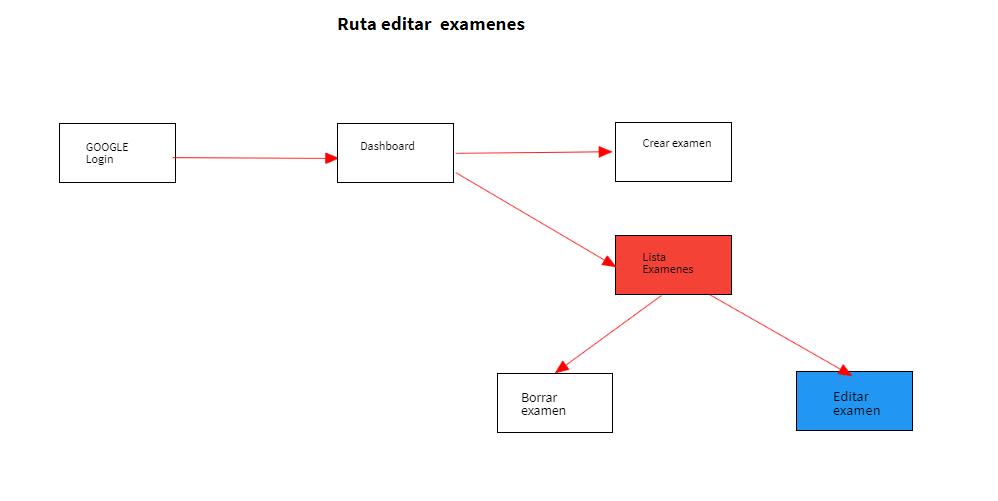
Como estamos trabajando en mongodb, no tendremos E/R, pero tendremos un json por cada examen con una estructura como esta:



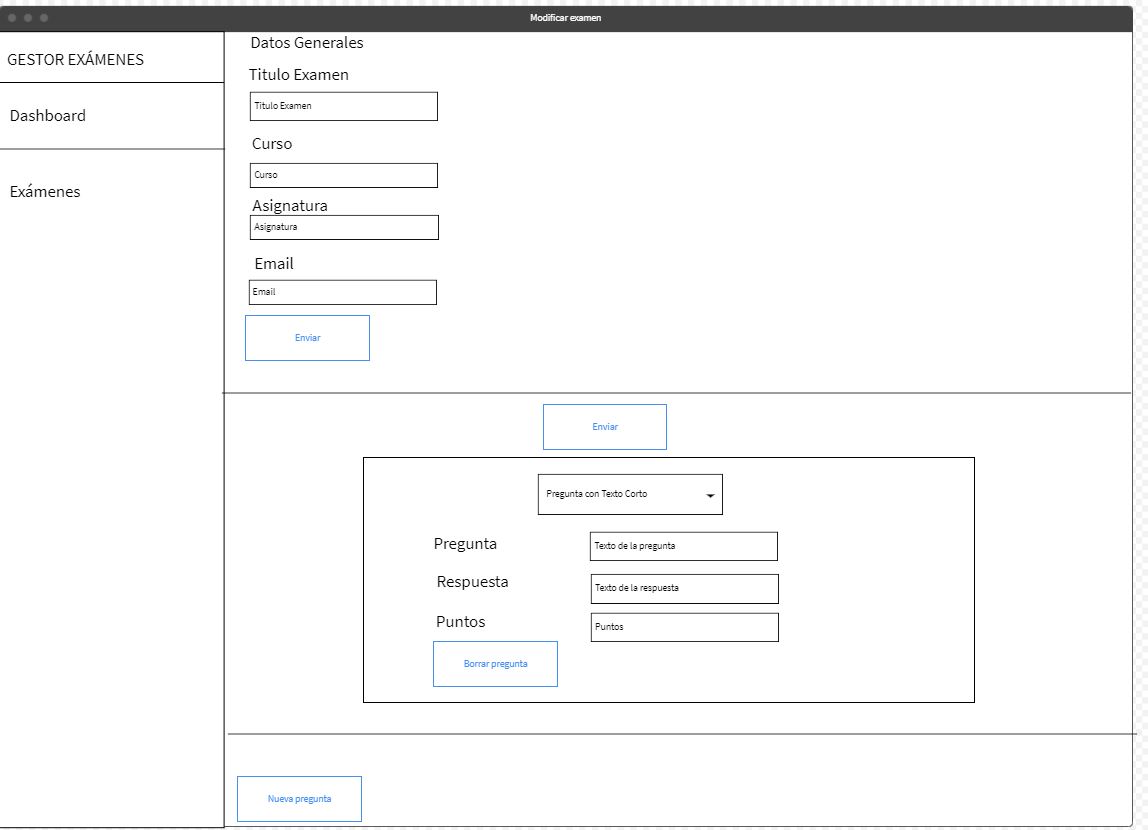
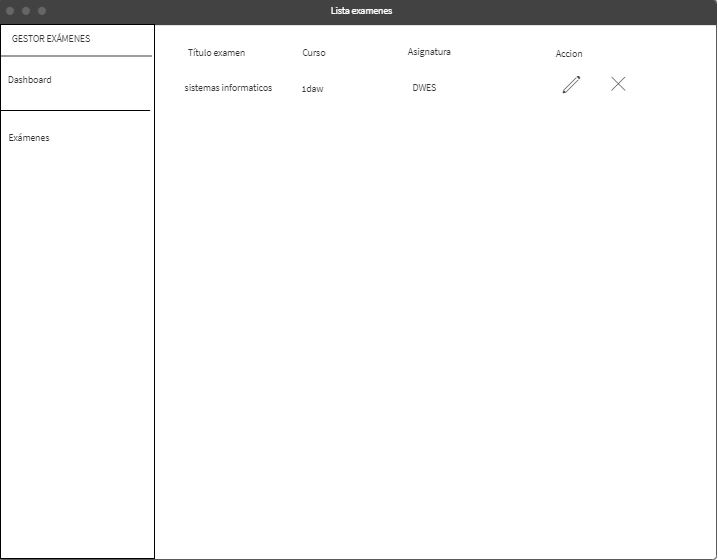
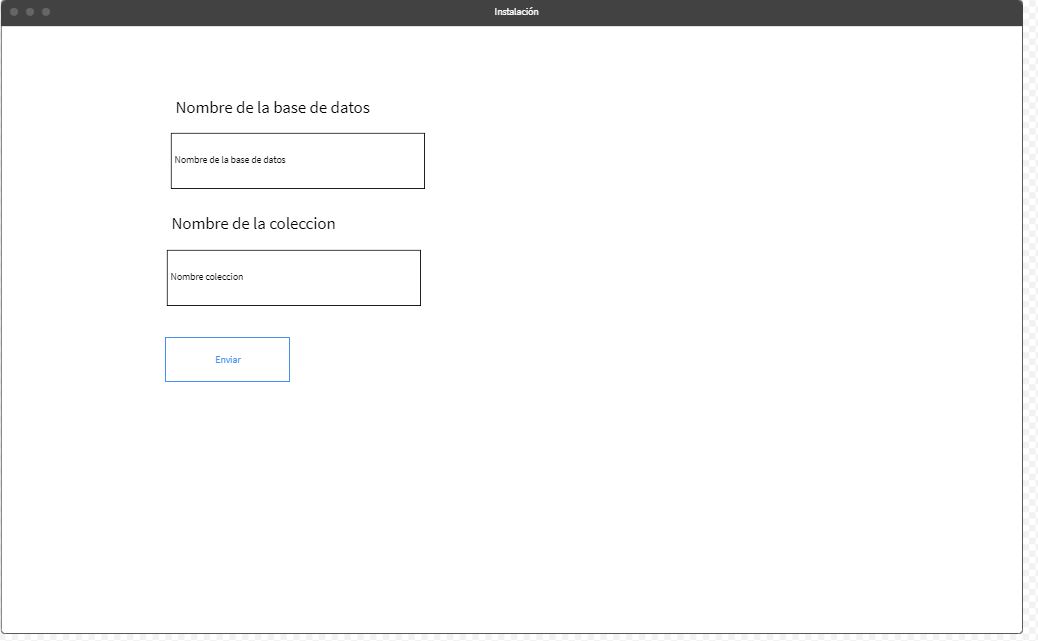
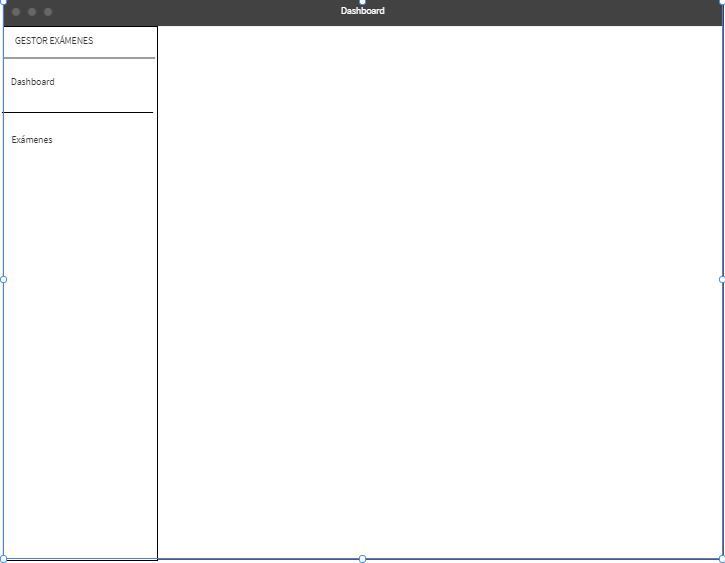
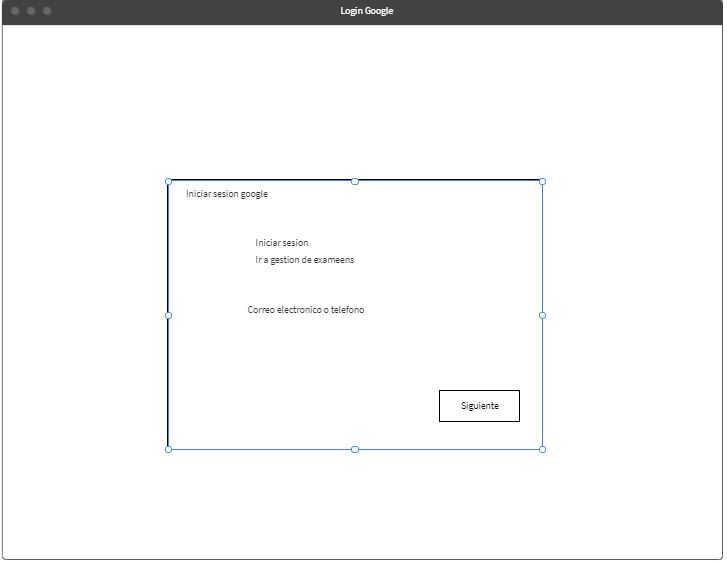
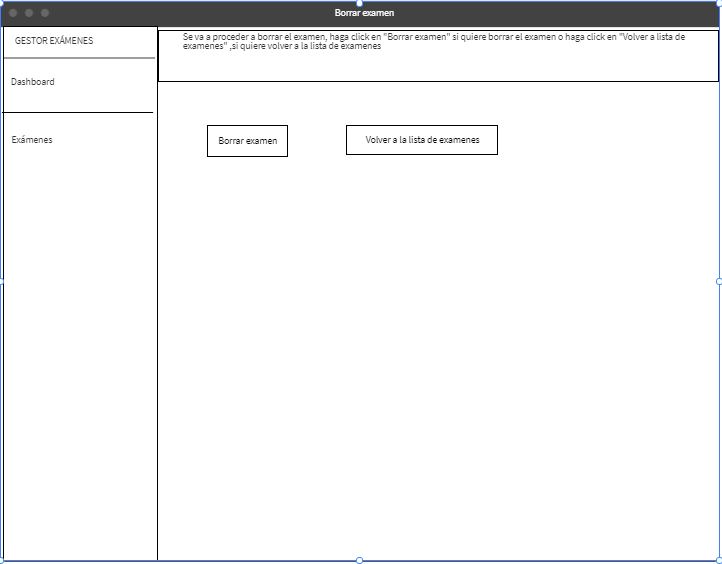
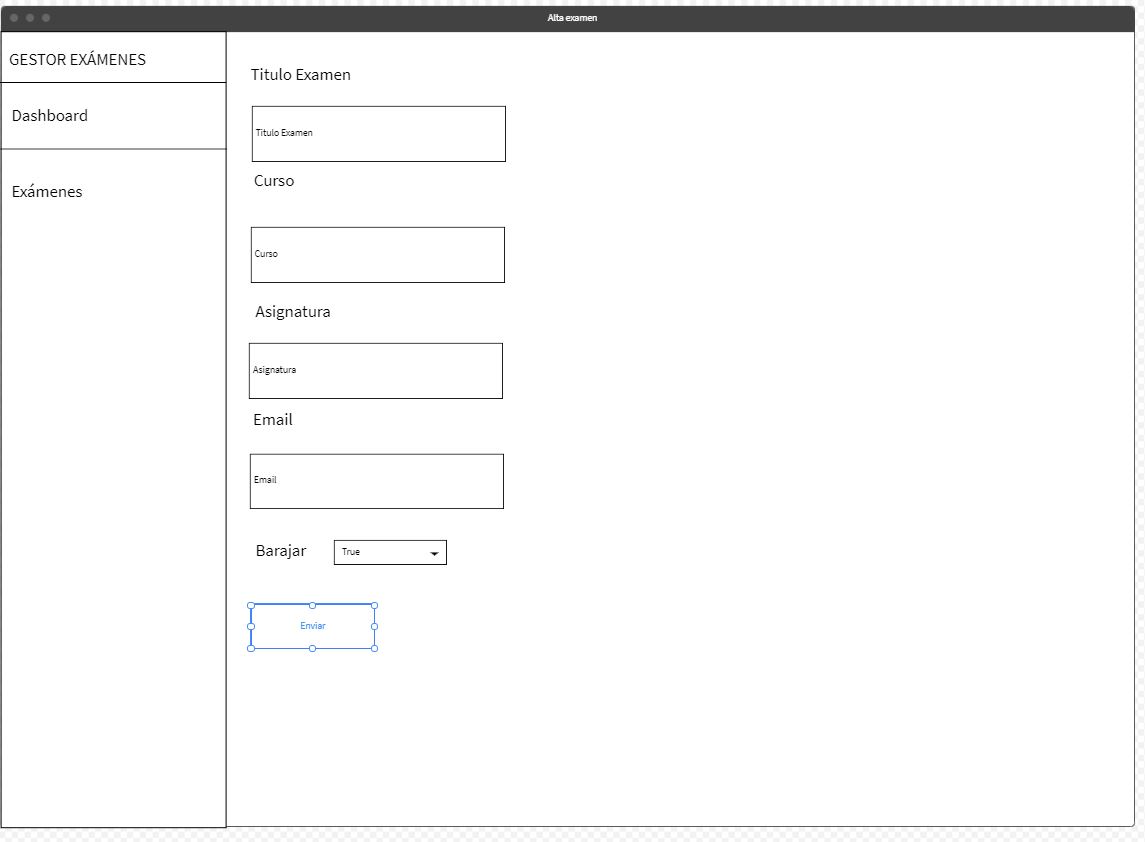
## 5.7.-Mapa de Navegación:



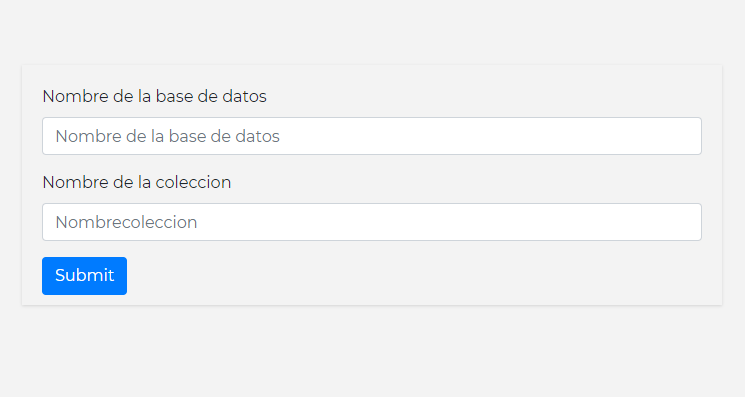


## 5.8.-Bocetos de Páginas:

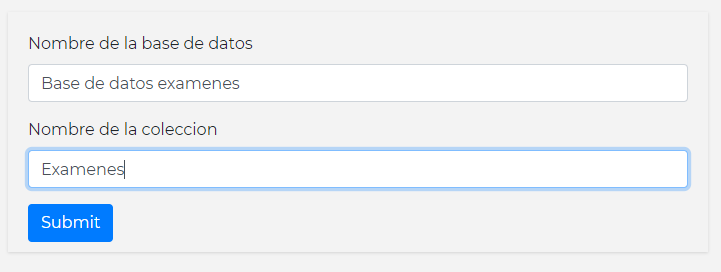


## 5.9.-Manual de instalación:

El primer paso es entrar en <http://localhost/GestorExamenes/> y nos saldrá la siguiente pantalla:



Aquí introduciremos en nombre de la base de datos y en nombre de la colección de esa base de datos



AVISO: En el archivo de configuración de mongodb el nombre de la base de datos no acepta espacios, así que si incluso el usuario pone el nombre con espacios el programa los va a quitar

Ejemplo: Hola mundo hola

Resultado: Holamundohola

Después el programa borrará los archivos de instalación y el usuario no podrá volver a acceder al proceso de instalación

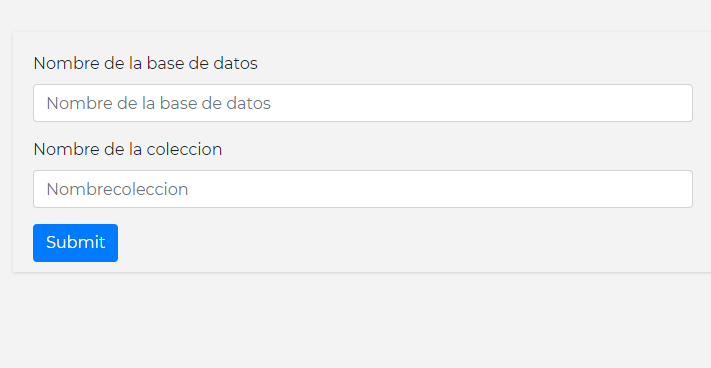
## 5.10.-Manual de usuario:

El usuario entrará a la aplicación y lo primero que le saldrá será el proceso de instalación. Desde aquí mismo probaremos que no podemos entrar a otros controladores por URL, para eso pondremos

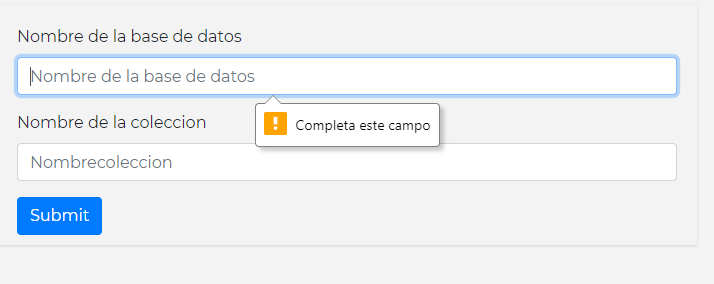
http://localhost/GestorExamenes/Examenes y <http://localhost/GestorExamenes/Dashboard> y comprobamos que no te deja acceder y te redirige al proceso de instalación si la aplicación no se ha instalado aun o al inicio de sesión si se ha instalado la aplicación

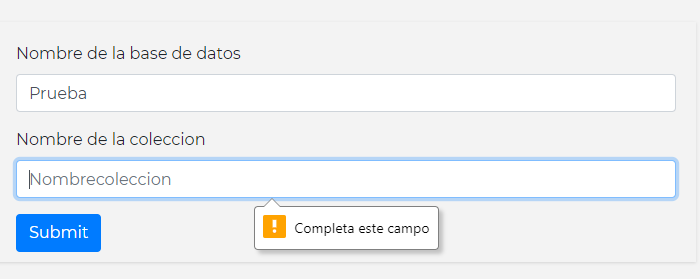
<https://youtu.be/P7IJV3WDOCc>

Una vez comprobado que no se puede acceder por URL, accederemos al proceso de instalación. Para ello pondremos <http://localhost/GestorExamenes/> y nos saldrá la siguiente pantalla

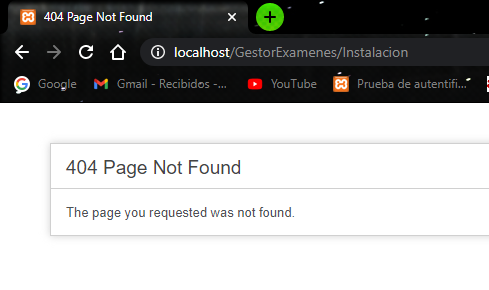


Si intentamos dejar el nombre de la base de datos vacío nos mostrará un mensaje que debemos rellenar el campo. También pasará lo mismo con el nombre de la colección

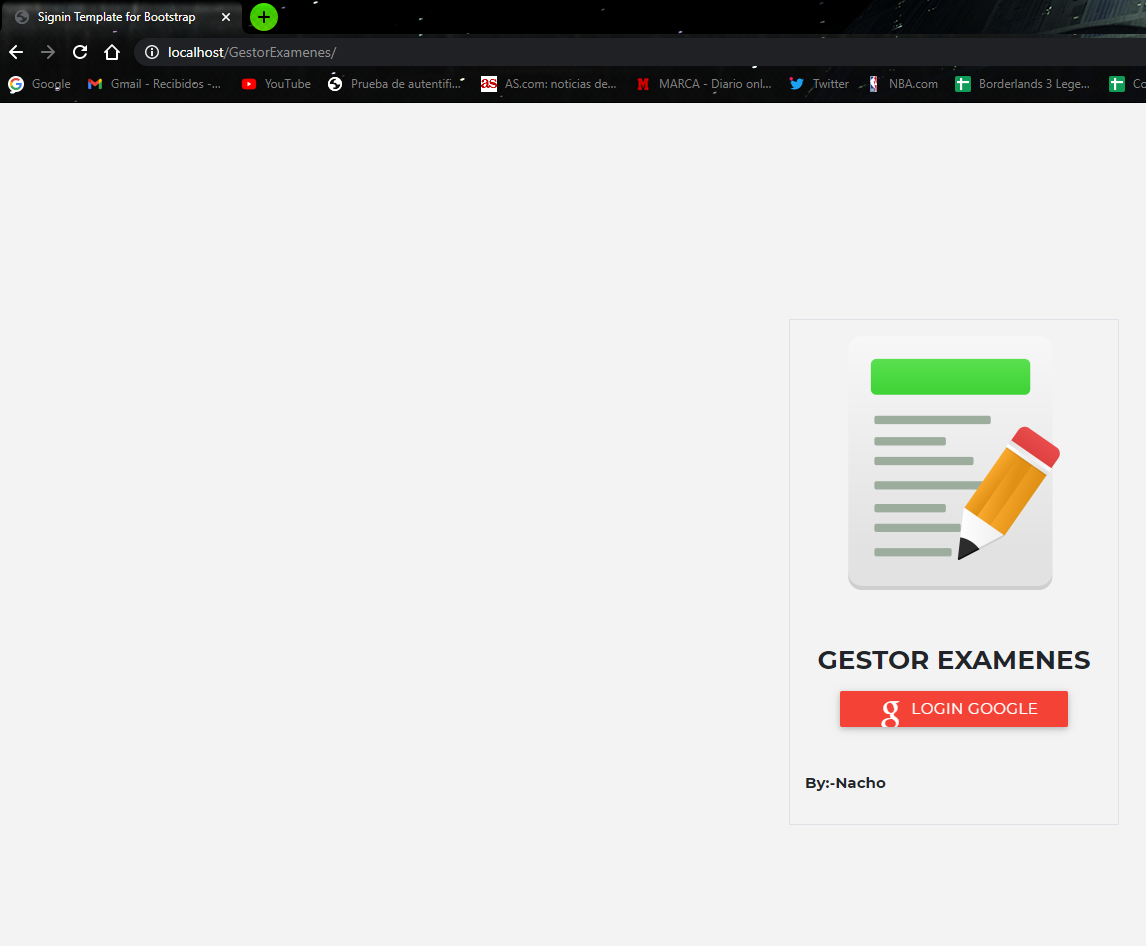




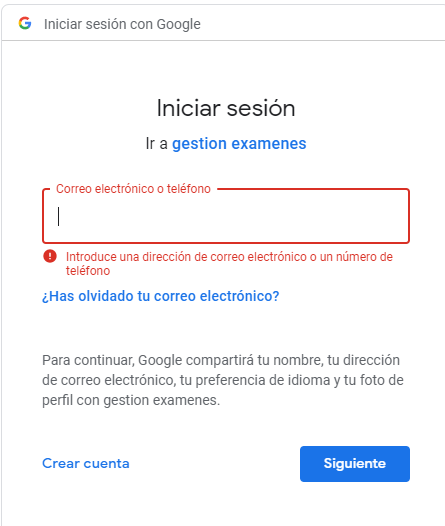
Si ponemos los datos correctos y le damos a enviar nos mandará a la página de autentificación, el programa por detrás nos borrará los archivos de instalación, para comprobar que están borrados teclearemos <http://localhost/GestorExamenes/Instalación> y nos saldrá un error 404

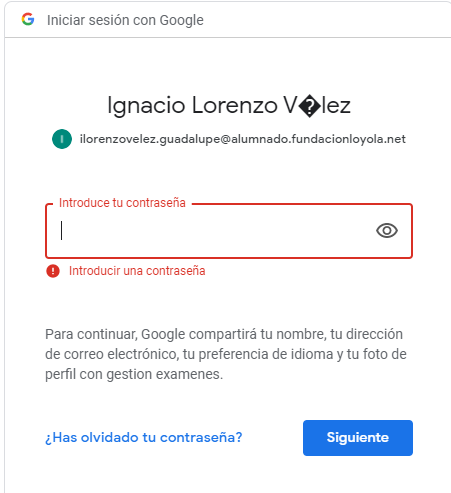


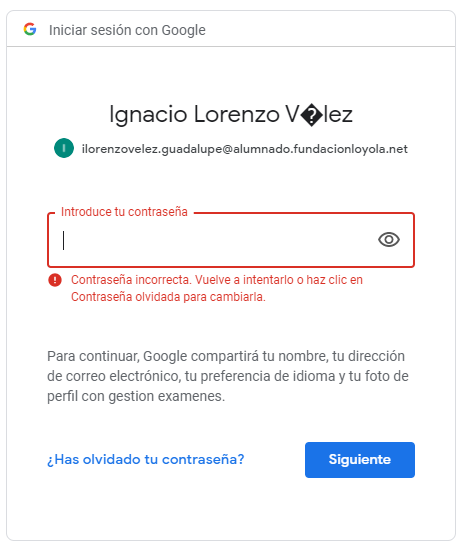
Como hemos borrado el controlador instalación, ahora nuestro controlador por defecto será el de autenticación, lo comprobamos si ponemos <http://localhost/GestorExamenes>



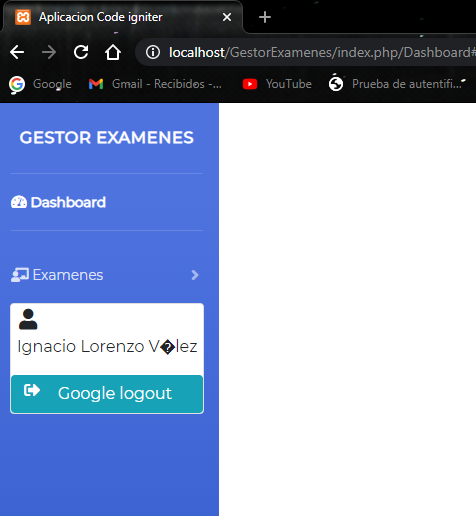
Ahora procederemos a hacer un inicio de sesión con una cuenta de Google, pero antes comprobaremos que no podemos dejar ni el correo ni la contraseña vacía, y la contraseña que tecleamos tiene que coincidir con la contraseña del servidor de Gmail.



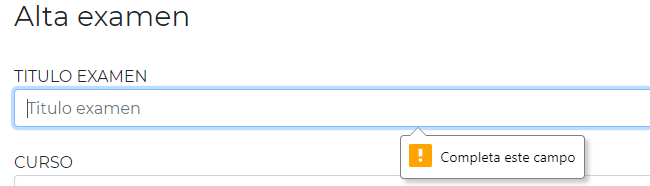


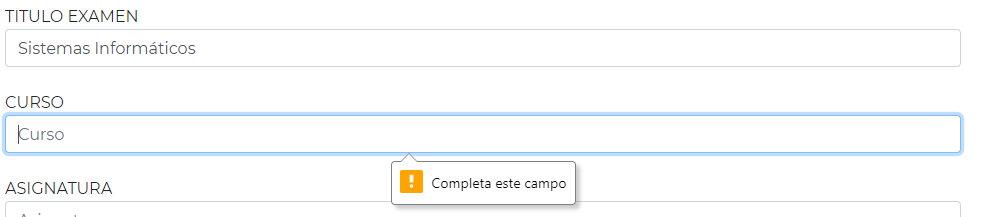


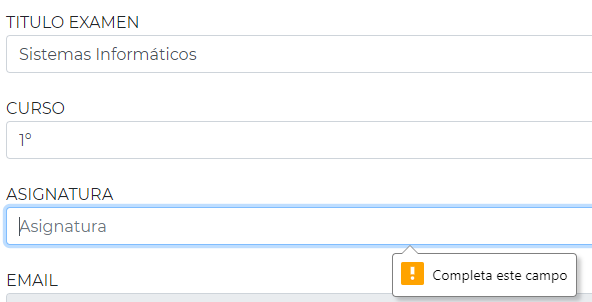
Ahora si ponemos el correo y la contraseña bien nos debería entrar al Dashboard. Si por alguna razón no se carga bien la imagen de Google por defecto se muestra la imagen que se ve en el recorte anterior.



Una vez iniciada la sesión procederemos a crear un examen, para eso nos vamos a Examenes->Crear Examen e iremos dejando los campos de creación del examen vacíos para comprobar que tienes que introducirlos todos. El email no es necesario introducirlo ya que se introduce solo de la sesión de Google.

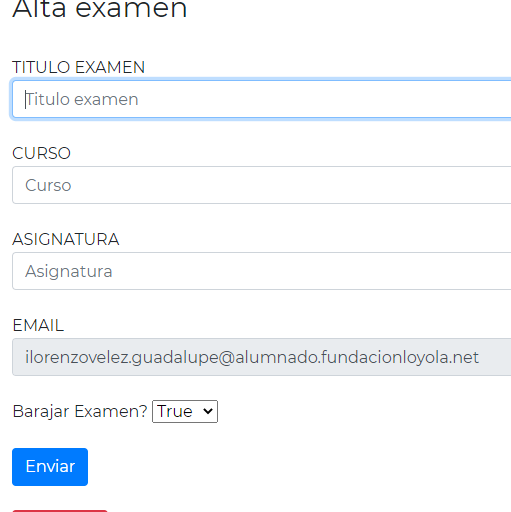


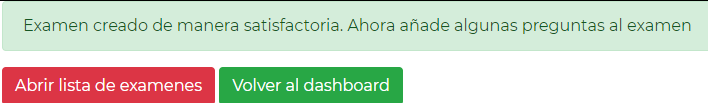




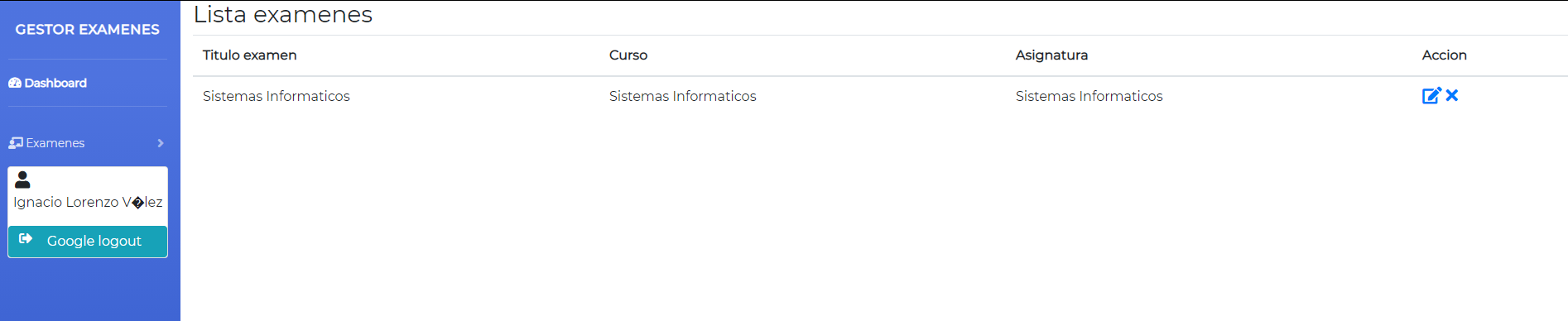
Si a la hora de introducir los datos le damos a cancelar la página nos redirige al Dashboard.

Una vez comprobada esta funcionalidad, volvemos a la pantalla de creación del examen y rellenamos todos los datos correctamente y pulsamos enviar. Nos debería salir un mensaje de éxito a la hora de crear el examen y un par de botones por si quieres volver a crear otro examen o quieres ir a la lista de exámenes.

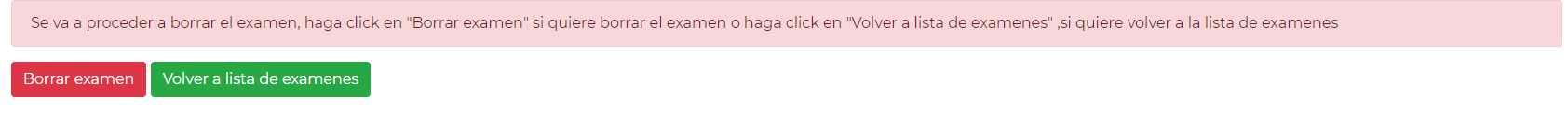




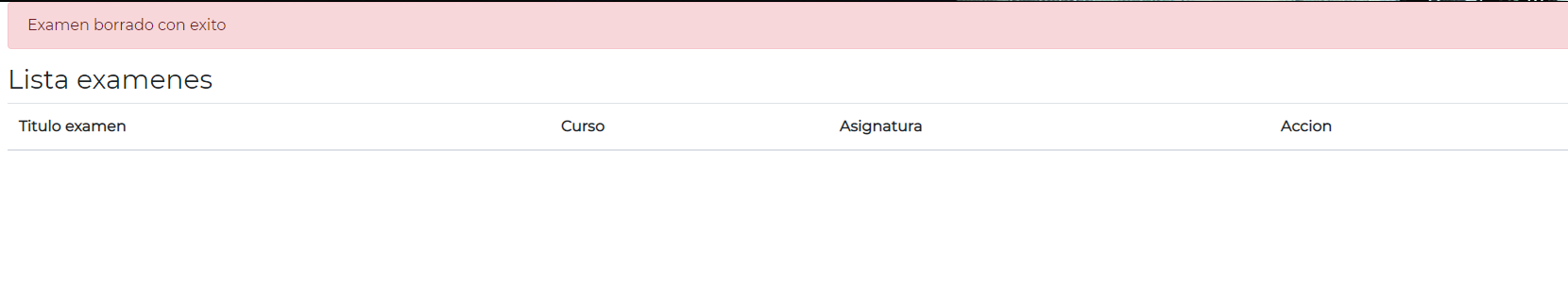
En este caso vamos a pulsar Abrir lista de exámenes y nos mostrará una tabla con los datos del examen y un par de opciones para editarlos o borrarlos.



Lo siguiente será probar el borrado del examen, para eso sobre la lista de exámenes donde estábamos antes pulsamos el botón con la X y nos saldrá una ventana de confirmación para ver si queremos borrarlo. Esta vez pulsaremos volver a la lista de exámenes y nos devolverá a la lista anterior.

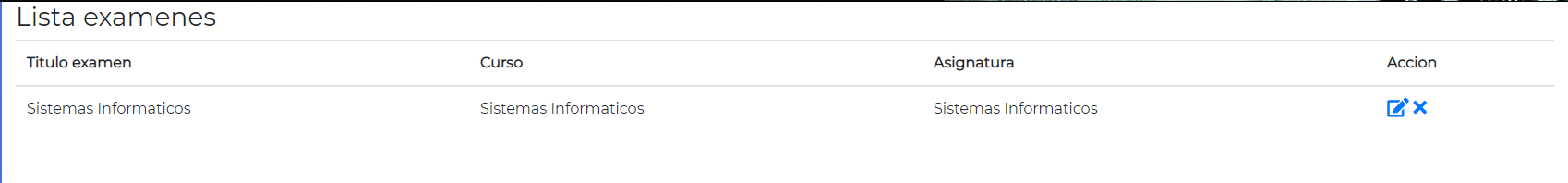


Una vez en la lista de exámenes pulsaremos esta vez la X y borrar examen y nos devuelve a la lista de exámenes con un mensaje de “Examen borrado con éxito”



Para seguir con las siguientes pruebas volveremos a crear el examen con los mismos datos de antes.

Una vez creado el examen vamos a probar el editarlo. Para ello vamos a hacer click en el icono del lápiz



En la siguiente pantalla vamos a proceder a cambiar los datos generales del examen, para ello introduciremos los siguientes datos para modificar el examen y haremos click en enviar:

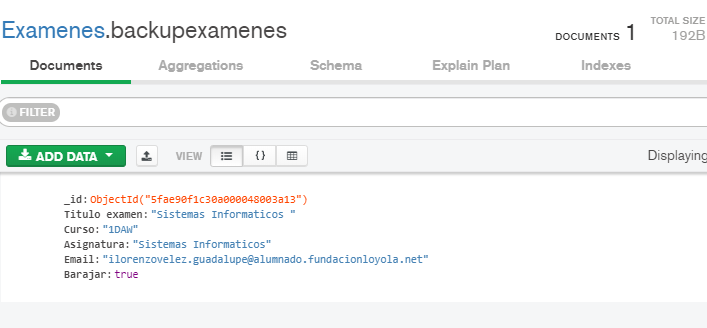
Titulo Examen: Sistemas Informáticos 1ºDAW

Curso:1ºDAW

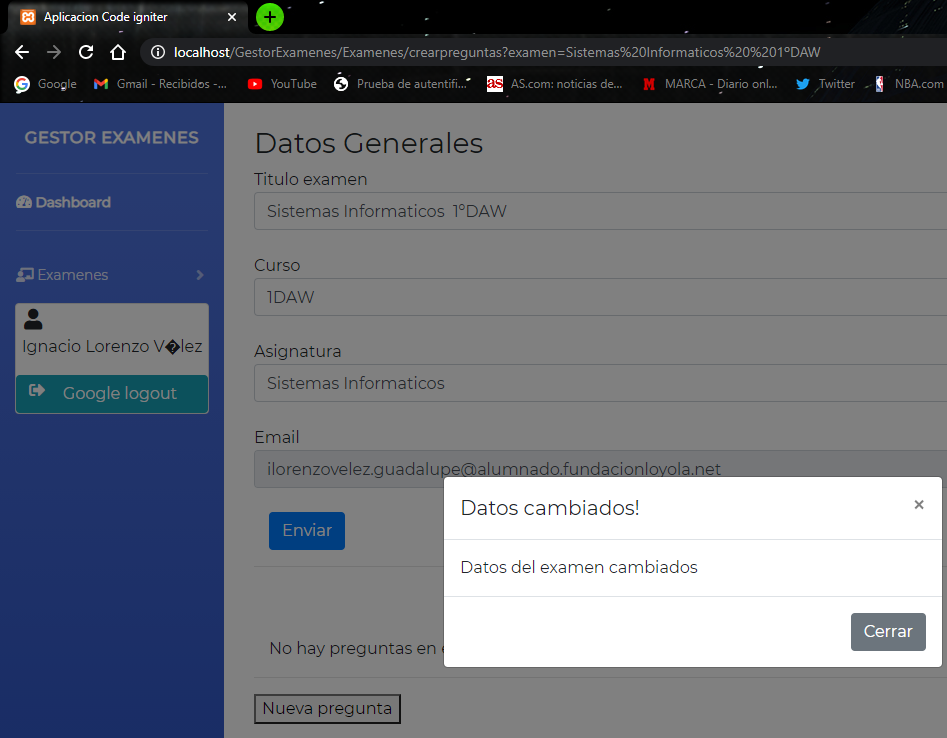
Asignatura:Sistemas Informaticos

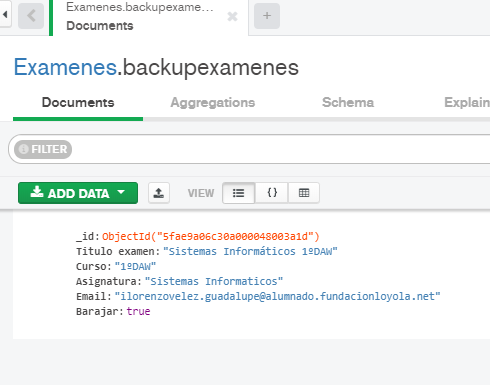
Barajar : True

Cuando le demos a enviar nos saltará una ventana con un mensaje de éxito, en la URL ha cambiado el nombre antiguo del examen por el nuevo sin actualizar la página. También comprobaremos en el gestor de mongo que el examen se ha cambiado bien

Antes: 

Después:



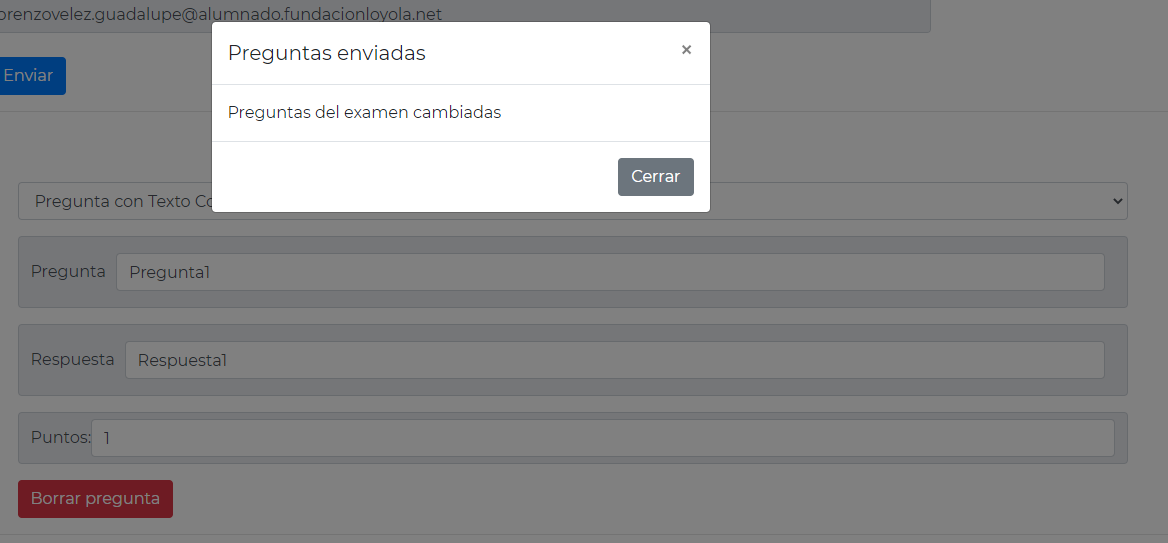


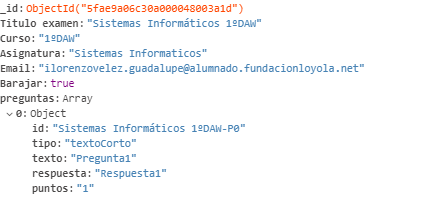
Ahora vamos a probar a introducir preguntas al examen, le damos a nueva pregunta y vemos que nos genera un bloque con la pregunta, la respuesta y los puntos. Si le damos a borrar pregunta nos borrará el bloque. Para hacer la primera prueba introduciremos los siguientes datos y nos saltará un mensaje de que se han cambiado las preguntas:

Pregunta: Pregunta1

Respuesta: Respuesta1

Puntos: 1



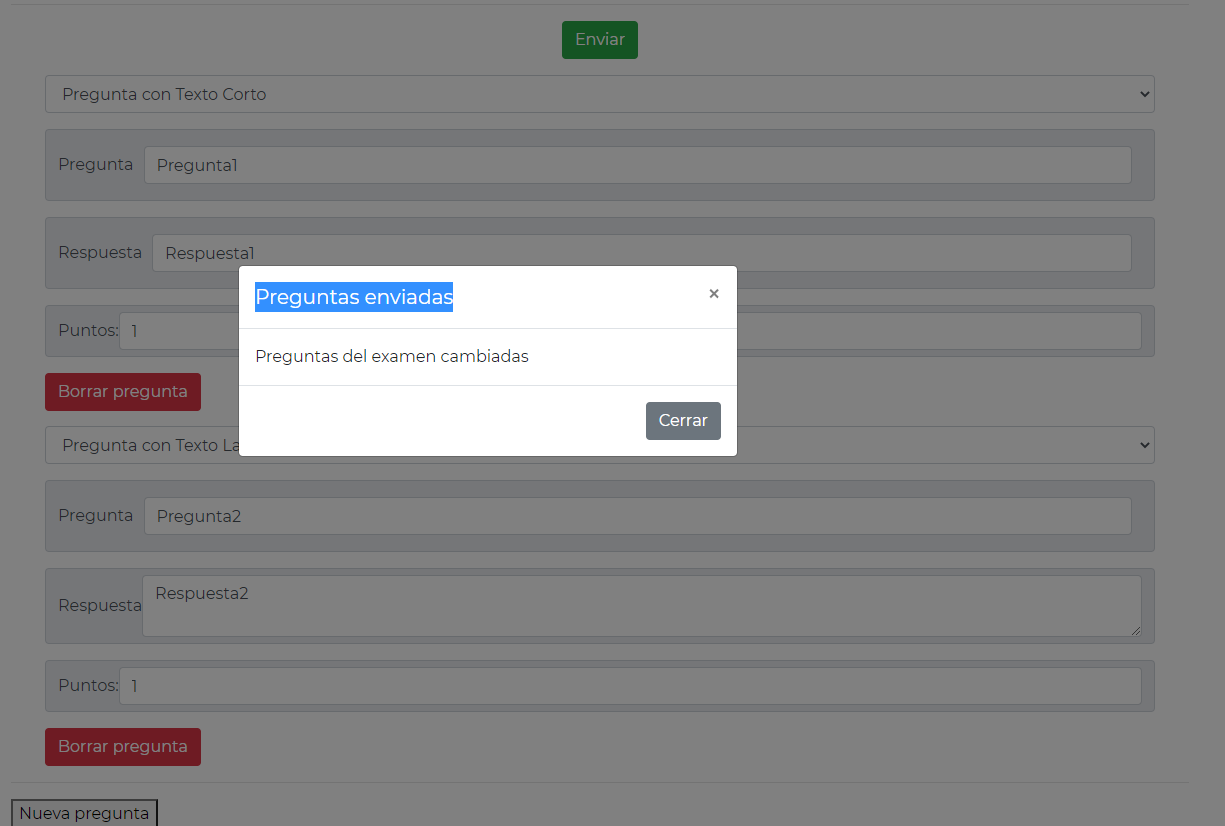


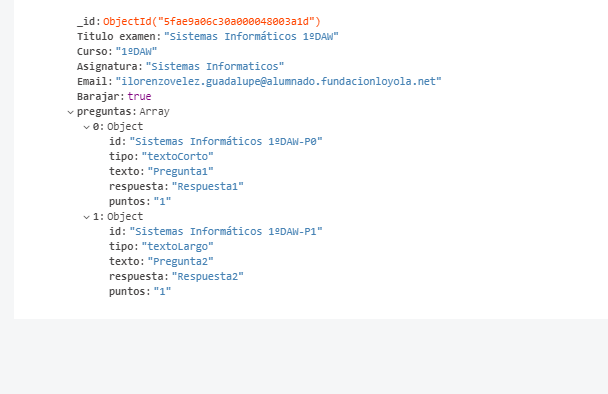
Ahora haremos el mismo proceso pero con una pregunta de texto largo, le damos a nueva pregunta y seleccionamos de las opciones “pregunta con texto largo” y nos generará otro bloque debajo del anterior. Si le diéramos a borrar pregunta nos borrará el bloque que hemos creado ahora. Para este ejemplo introduciremos:

Pregunta: Pregunta2

Respuesta: Respuesta2

Puntos: 1





Para la pregunta tipo test haremos lo mismo, haremos click en nueva pregunta y nos generará otro bloque después del anterior, pero para este tipo de preguntas podemos tener varias opciones como respuestas. Podemos generar las opciones que queramos si hacemos click en “nueva opción”, por defecto se genera solo una y podremos borrarla si hacemos click en “borrar opción” y también podremos borrar la pregunta entera si hacemos click en “borrar pregunta”. Los datos que usaremos para este ejemplo serán:

Pregunta: Pregunta3

Opción: Opcion1

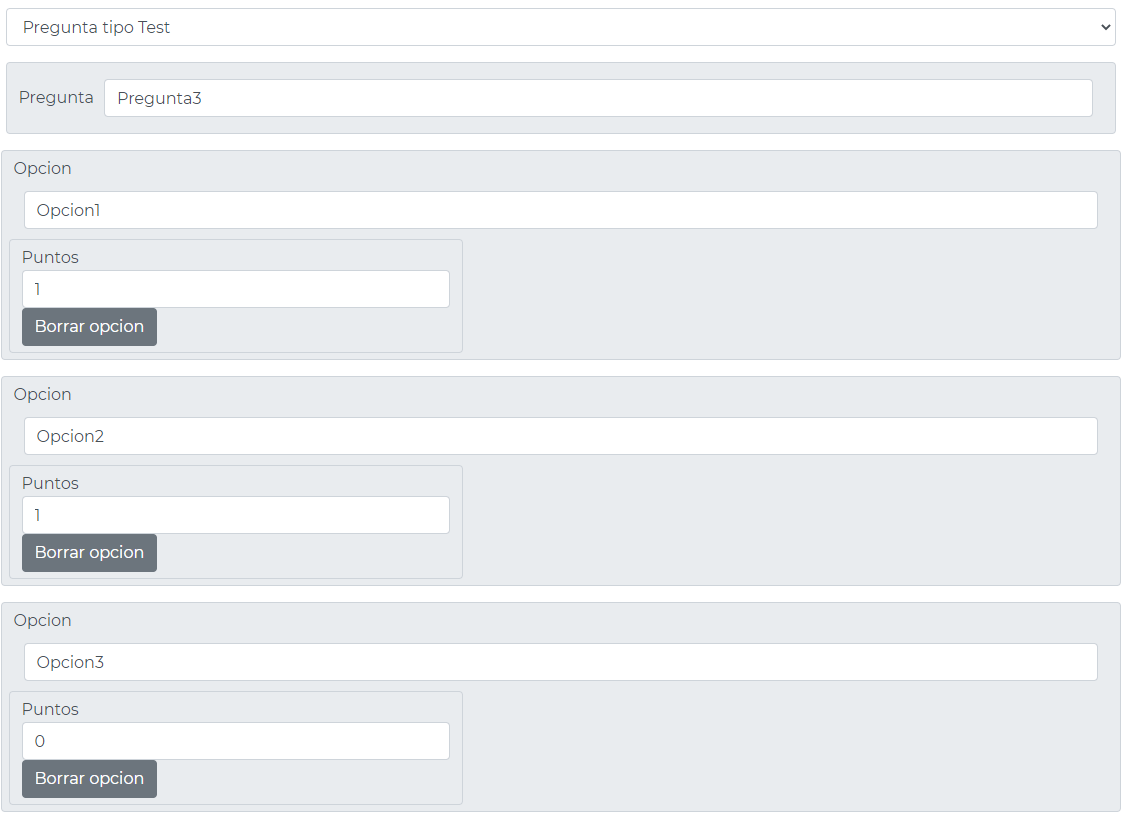
Puntos: 1

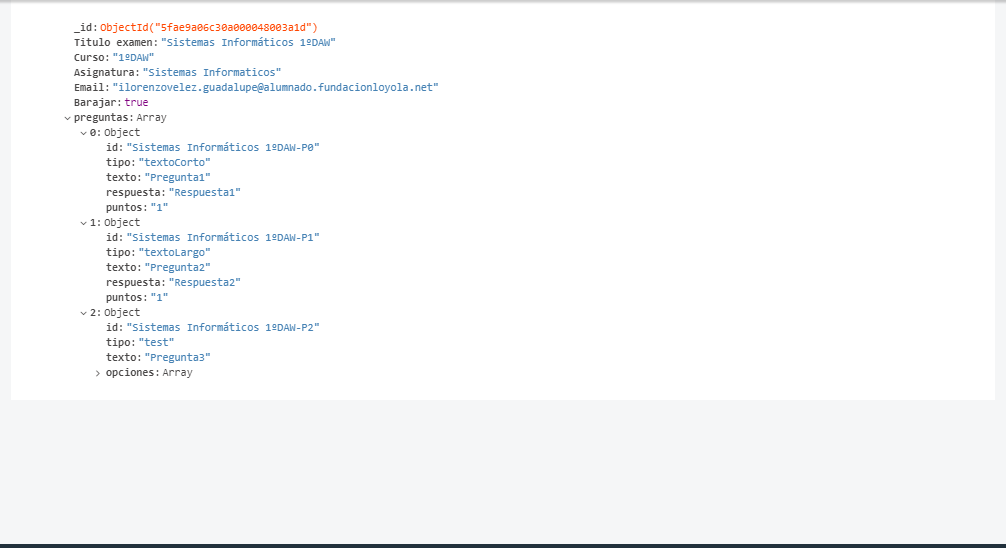
Opción: Opcion2

Puntos: 1

Opción: Opcion3

Puntos: 0





Por último, la pregunta de tipo respuesta múltiple funciona igual que la de tipo test, podremos agregar opciones, borrar opciones, y borrar la pregunta si lo deseamos. Para este último ejemplo usaremos los siguientes datos:

Pregunta: Pregunta4

Opción: Opcion1

Puntos: 1

Opción: Opcion2

Puntos: 1

Opción: Opcion3

Puntos: 0

## 5.11-Manual del programador

Esto lo agregaré en otro momento por si se introducen mas cosas al codigo

## 5.12-Plan de pruebas



# 6.-Planes de mejora:

* La paginación en la zona de incidencias.
* Identificar la ubicación en un google maps ilustrado.
* Dar opción a recuperar contraseña.
* Gestión de comentarios.

# 7.-Tecnología usada.

Como novedad en este proyecto se ha usado MongoDb como base de datos no relacional, ya que los datos que se almacenan no dependen de otras tablas y se pueden guardar como objeto(JSON).

Para el desarrollo del proyecto se ha optado por el framework Codeigniter3, una opción más cómoda ya que estoy familiarizado con la tecnología.

El backend se ha desarrollado con PHP y el front ha sido una combinación de HTML+CSS+JS+AJAX+JQUERY usando la librería de fontawesome para la iconografía de la web

# 9-Estudio de seguridad

Se han tomado tres medidas de seguridad en la aplicación:

1. Control de acceso a usuarios por URL
2. Controlado inyección código SQL
3. Filtrado de inputs de la aplicación

Para el control de acceso a usuarios se comprueba el token de la sesión de Google en cada controlador, si no estás logueado el sistema te redirige al login.

El segundo punto no nos afecta en nuestra aplicación ya que usamos MongoDb como base de datos y no SQL

Para el tercer punto se han usado los métodos **filter\_input** y **strip\_tags** de PHP.

# 9-Estudio de Accesibilidad de la Aplicación

Se ha utilizado la siguiente página para el estudio de accesibilidad de la aplicación, pegando el código HTML porque la página no acepta los archivos .php y usando el filtro WCAG 2.0 AAA corrigiendo todos los posibles fallos de accesibilidad de la aplicación.

<https://achecker.ca/checker/index.php>

# 8.-Problemas encontrados.

Este proyecto ha tenido dos grandes dificultades:

1. Investigación de mongo.

Al ser una tecnología nueva ha sido necesario mucha formación para entender cómo funciona. Una vez familiarizado con Mongo había que descargar una librería de Mongo para Codeigniter.

El problema con esto fue que no había manera de saber la consulta que estaba mandando el framework, cosa que no pasa con MySQL. Al estar lanzando las consultas a ciegas ha sido necesario mucho tiempo para saber lo que estaban haciendo las consultas, ya que venía muy poca información en la librería sobre cómo funcionan los métodos.

1. El FRONT de la aplicación

La otra gran dificultad ha sido gestionar la modificación del examen. Había que generar divs dinámicos en la página y hacer la estructura del html fue muy complicado. También el cómo se generan los elementos HTML desde JS y generar el objeto del examen fue muy complicado por las carencias de conocimientos de JS en general.

# 9-Lecciones aprendidas

## 1-Consistencia en trabajar (conseguido) y consistencia al programar (no conseguida)

Lo primero que me planteé fue un calendario de sprints para realizar el trabajo, aunque en Noviembre ya estaba muy fuera de sprints ya que iba muy adelantado no dejé de trabajar y llegué al final del proyecto muy tranquilo y liberado.

También me he acostumbrado a guardar todos los enlaces que he usado y a documentar todo lo que he hecho en el dia.

Sin embargo, a la hora de programar sigo sin ser consistente, hago cosas que digo “lo pongo ahora así para probar y luego lo cambio” en vez de hacerlo bien desde el principio. Tampoco soy capaz de ser consistente con los nombres de archivo, algunos son muy genéricos. También tengo que acostumbrarme a comentar más el código

## 2-FRONTEND

El 90% de la aplicación reside en el proceso de modificar examenes, y ese proceso es 90% FRONT, asi que buena parte del proyecto estuvo atascado en esta parte. Gracias a la estructura que me pasó Miguel aprendí un montón de su código y le he cogido el gusto a la parte de FRONT que tenía mas floja.

Gracias a la confianza que me dio mejorar en el FRONT un par de procesos de modificar exámenes se modificaron a AJAX , que también me ayudó a entender esa parte que también llevaba floja.

## 3-Descansar es importante

Hubo muchos dias en el primer desarrollo de la aplicación (Abril-Junio) que no soltaba el equipo hasta que no salía lo que estaba haciendo, que repercutía al rendimiento de lo que quedaba de la semana. En el segundo desarrollo fui poniéndome pequeñas metas y el rendimiento se triplicó

# 10-Horas dedicadas

Hasta el dia 29/11/2020 SIN INCLUIR

175,25 H

# ANEXO1-Bibliografía:

**Guía de estilos:**

[https://fontawesome.com/](https://fontawesome.com/%20)

<https://www.flaticon.es/>

Tipografía:

<https://fonts.google.com/specimen/Montserrat>

Librería mongodb

<https://github.com/intekhabrizvi/codeigniter-mongodb-library>

Librería google

<https://github.com/bmaniar/Google-Login-in-CodeIgniter>

Tutorial integración google en codeigniter

<https://www.webslesson.info/2020/03/google-login-integration-in-codeigniter.html>

Configurar mongo con xampp

<https://learnedia.com/install-mongodb-configure-php-xampp-windows/>

Archivo necesario para integrar mongo con xampp

<https://pecl.php.net/package/mongodb>

Página para configurar el login de Google

https://console.developers.google.com/

Página para los manuales del programador

<https://jsdoc.app/>

<https://www.phpdoc.org/>

Bootstrap

<https://getbootstrap.com/>

Paleta de colores

<https://coolors.co/>

Tutorial mongodb:

<https://www.youtube.com/watch?v=lWMemPN9t6Q>

Para dudas concretas:

<https://www.google.es/>

<https://developer.mozilla.org/es/>

<https://es.stackoverflow.com/>

# ANEXO2-Código de la aplicación:

https://github.com/mjaqueFL/Sistema-Examenes