Universidad Complutense de Madrid.

DG ADE + INGENIERÍA INFORMÁTICA

Asignatura: Aplicaciones Web

MEMORIA PROYECTO FINAL “FLEXAM”

Miembros:

Daniel García Miguel

Martin Veselinov Georgiev

Alejandro Paniagua López

Alejandro de Mateo Bodegas

10/05/2024

ÍNDICE

1. Introducción a FLEXAM.

La presente Memoria hace referencia a la Entrega del Proyecto Final de la asignatura Aplicaciones Web en la titulación del Doble Grado de ADE + Ingeniería Informática en la Universidad Complutense de Madrid.

FLEXAM nació inicialmente de la inquietud incipiente que teníamos cada uno de los miembros del grupo en referencia a no solo intentar plantear un Proyecto en la asignatura que nos motivase sino que fuese útil, necesario y satisfaga una necesidad en nuestro camino universitario. Es por ello que decidimos plantearlo como temática y Business Case para nuestro Proyecto y por fin podemos exponer el resultado final que tuvimos en nuestra cabeza hace unos meses.

1. 1. ¿Qué es FLEXAM?

Tal y como lo concebimos, FLEXAM es una **plataforma web** que **redefine** la manera de **estudiar, aprender y planear** la preparación de un examen para un alumno mediante **test personalizables**, públicos y creados por **cualquier** usuario de la red.

Con FLEXAM, profesores, estudiantes y profesionales de la Educación se convierten en arquitectos de su **propia experiencia educativa**, creando y compartiendo pruebas que **se adaptan a sus necesidades** únicas.

Imaginamos una aplicación donde cada detalle, desde la puntuación hasta la selección de preguntas, se ajusta **como tú quieras**, ofreciendo una experiencia de aprendizaje que trasciende los límites de lo convencional.

Y la imaginamos pensando en **nosotros mismos hace 4 años**, empezando la Carrera, o inclusive en la convocatoria de exámenes pasada de enero sin ir más lejos. Es indescriptible lo mucho que nos hubiera gustado una **plataforma** así para **poder practicar y preparar** a la perfección todos los **exámenes tipo-test** que hemos tenido a lo largo de la cerrera (que sí, han sido unos cuantos). Ahora, por fin, podemos decir que la tenemos.

El usuario final no solo podrá resolver test o incluso crearlos, sino que consultará sus estadísticas, podrá mejorar con respecto a sus intentos pasados y mucho más. Hemos evolucionado la idea inicial que tuvimos hasta derivar en todo lo que detallaremos en las páginas sucesivas de esta memoria.

Bienvenido/a a FLEXAM.

1. 2. Funcionalidades principales de la aplicación.

FLEXAM no solo se queda en una simple plataforma para hacer test, hemos querido fomentar el concepto de Comunidad y el hecho de que CUALQUIER persona pueda crear un test, hacerlo con un formato de examen real y compartirlo dinamita la idea que teníamos sobre una Comunidad Educativa de la que cualquiera, desde profesores hasta estudiantes, pudieran sentirse parte.

Es por ello que creemos que es importante que todos los usuarios tengan funcionalidades y roles iguales, para favorecer a crear esa Comunidad, de manera que cualquier puede ser “hacedor” o creador de test a la vez que cualquiera puede practicar con los test ya creados.

A continuación, pasaremos a detallar las funcionalidades principales y más importantes de las que goza nuestra aplicación web, esforzándonos en el hecho de que hemos pretendido crear algo útil y perdurable en el tiempo.

Obviamos la funcionalidad evidente de registro y log-in de usuario, y la gestión completa de usuarios por tanto, y detallamos las funcionalidades fundamentales sobre las que se sustenta la aplicación:

1.2.1. Creación de test

El usuario puede crear test propios que decida en base a las asignaturas que tenga en su Carrera. Hemos querido cuidar mucho tanto la parte funcional como la parte estética y visual de este apartado, profundizando en la edición de cada pregunta y del test en su conjunto.

El usuario podrá elegir libremente cuántas preguntas quiere que tenga el test y cuántas opciones en cada una de las preguntas, creando así una especie de "gestor de formulario" incluso similar a gestores grandes como Google Forms o similares.

Las posibilidades de test que se pueden crear se conviertene en casi infinitas, pudiendo el usuario, al darle la opción de introducir tantas opciones como desee, crear preguntas de muchos tipos:

- Preguntas dicotómicas.
- Preguntas de prueba de verdad (Verdadero/Falso).
- Preguntas con múltiples opciones y solo una correcta.

La característica fundamental de crear un test como **PÚBLICO**, es que cualquier usuario puede verlo y realizar intentos con él. Mientras un test **PRIVADO** es aquel que solo puede ser visible y realizado por el usuario que lo creó.

Además, los usuarios al crear un test podrán elegir si quieren que aparezca su nombre como autores o que el autor sea ANÓNIMO en cambio.

1.2.2. Edición de los test creados por el usuario y añadir nuevas preguntas

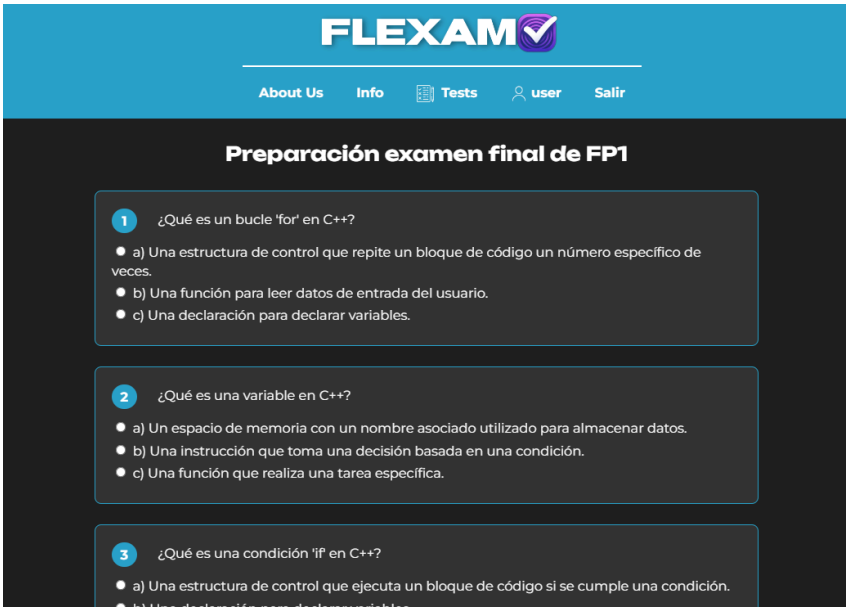
Una vez creado el test, si por cualquier casual no te ha acabado de gustar, se puede editar el texto de las preguntas y de todas las opciones. Además de poder añadir opciones y modificar cuál era la correcta. Obviamente, además, el usuario puede añadir preguntas nuevas.

Pensamos en un profesor que quisiera que sus alumnos tuvieran el test actualizado a medida que se ve la teoría en clase, añadiendo preguntas a medida que avanzan.

Añadiendo esta opción además le añadimos robustez a la aplicación, permitiendo al usuario modificar sus test en caso de querer aportarles mayor dificultad o añadir más preguntas

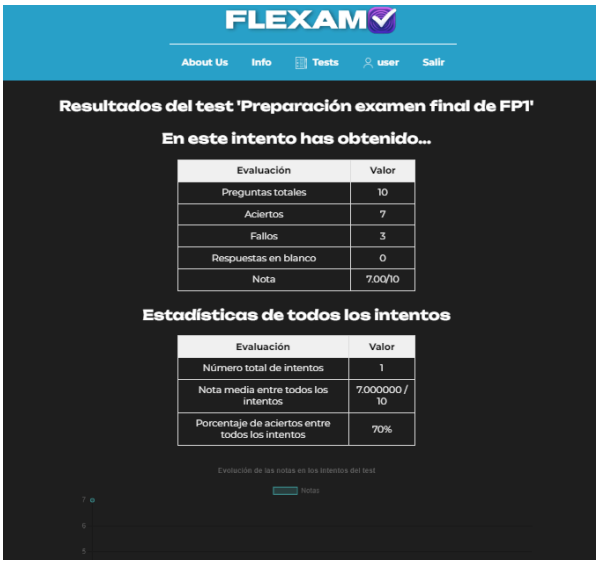
1.2.3. Realización de test públicos (y privados del propio usuario)

El usuario puede realizar todos los test **PÚBLICOS** creados por él o por otros usuarios en asignaturas de su propia carrera. Además, le aparecerán sus propios test **PRIVADOS** que solo puede ver y realizar él.



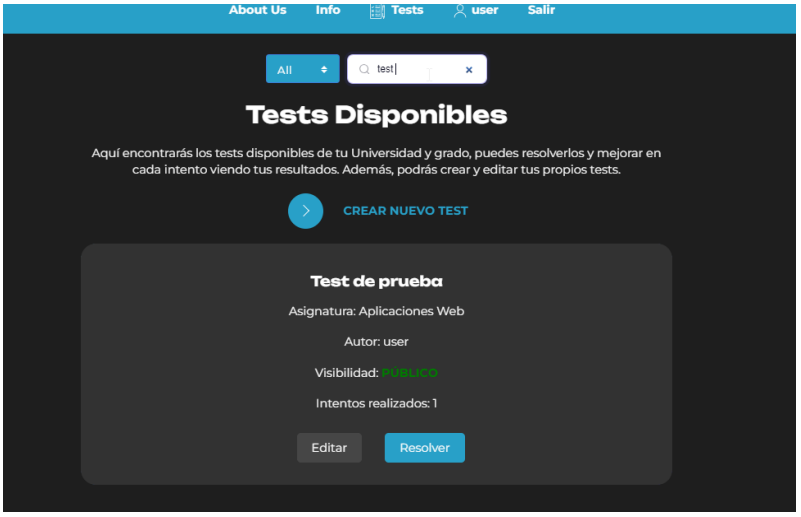
1.2.4. Estadísticas de test al finalizar

Al finalizar un test determinado, al usuario le aparecerán una serie de opciones y estadísticas para ayudarle, apoyarle y motivarle con sus datos concretos de todos los intentos que ha realizado además de ese test en concreto. Podrá ver sus Aciertos y sus Fallos, además del número de intentos realizados con ese test y una intuitiva gráfica con todos los intentos que ha hecho.



1.2.5. Búsqueda y filtrado de test

Al llegar a la pantalla de todos los Tests que tiene disponibles un usuario, este podrá filtrar por asignatura, apareciéndole solo los test de la asignatura que elija o directamente buscar por nombre, filtrando también de esta manera.



1.2.6. Estadísticas y menú de usuario

Tendremos aquí disponibles todos los tests que ha realizado el usuario. Si ha realizado tests que además él creó, podrá editarlos pinchando en el botón consecuente.

2. Instrucciones de instalación y uso del VPS.

En cuánto a las instrucciones de instalación, hemos creado nueve usuarios para diferenciar los tests que hay por cada grado. Por ejemplo, cuando inicias sesión con el usuario flex, solamente salen los tests para el grado de ade-ing.informática y así con todos.

	usuario	psw	uni	grado
1	flex	flexam	ucm	ade-ing.informática
2	admin	admin	NULL	NULL
3	dani	1234d	ucm	ing.software
4	sara	1234s	ucm	psicología
5	pani	1234p	upm	arquitectura
6	martin	1234m	upm	ing.aeroespacial
7	demateo	1234dm	ub	derecho
8	javi	1234j	ub	medicina
9	pablo	1234pa	ie	business

Tabla 1: Tabla de usuarios

Respecto al servidor proporcionado por la UCM, nuestra aplicación web está alojada en: https://vm024.containers.fdi.ucm.es/ . Se ha subido al servidor de producción el proyecto final.

Enlace a GitHub del proyecto: https://github.com/Alex7pl/PracticasAW

Los scripts para la creación de la base de datos se encuentran en el directorio BBDD dentro de la carpeta flexam de todo el proyecto (flexam.zip).

3. Manual de uso de FLEXAM.

¡Muy bien! Una vez configurado todo ya podemos empezar a utilizar FLEXAM.
A continuación, detallaremos el recorrido que puedes hacer con la aplicación ahora que ya conocemos todas las funcionalidades posibles que ofrece.

Vamos a pasar por puntos diferentes a medida que explicaremos qué tienes que hacer en cada paso y cómo están organizadas las diferentes pantallas. Tienes disponible en el **índice del documento** el punto concreto por si te interesa explorar una **funcionalidad específica**.

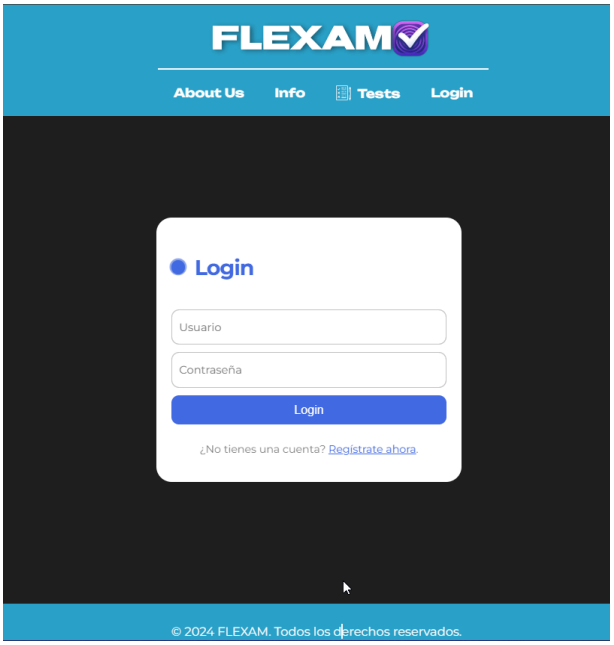
3. 1. Login y Registro en la aplicación.

Lo primero que vemos al acceder a FLEXAM es la siguiente pantalla de Inicio, en donde podemos visualizar toda la información que se nos muestra a continuación:

Es nuestra *'landing page'* desde la que podrás acceder a los Tests, a la página Sobre Nosotros, Información sobre FLEXAM o directamente registrarte. Los elementos como la imagen o el cuadro de texto están estilizados de manera que tengan un leve efecto de 'hover' al pasar el ratón por encima, en caso de que estés accediendo a la web desde PC. La página también está adaptada a otras pantallas como las de cualquier teléfono móvil.

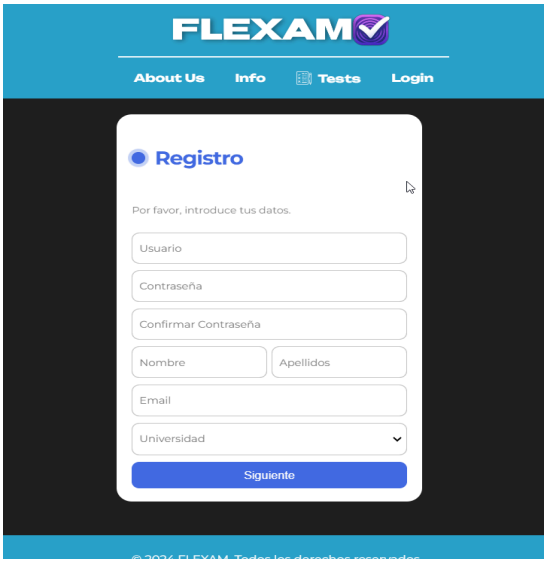
Vamos a iniciar sesión o a registrarnos. Para ello accedemos a la página pertinente pinchando en el botón **Login**.

A continuación nos debería de aparecer la siguiente pantalla:



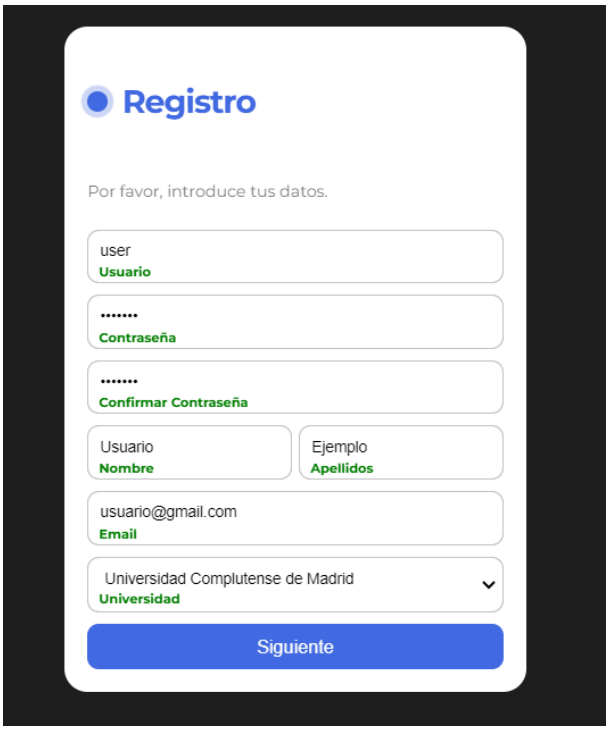
Si ya tenemos un usuario creado introducimos nuestra contraseña y procedemos a hacer login. Para este manual, **vamos a registrarnos** para ver ambos formularios. Pinchamos en **Regístrate ahora.**

Ahora se nos debería de mostrar la siguiente pantalla:

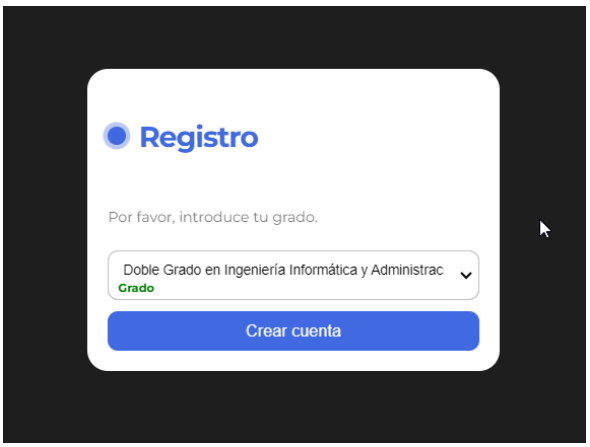


Tendremos que elegir el nombre de usuario, una contraseña, un nombre y apellidos, un email y, sobre todo, la Universidad. Esto último es muy importante ya que se nos mostrarán los test de nuestra carrera concreta, que elegiremos en el siguiente paso al pinchar en **Siguiente.**
En el ejemplo vamos a rellenar el formulario con los siguientes datos que ponemos a continuación. Al final de esta Memoria tienes disponibles usuarios y contraseñas ya creadas para poder probar la aplicación si así lo prefieres.

- Usuario: user
- Contraseña: user123 (la contraseña debe tener una extensión mínima de 5 caracteres. Lo hacemos para garantizar mínimamente contraseñas complejas. Además están cifradas con Hash y sal)
- Nombre: Usuario
- Apellidos Ejemplo
- Email: usuario@gmail.com
- Universidad: Universidad Complutense de Madrid



Pinchamos en **Siguiente.** A continuación, nos saldrá una venta así:



Ahora elegimos una carrera dentro de esa Universidad. Para el ejemplo vamos a elegir el DG en Ingeniería Informática y ADE. Pincharemos en **Crear Cuenta**

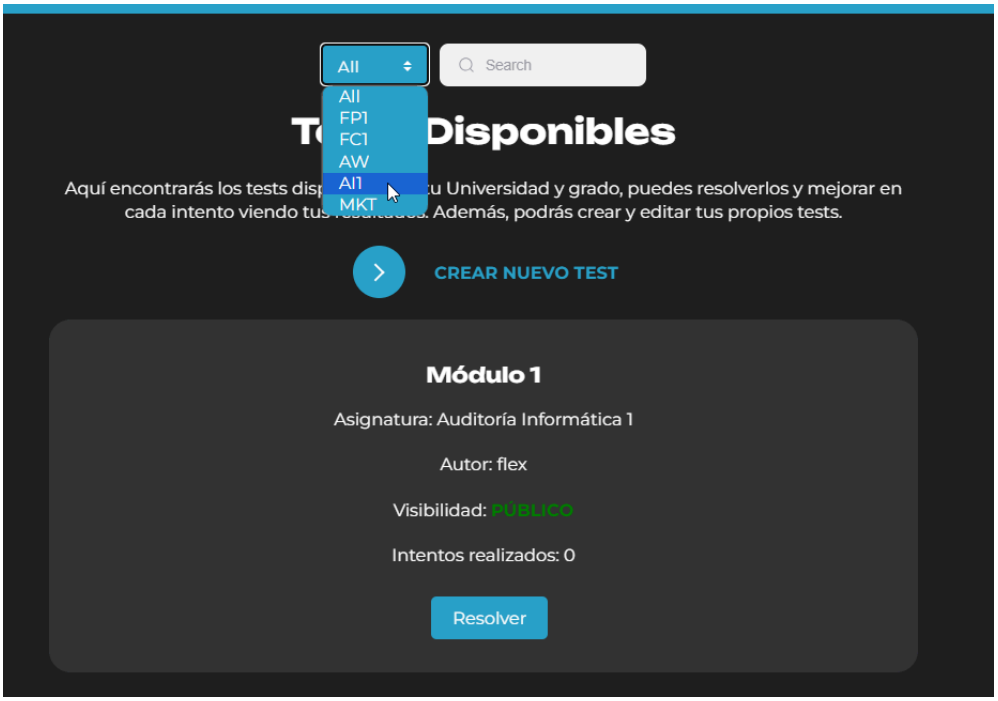
¡Ya está! Ya estamos registrados en la aplicación correctamente. Ahora nos aparece un header adaptado con nuestro botón para consultar nuestro menú de usuario. Lo veremos posteriormente.+

3. 2. Ventana de Tests disponibles

Ahora vamos a ir a la ventana de tests para consultar los tests públicos que hay de nuestra carrera dentro de FLEXAM. Recordemos que acabamos de crear un usuario, es decir, aun no ha creado ningún test en el que el Autor sea él. Pinchamos en **Tests** desde el *Header*. Nos aparece la siguiente ventana:

Como vemos, aquí se nos está mostrando EL MENÚ, donde podemos ver todos los test de nuestra carrera. Como el usuario en cuestión que acabamos de crear aun no ha creado ningún test solo nos deberían salir los test públicos que hay en la aplicación de su carrera.

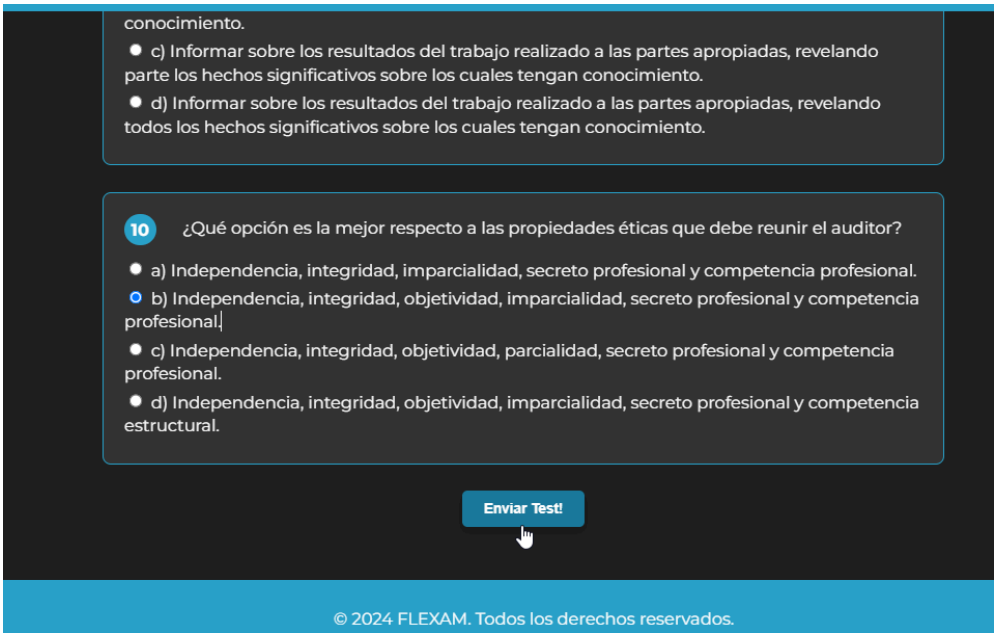
Podemos filtrar por asignatura en la parte superior o buscar por nombre, lo cual es otro filtrado. En este caso, para probar esta funcionalidad, filtraremos por asignatura. Poniendo AI1.



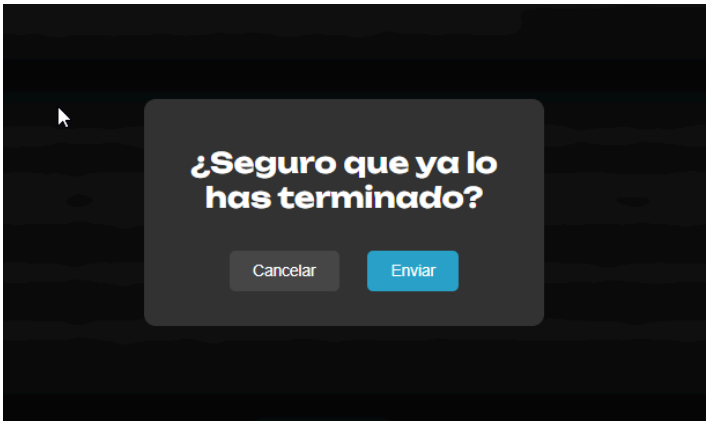
Como vemos, Auditoría Informática I tiene un test, el del Módulo 1. Vamos a probar a Resolverlo. Pinchamos en **Resolver.**

3. 3. Resolver un test

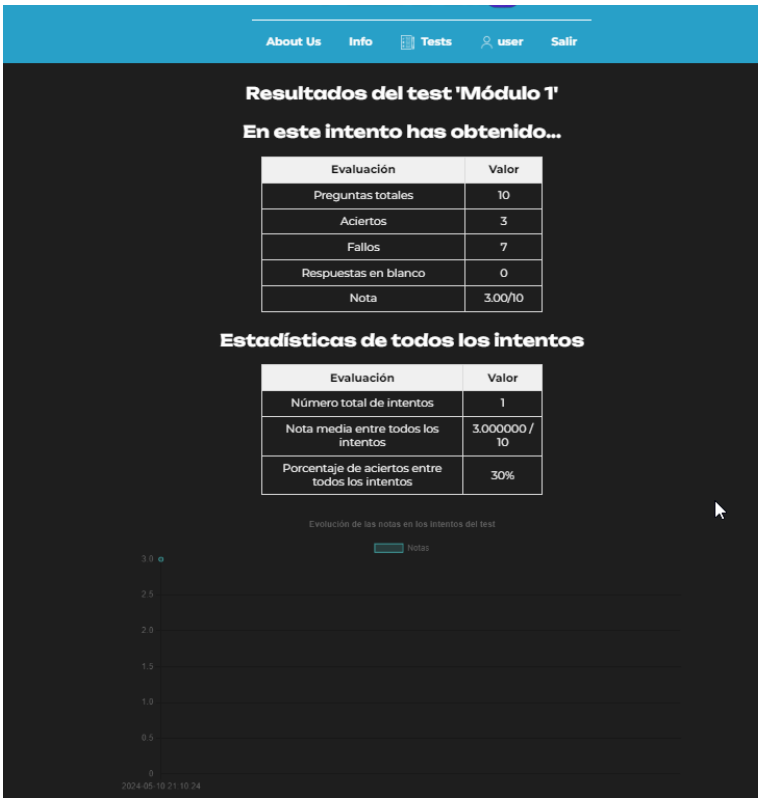
Una vez dentro del test se nos muestran todas las preguntas pudiendo elegir la correcta en cada una. Cuando hayamos completado nuestra selección. Podremos corregir el test. Pinchamos en **Enviar Test!**

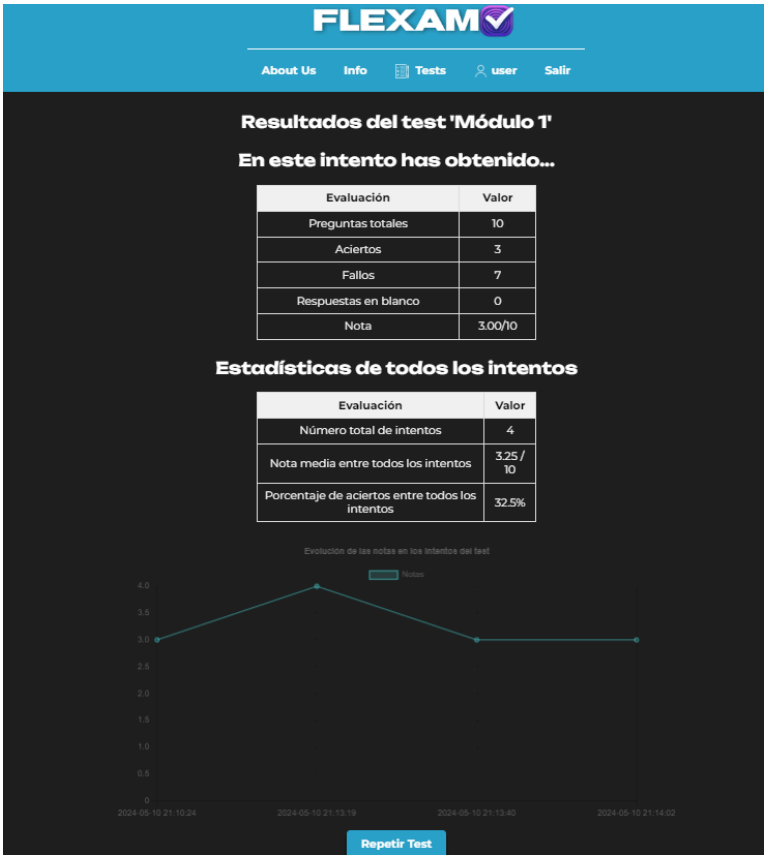


Se nos muestra un Pop-Up de confirmación:



Y a continuación, lo que veremos por pantalla será una gráfica con nuestras estadísticas al resolver el test en cuestión. Como es la primera vez que lo intentamos, la gráfica aun no representa nuestra evolución, cuantos más intentos hagamos más completa estará:


About Us
Info
Tests
user
Salir
Resultados del test 'Módulo 1'
En este intento has obtenido...
Evaluación | Valor
Preguntas totales | 10
Aciertos | 3
Fallos | 7
Respuestas en blanco | 0
Nota | 3.00/10
Estadísticas de todos los intentos
Evaluación | Valor
Número total de intentos | 1
Nota media entre todos los intentos | 3.000000 / 10
Porcentaje de aciertos entre todos los intentos | 30%
Evolución de las notas en los intentos del test
Notas


FLEXAM
About Us
Info
Tests
user
Salir
Resultados del test 'Módulo 1'
En este intento has obtenido...
Evaluación | Valor
Preguntas totales | 10
Aciertos | 3
Fallos | 7
Respuestas en blanco | 0
Nota | 3.00/10
Estadísticas de todos los intentos
Evaluación | Valor
Número total de intentos | 4
Nota media entre todos los intentos | 3.25 / 10
Porcentaje de aciertos entre todos los intentos | 32.5%
Evolución de las notas en los intentos del test
Notas
Repetir Test

Ejemplo con varios intentos

Ahora, vamos a proceder a crear un test.

3. 4. Crear un test

Desde el Header mismamente, podemos pinchar en Tests o en el icono de nuestro Usuario. En ambos, tendremos la opción de **Crear un Nuevo Test**. Pinchamos, por ejemplo, desde nuestra ventana de Usuario. Como vemos, debajo nos aparecerían todos los Tests que hemos realizado con sus respectivos intetos:

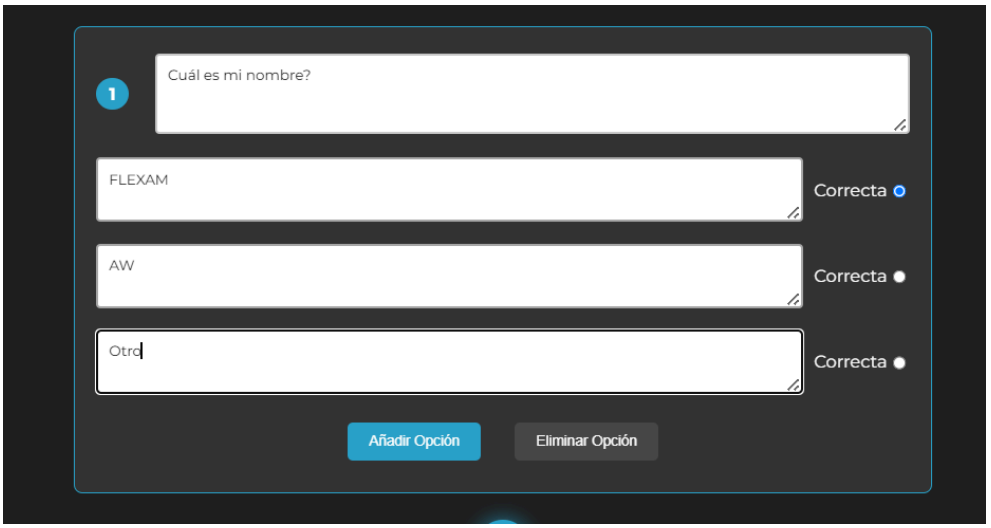
Una vez dentro de esta ventana, se nos muestra algo así:

Tendremos que fijar un nombre para el test, decidir si queremos que sea Público o no (lo haremos marcando o desmarcando la casilla correspondiente), y decidir si queremos que se muestre nuestro nombre o no. De nuevo, marcamos la casilla o la desmarcamos.

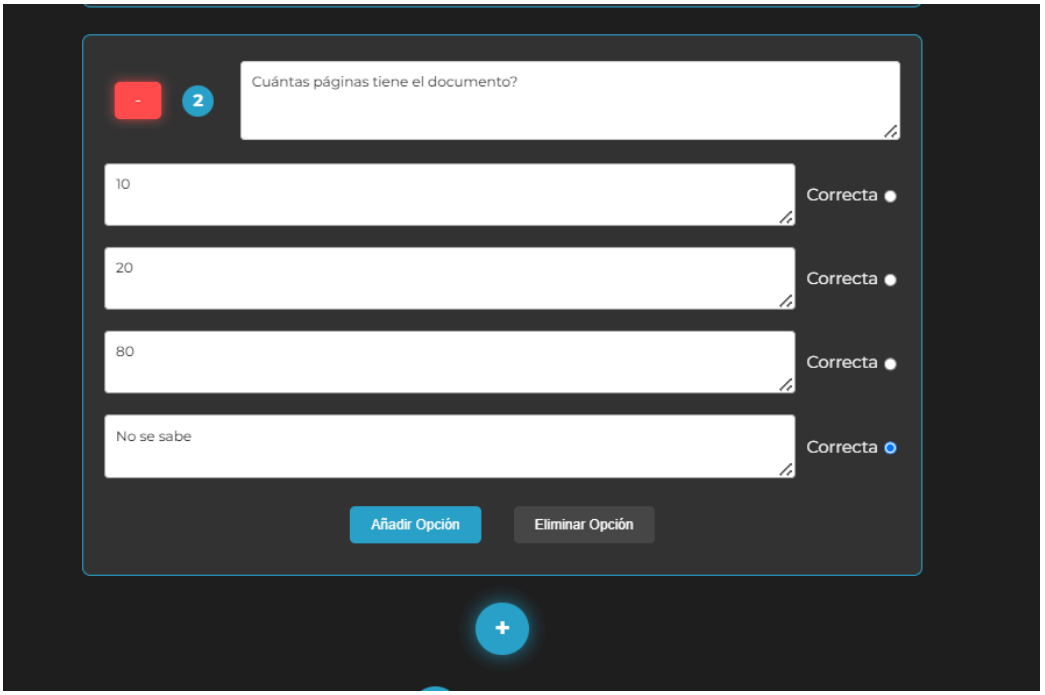
Después, tendremos que elegir la asignatura a la que va dedicado el test que crearemos. Vamos a hacer uno de prueba y pincharemos en **Crear**:

Ahora, ya deberíamos de estar en la ventana para crear nuestro test. Como vemos, es muy similar a muchos gestores de Formularios, en donde, con contenedores dinámicos, podemos escribir directamente la Pregunta en cuestión y las diferentes opciones. Nuestras preguntas son multi-opción, como comentábamos:

Y es que podemos añadir y eliminar opciones con un simple botón. Vamos a poner una pregunta de prueba:



Ahora, podríamos añadir una pregunta, pinchando en el icono redondo con el "**+**":



Al añadir esta, vemos que ahora nos aparece el botón rojo con el "**-**" para ELIMINAR la pregunta.

Podremos eliminar las preguntas, pero como mínimo debe haber una por test, si no sería inútil, la primera no se puede borrar.

Con esto sería suficiente, vamos a crear el test, y probaremos a resolverlo.

Ahora, volveremos a todos los test para poder resolverlo igual que hicimos con el anterior, pudiendo, de la misma manera, resolverlo las veces que queramos viendo nuestras estadísticas, etc.

Vamos a probar a editarlo, para acabar de ver esa funcionalidad.

3. 4. Editar un test ya creado

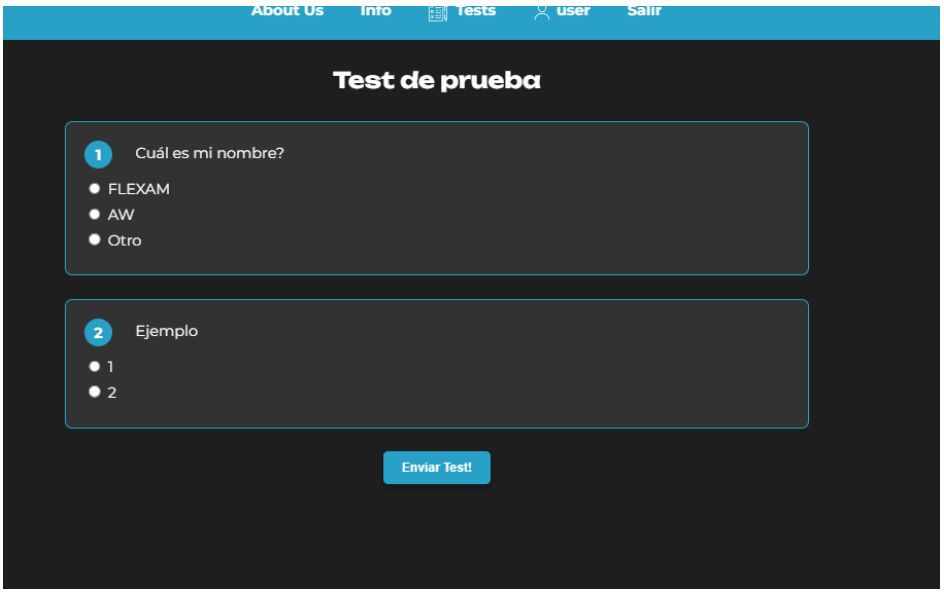
Vamos a proceder a editar un test que acabamos de crear, este en concreto. Como vemos, no solo nos aparece el botón de Resolver, sino que también el de **Editar**, pinchamos:

Una vez dentro del editor, hacemos alguna pequeña modificación, como borrar la segunda pregunta y cambiar la opción en la primera. Probaremos si al intentar resolverlo eso nos aparece bien modificado.

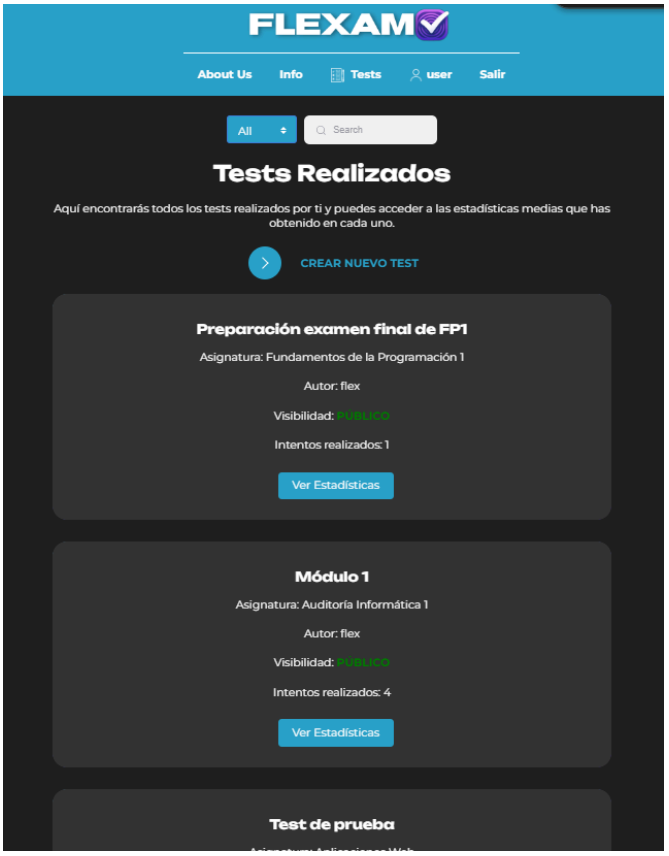
Desde aquí, siempre que MODIFIQUEMOS algo dentro de una pregunta, tendremos que darle al botón de Guardar, para guardar la selección. En el caso de que queramos ELIMINAR una pregunta, simplemente pulsamos en Eliminar Pregunta y quedaría como eliminada. podemos modificar tantas preguntas como queramos e incluso añadir nuevas.

Para esto último pulsamos en el botón de la parte superior y nos llevará al menú para añadir las preguntas que queramos. Como vemos:

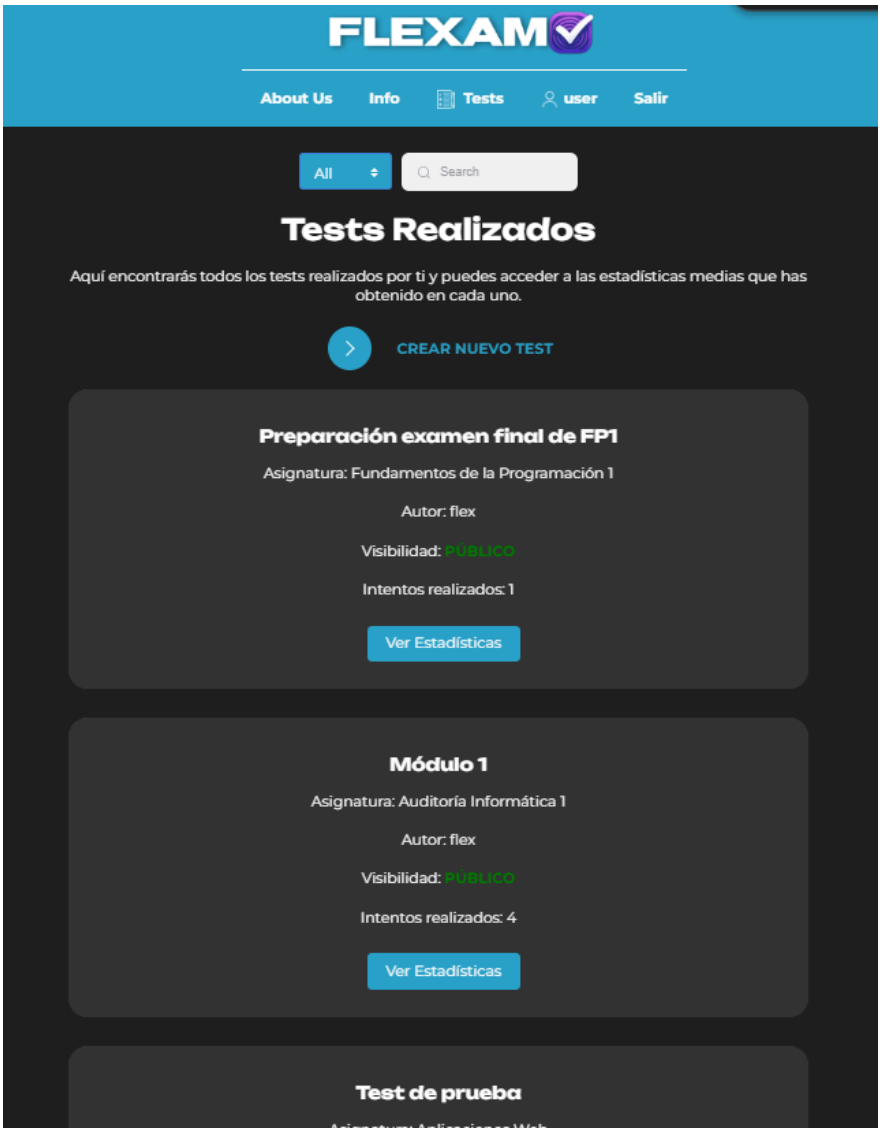
Cuando hayamos acabado pinchamos en **Guardar.**
Como vemos, efectivamente al ir a buscar de nuevo el test e intentar resolverlo se ha modificado correctamente:



Podemos ver el menú de usuario por último, desde el header, pinchando en el icono con nuestro nombre de usuario:



La información que aquí se muestra son los tests que hemos completado con sus respectivos intentos.



Las posibilidades son muy amplias y puedes proceder a probar FLEXAM de muchas maneras. Incluimos en este documento un total de nueve usuarios de diferentes universidades para poder probar bien la funcionalidad en diferentes situaciones.

4. Arquitectura de FLEXAM.

Procedemos a detallar la arquitectura y diseño concretos que hemos seguido.

Hemos aplicado la arquitectura de 3 capas en el Proyecto tras una completa refactorización con respecto a las primeras entregas preliminares. En esta ocasión empleamos scripts de tipo DAO, SA, TO, y además scripts adicionales de JavaScript dentro de la carpeta js, dedicados a la lógica dinámica para funciones como añadir una opción nueva a un test, etc, siendo un apoyo al FrontEnd.

Los scripts de funcionalidad son los que aparecen en el árbol junto con Aplicacion.php, incluido. Mencionar que utilizamos clases para una correcta encapsulación y teniendo algunas que heredan de otras, como en el claso de Formulario, FormularioLogin, etc.

Otros scripts adicionales interesantes son los comunes a todo el proyecto (config.php, footer.php, header.php).

Por último, también tenemos scripts adicionales para las vistas y algunos para pantallas que no interfieren en funcionalidad, como pudiera ser aboutus.php, por ejemplo.

Como no podía ser de otra manera, también tenemos una carpeta de recursos con las fotografías e imágenes incluidas dentro de la aplicación. A continuación se puede comprobar el árbol completo y la organización de ficheros de nuestro Proyecto, que adjuntamos también de manera completa con esta Entrega:

```
+---BBDD
|      Flexam_create_database.sql
       Flexam_insert_data.sql
|
+---includes
|   +---comun
|   |       config.php
|   |       footer.php
|   |       header.php
|   |
|   +---DAO
|   |       DAOAsignatura.php
|   |       DAOGrado.php
|   |       DAOIntento.php
```

```
|   |       DAOOpcion.php
|   |       DAOPregunta.php
|   |       DAOTest.php
|   |       DAOUniversidad.php
|   |       DAOUsuario.php
|   |
|   +---SA
|   |       SAAsignatura.php
|   |       SAGrado.php
|   |       SAIntento.php
|   |       SAOpcion.php
|   |       SAPregunta.php
|   |       SATest.php
|   |       SAUniversidad.php
|   |       SAUsuario.php
|   |
|   \---TO
|           AsignaturaTO.php
|           GradoTO.php
|           IntentoTO.php
|           OpcionTO.php
|           PreguntaTO.php
|           TestTO.php
|           UniversidadTO.php
|           UsuarioTO.php
|   |   AddQuestions.php
|   |   Aplicacion.php
|   |   ClasificarTests.php
|   |   CreacionTest.php
|   |   DatosIntentos.php
|   |   EditQuestions.php
|   |   Formulario.php
|   |   FormularioLogin.php
|   |   FormularioRegistro1.php
|   |   FormularioRegistro2.php
|   |   Grafica.php
|   |   ListarTests.php
|   |   Logout.php
|   |   PreguntasTest.php
|
+---js
|       add_questions.js
|       edit_questions.js
|       grafica_resultados.js
|       header_mobile.js
```

```
|      jquery-3.7.1.min.js
|      realize_tests.js
|      search_test.js
|
+---resources
|  \---imagenes
|          alex.png
|          dani.png
|          estudiantes.jpg
|          flexam_sinfondo.png
|          grupo_trabajando.jpg
|          logo_letras.png
|          logo_sombra.png
|          martin.png
|          pani_cut.png
|
\---styles
       styles.css
|  .htaccess
|  aboutus.php
|  add_questions.php
|  create_test.php
|  edit_questions.php
|  index.php
|  info.php
|  login.php
|  menu_tests.php
|  realize_test.php
|  resultados_test.php
|  resultados_totales.php
|  sign_up1.php
|  sign_up2.php
|  test_created.php
|  user_menu.php
```

4. 1. BBDD

La base de datos “flexam” del proyecto está diseñada para gestionar una plataforma de exámenes online, cubriendo diversos aspectos de usuarios, universidades, grados académicos, asignaturas, y las relaciones entre ellos, así como la gestión de pruebas y sus resultados.

La estructura de la base de datos refleja una jerarquía clara y bien organizada entre sus entidades principales. Las tablas usuarios, universidades, grados, asignaturas, y tests forman el núcleo de la base de

datos. Los usuarios están vinculados a universidades y grados a través de claves foráneas, lo que permite una gestión flexible de los datos en un contexto educativo. Además, la tabla grados está relacionada con universidades, y la tabla asignaturas puede estar vinculada tanto a universidades como a grados específicos mediante la tabla de relación grado_asignatura, indicando qué asignaturas son impartidas en cada grado y universidad.

La gestión de los exámenes se realiza a través de la tabla tests, que está relacionada con la tabla usuarios para indicar quién ha creado cada examen. Esta tabla se conecta también con asignaturas a través de test_asignatura, lo que especifica a qué asignatura pertenece cada examen. Los exámenes se componen de preguntas, que a su vez incluyen múltiples opciones, facilitando la creación de pruebas variadas y adaptativas. Además, la tabla respuesta_usuario registra los intentos de los exámenes por parte de los usuarios, almacenando resultados y otras métricas relevantes como la fecha del intento y las restricciones aplicadas, proporcionando así una herramienta robusta para análisis y seguimiento del rendimiento estudiantil.

En resumen, la base de datos está diseñada para ofrecer una gestión integral y flexible de pruebas y evaluaciones en un contexto educativo universitario, facilitando la administración de usuarios, cursos y resultados de manera eficiente y estructurada. Cada inserción y relación en la base de datos apoya el proceso de evaluación, permitiendo la personalización y adaptación según las necesidades específicas de cada universidad y grado.

4. 2. Includes

4.2.1. comun

4.2.1.1. config.php

El archivo config.php en el proyecto inicializa la configuración del entorno y la conexión a la base de datos, adaptándose a entornos de desarrollo local y de producción mediante la definición de constantes para la base de datos y rutas base, además de inicializar la aplicación.

Importante! Será necesario poner la contraseña a vacío:

```
define('DB_PASSWORD', '');
```

En el caso de que queramos probarlo en localhost

4.2.1.2. footer.php

El archivo footer.php en el proyecto define el pie de página de la aplicación, mostrando derechos de autor y un enlace a GitHub para información adicional sobre el proyecto.

4.2.1.3. header.php

El archivo header.php define la cabecera del sitio, incluyendo enlaces de navegación, manejo de sesión, y ajustes responsivos, complementados con estilos y fuentes externas.

4.2.2. DAO

Los DAOs (Data Access Objects) desempeñan un papel crucial en la capa de acceso a datos, permitiendo que la aplicación interactúe de manera eficiente y segura con la base de datos. Los archivos PHP en el directorio DAO corresponden a diferentes entidades del modelo de datos como **DAOAsignatura**, **DAOGrado**, **DAOIntento**, **DAOOpcion**, **DAOPregunta**, **DAOTest**, **DAOUniversidad**, y **DAOUsuario.** Cada uno de estos archivos contiene métodos específicos para manejar las operaciones de base de datos relacionadas con su entidad correspondiente.

Un DAO en nuestro Proyecto funciona como un intermediario entre la base de datos y las capas superiores de la aplicación. Por ejemplo, el DAOAsignatura tiene métodos como obtenerNombreAsignaturaTest y obtenerAsignaturasPorUsuario, que manejan consultas específicas relacionadas con asignaturas. Este enfoque encapsula las operaciones de base de datos, haciendo que el código sea más modular y fácil de mantener. Además, los DAOs utilizan técnicas como la preparación de declaraciones para proteger contra inyecciones SQL y asegurar que los datos sean manejados de forma segura.

En el contexto de nuestro proyecto, los DAOs son fundamentales para gestionar las interacciones con la base de datos relacionadas con la administración de tests y usuarios, entre otros. Proporcionan una interfaz de programación clara para realizar consultas, inserciones, actualizaciones y eliminaciones de datos. Por ejemplo, mediante el DAOAsignatura, la aplicación puede recuperar no solo los nombres de las asignaturas sino también listar todas las asignaturas asociadas a un usuario en particular, lo cual es esencial para funciones como la personalización del contenido y la gestión de accesos y permisos dentro de la plataforma educativa.

4.2.3. SA

En nuestro proyecto, los SA (Servicios de Aplicación) desempeñan un papel fundamental en la arquitectura de la aplicación, actuando como una capa intermedia entre la interfaz de usuario y la lógica de acceso a datos implementada en los DAOs. Los archivos PHP en el directorio SA representan los diferentes Servicios de Aplicación para cada entidad de nuestro modelo como **SAAsignatura, SAGrado, SAIntento, SAOpcion, SAPregunta, SATest, SAUniversidad, y SAUsuario.** Estos servicios encapsulan la lógica de negocio, asegurando que las operaciones sobre los datos sean coherentes y estén correctamente gestionadas, más allá del simple acceso a los datos que ofrecen los DAOs.

Por ejemplo, la clase SAOpcion ilustra cómo se pueden manejar las operaciones complejas que involucran las opciones de las preguntas en los tests. Desde listar opciones por pregunta específica, crear, actualizar, hasta eliminar opciones, todos estos métodos de la clase proporcionan una interfaz simplificada para realizar estas tareas. Estos métodos, a su vez, llaman a métodos específicos de DAOOpcion para interactuar con la base de datos. Esto demuestra una clara separación de preocupaciones, donde SAOpcion se centra en la lógica de negocio y DAOOpcion en la interacción directa con la base de datos.

La utilidad de los Servicios de Aplicación en nuestro proyecto se refleja en su capacidad para proporcionar puntos de acceso unificados y controlados para las operaciones relacionadas con las entidades. Por ejemplo, en el método crearOpciones, no solo se añaden opciones a la base de datos, sino que también se pueden implementar reglas de negocio adicionales, como la validación de las entradas o la verificación de permisos de usuario. Esta capa de servicios es esencial para mantener la integridad y la seguridad de las operaciones en la aplicación, facilitando la escalabilidad y mantenimiento del sistema al centralizar la lógica de negocio en un solo lugar.

4.2.4. TO

En nuestro proyecto, los objetos de transferencia de datos (TO, por sus siglas en inglés "Transfer Objects") son cruciales para manejar y transferir datos entre las diferentes capas de la aplicación, especialmente entre los DAOs, SAs y la interfaz de usuario. Los TOs actúan como estructuras de datos simples que no contienen lógica empresarial, más allá de almacenar y recuperar sus propios datos internos. En el directorio TO, tenemos diferentes clases como

AsignaturaTO, GradoTO, IntentoTO, OpcionTO, PreguntaTO, TestTO, UniversidadTO, y UsuarioTO, cada una correspondiente a una entidad específica de la base de datos.

Tomemos como ejemplo IntentoTO, que encapsula toda la información relacionada con los intentos de un usuario para realizar un test. Esta clase contiene propiedades como idTest, idUsuario, idIntento, nota, aciertos, fecha y restriccion, junto con sus respectivos métodos getter que permiten acceder a estos valores. La clase se inicializa opcionalmente con valores para cada uno de estos campos, lo cual facilita la creación de un objeto IntentoTO en diferentes contextos de la aplicación, dependiendo de los datos disponibles en ese momento.

La utilización de TOs en la aplicación tiene varias ventajas. Primero, simplifican el paso de datos complejos entre las capas de la aplicación, ya que encapsulan varios atributos en un único objeto. Esto reduce el número de parámetros necesarios en las llamadas de método y clarifica el uso de datos a través de la aplicación. Segundo, como contenedores de datos, los TOs ayudan a desacoplar la lógica de negocio de las operaciones de base de datos, lo que permite modificar una capa sin afectar a las otras. Por último, ofrecen una forma de organizar y estandarizar la estructura de datos que se utiliza en el proyecto, lo cual es especialmente útil en proyectos grandes donde la consistencia en la manipulación de datos es crítica.

4.2.5. Scripts de las Clases dedicadas a la funcionalidad:

4.2.5.1. Addquestions.php

El archivo **AddQuestions.php** contiene la clase AddQuestions, que se utiliza para añadir preguntas a tests específicos mediante un formulario interactivo, mostrando y procesando dinámicamente preguntas y opciones. Utiliza las clases SATest, SAPregunta, y SAOpcion para manejar la creación y actualización de preguntas y opciones en la base de datos basándose en entradas del usuario.

4.2.5.2. Aplicación.php

Aplicacion.php define una clase que mantiene el estado global del sistema, gestionando la inicialización de la aplicación, la conexión a la base de datos, y asegurando una única instancia de la clase mediante el patrón Singleton.

4.2.5.3. ClasificarTests.php

ClasificarTests.php facilita la búsqueda y clasificación de tests basada en criterios específicos como el nombre del test y la asignatura, utilizando para ello métodos de las clases SATest, SAAsignatura, SAUsuario, y SAIntento.

4.2.5.4. CreacionTest.php

CreacionTest.php incluye la clase CreacionTest que ofrece funcionalidades para crear nuevos tests, proporcionando un formulario para la entrada de datos del test y procesando la creación de estos en la base de datos con ayuda de la clase SATest.

4.2.5.5. DatosIntentos.php

DatosIntentos.php implementa una clase para construir tablas de resultados y estadísticas de intentos de tests, ofreciendo métodos para visualizar tanto los resultados individuales como las estadísticas acumuladas.

4.2.5.6. EditQuestions.php

EditQuestions.php contiene la clase EditQuestions que facilita la edición de preguntas de tests existentes, permitiendo a los usuarios modificar, añadir opciones o eliminar preguntas a través de una interfaz de formulario interactivo.

4.2.5.7. Formulario.php

Formulario.php define una clase abstracta base para la gestión de formularios, proporcionando métodos para la generación de formularios, manejo de errores, y procesamiento de datos de formulario, diseñada para ser extendida por formularios específicos que implementan sus propios campos y lógica de procesamiento.

4.2.5.8. FormularioLogin.php

FormularioLogin.php extiende la clase Formulario para implementar un formulario de login específico, gestionando la entrada de datos del usuario, validación, y autenticación mediante interacción con la clase SAUsuario, y redireccionando a la página principal una vez el login es exitoso.

4.2.5.9. FormularioRegistro1.php

FormularioRegistro1.php es una clase que extiende Formulario para manejar el registro inicial de usuarios, generando campos de formulario para la entrada de nombre de usuario, contraseña, confirmación de contraseña, nombre, apellidos, email, y selección de universidad. Procesa los datos del formulario, valida la información, y almacena los datos en la sesión para su uso en el siguiente paso del registro.

4.2.5.10. FormularioRegistro2.php

FormularioRegistro2.php continúa el proceso de registro de FormularioRegistro1, permitiendo al usuario seleccionar su grado académico. Procesa y valida los datos de entrada y completa el registro del usuario en el sistema, gestionando la autenticación y redirección al inicio de sesión.

4.2.5.11. Grafica.php

Grafica.php es un script que genera datos estadísticos de los intentos de un usuario para un test específico, entregando un JSON con las fechas y notas de los intentos para su uso en gráficas y análisis de rendimiento.

4.2.5.12. ListarTests.php

ListarTests.php proporciona funcionalidades para listar tests disponibles o realizados por el usuario, mostrando detalles relevantes como el autor, visibilidad y resultados, permitiendo además acciones como resolver tests o ver estadísticas, todo presentado a través de una interfaz de tarjetas interactivas.

4.2.5.13. Logout.php

Logout.php gestiona el proceso de cierre de sesión para los usuarios, limpiando la sesión y redirigiendo al usuario a la página de inicio.

4.2.5.14. PreguntasTest.php

PreguntasTest.php facilita la visualización y procesamiento de tests, mostrando preguntas y opciones a los usuarios y permitiendo la presentación y evaluación de sus respuestas, calculando la nota final y registrando el intento en la base de datos.

4. 3. Js

4.3.1. addquestioons.js

Este script JavaScript se encarga de gestionar la interfaz para agregar y editar preguntas y opciones en un formulario de test. Al cargar la página, inicia añadiendo una pregunta predeterminada y proporciona funcionalidades para añadir más preguntas y opciones, eliminarlas y enviar todas las preguntas procesadas al servidor para su almacenamiento, facilitando la creación dinámica de tests con múltiples opciones por pregunta.

4.3.2. edit_questions.js

Este script JavaScript facilita la edición de preguntas y opciones en un formulario de test. Permite agregar opciones a las preguntas existentes, guardar cambios realizados a las preguntas y sus opciones, eliminar preguntas completas, y manejar confirmaciones para la eliminación, todo interactuando con el backend a través de peticiones asíncronas para una experiencia de usuario fluida sin recargar la página.

4.3.3. grafica_resultados.js

Este script se encarga de generar una gráfica de línea utilizando la biblioteca Chart.js. Se realiza una solicitud AJAX para obtener los datos de los intentos del usuario en un test específico, incluyendo fechas y notas. Estos datos se utilizan para etiquetar y trazar la gráfica, mostrando la evolución de las notas en cada intento.

4.3.4. header_mobile.js

Este código gestiona la visibilidad del menú en dispositivos móviles. Agrega un escuchador de eventos que oculta o muestra el menú de navegación basado en el tamaño de la pantalla. Si el dispositivo es móvil, el menú se oculta por defecto y se muestra al hacer clic en el botón de menú.

4.3.5. jquery-3.7.1.min.js

Este es el archivo minificado de jQuery versión 3.7.1, una biblioteca JavaScript rápida y concisa que simplifica la manipulación del HTML, el manejo de eventos, las animaciones y las interacciones AJAX para un desarrollo web rápido.

4.3.6. realize_tests.js

Este script maneja la formatación de preguntas que incluyen elementos de LaTeX y código, utilizando la biblioteca highlight.js para resaltar la sintaxis del código. Además, agrega un escuchador de eventos para detectar clics en enlaces, preguntando al usuario si desea reiniciar una acción específica, con la excepción del botón de subir al principio de la página.

4.3.7. search_test.js

Este script gestiona la búsqueda en la plataforma, utilizando AJAX para enviar y procesar solicitudes de búsqueda sin recargar la página. Esto mejora la interactividad y permite a los usuarios encontrar tests específicos de manera rápida y eficiente.

4. 4. Styles

En cuanto a la parte referente a styles, aquí es donde se encuentra nuestra hoja de estilos. styles.css con los estilos para todos los elementos visuales que necesitamos en el Proyecto y hemos querido refinar.

Empleamos la hoja de estilos separada en sub-apartados delimitados con comentarios durante todo el script para poder diferenciar donde se encuentran las secciones para cada tipo de elemento y estilo. Algunos de los encabezados más importantes en la hoja de estilos son:

Estilos generales: Define las fuentes importadas de Google Fonts y establece estilos básicos para el cuerpo del documento, como la familia de fuente, color de fondo, color de texto, y configuraciones de margen y ancho.
Estilos de encabezado: Incluye estilos específicos para el header, como la configuración de color de fondo, disposición de elementos, y tipografía. También maneja los estilos para elementos como .logo, .separator, y la navegación dentro del encabezado, así como el icono del menú y las adaptaciones para dispositivos móviles.
Media queries para dispositivos móviles: Ajusta varios elementos como el encabezado y los menús de navegación para mejorar la visualización y funcionalidad en dispositivos con pantallas más pequeñas.
Estilos de pie de página: Configura el estilo del footer, especificando color de fondo, color de texto, y alineación.
Estilos de botones: Define los estilos para diferentes tipos de botones, incluyendo colores, disposición, efectos de transición, y estilos para hover.
Estilos de contenedor: Abarca estilos para los contenedores generales, así como para contenedores específicos como .wrapper y contenedores para pruebas y resultados.

Estilos de formularios: Contiene estilos para diferentes tipos de formularios y sus elementos, incluyendo estilos para entradas de texto, etiquetas y botones dentro de los formularios.
Estilos de tarjetas: Maneja el diseño y los estilos de visualización para diferentes clases de tarjetas, incluyendo efectos para hover y configuraciones de fondo.
Estilos específicos para la creación de tests: Incluye estilos para formularios y elementos específicos utilizados en la creación de pruebas, como selectores de asignaturas y botones para añadir o eliminar preguntas.
Estilos adicionales: Cubre estilos para varios componentes adicionales como tablas, listas, imágenes, cajas de información, y secciones de texto.

4. 5. Vistas

En el proyecto FLEXAM, las vistas se estructuran para asegurar que la funcionalidad y la presentación están bien separadas, manteniendo el código HTML limpio y enfocado en la estructura y el diseño visual, mientras que los scripts PHP manejan la lógica de negocio y la interacción con la base de datos. Esta organización permite que el contenido dinámico sea gestionado eficientemente, cargando información específica según las necesidades del usuario y las interacciones realizadas en el sitio.

Por ejemplo, en la vista principal **index.php**, la página carga elementos estáticos como encabezados y pies de página desde archivos externos incluidos, lo que promueve la reutilización del código y facilita el mantenimiento. Además, la vista proporciona enlaces directos a otras áreas importantes del sitio como los tests, información sobre el proyecto y el acceso al sistema, todos accesibles mediante botones que mejoran la experiencia de navegación del usuario.

En vistas más interactivas, como **menu_tests.php**, se emplea una combinación de PHP y HTML para mostrar de forma dinámica los tests disponibles basados en la sesión del usuario y sus selecciones previas. Si un usuario no está autenticado, la vista lo redirige automáticamente a la página de inicio de sesión, lo que asegura que sólo los usuarios autorizados puedan acceder a contenido específico.

5. Parte de actividades.

Miembro del grupo	**Tareas**
Martin Veselinov Georgiev	- Scripts adicionales de vistas - Redacción, organización y creación de la Memoria - Refactorización de parte visual dinámica - Vistas de funcionalidades - Funcionalidad editar test y pregunta editada - Responsiveness
Alejandro Paniagua López	- Refactorización parte funcional completa - Funcionalidad menú de usuario - Funcionalidad editar test - Funcionalidad registro - Gestión de usuarios - Memoria - Servidor - Complemento a la BBDD y diseño de esta
Daniel García Miguel	- Refactorización parte funcional, pulido - Funcionalidad Crear test - Refactorización de parte visual dinámica - Funcionalidad completa referida a realización de test - Estadísticas de test - Funcionalidad editar pregunta - Memoria - Servidor - Responsiveness
Alejandro De Mateo Bodegas	- Organización y diseño de la base de datos - Funcionalidad menú de usuario - Testeo y corrección de funcionalidades completas (QA) - Funcionalidad editar pregunta - Memoria - Responsiveness

Tabla 2: Tabla de Parte de Actividades