

TRABAJO FIN DE GRADO

INGENIER´IA EN INFORMÁTICA

**PLATAFORMA WEB DE ORGANIZACIÓN PERSONAL**

Desarrollo de Una aplicación web en C#

Autor

Fco Javier Fuentes Barragán

Director

Juan Antonio Holgado Terriza



Escuela Teécnica Superior de Ingenier´ía Informa´tica y de Telecomunicacion

—

Granada, mes de 201

Contenido

[**INTRODUCCIÓN** 3](#_Toc520463469)

[**1.1** **MOTIVACIÓN** 3](#_Toc520463470)

[**1.2** **OBJETIVOS** 3](#_Toc520463471)

[**1.3** **ESTRUCTURA** 4](#_Toc520463472)

[**TECNOLOGÍAS WEB EMPLEADAS** 5](#_Toc520463473)

[**2.1 HERRAMIENTAS CLOUD DE MICROSOFT (AZURE)** 5](#_Toc520463474)

[**GESTIÓN DEL PROYECTO** 6](#_Toc520463475)

[**ESPECIFICACIÓN DE REQUISITOS** 7](#_Toc520463476)

[**4.1 ESPECIFICACIÓN INFORMAL** 7](#_Toc520463477)

[**4.2 MODELO FUNCIONAL** 7](#_Toc520463478)

[**ANÁLISIS** 8](#_Toc520463479)

[**DISEÑO** 9](#_Toc520463480)

[**6.1 ARQUITECTURA** 9](#_Toc520463481)

[**6.2 DIAGRAMA ENTIDAD RELACIÓN** 9](#_Toc520463482)

[**6.3 MOCKUPS** 9](#_Toc520463483)

[**IMPLEMENTACIÓN** 10](#_Toc520463484)

[**7.1 SEGURIDAD** 10](#_Toc520463485)

[**7.2 USABILIDAD** 10](#_Toc520463486)

[**7.3 PERSISTENCIA** 10](#_Toc520463487)

[**CONCLUSIONES** 11](#_Toc520463488)

[Ilustración 1. Captura del panel de Azure. 6](#_Toc522043581)

[Ilustración 2. Diagrama de casos de uso genérico. 9](#_Toc522043582)

**CAPÍTULO 1**

# **INTRODUCCIÓN**

Este proyecto consistirá en el desarrollo de una aplicación web que se encargará de ayudar a los usuarios a realizar una correcta organización de los eventos personales, tanto propios como comunes entre varias personas. En este trabajo se desarrollará una plataforma web responsiva mediante la cual los usuarios podrán llevar el control de sus actividades y eventos personales. La web permitirá la opción de definir nuevos eventos que podrán sincronizarse entre varios usuarios.  La plataforma se encargará de registrar eventos, buscar huecos para asignación de eventos, notificaciones, etc… Entre otras funcionalidades.

## **MOTIVACIÓN**

Hoy en día existen numerosas plataformas que se encargan de ayudar a la organización personal, pero en muchas ocasiones parecen no ser suficiente. A diario vemos muchas situaciones en las que hay falta de organización por parte de los asistentes a un evento y esto fuerza a los organizadores de los eventos a realizar modificaciones improvisadas de última hora que muchas veces afectan al transcurso normal del evento planeado. Estos errores de los que estoy hablando consisten en su mayoría de errores humanos, despistes o errores en la coordinación de las personas que a menudo suponen muchos contratiempos a la hora de organizarse.

## **OBJETIVOS**

Mi objetivo en este proyecto es crear una plataforma que ayude a la organización y que sea capaz de mitigar el error humano a la hora de realizar. Esta plataforma se encargará por una parte de proporcionar una herramienta para la organización personal mediante un calendario web, el cual llevará la cuenta de los eventos que nuestro usuario registrado tendrá en nuestro sistema. También tendremos la posibilidad de compartir los eventos con los demás usuarios del sistema, siendo este el que mediante una ayuda visual les mostrará a los usuarios su disponibilidad para asistir a dicho evento y, en caso de que esa disponibilidad sea negativa proporcionará las herramientas necesarias para poder modificar los eventos que ya tiene agregados con el fin de poder asistir al evento que desee.

## **ESTRUCTURA**

A lo largo de este documento voy a explicar el proceso de desarrollo que he seguido. Pasando primero por las tecnologías que he empleado, que como es sabido existen múltiples frameworks para el desarrollo web. Mi organización a lo largo de este proyecto, que metodología de desarrollo he seguido para conseguir optimizar el tiempo de trabajo. Los requisitos y el análisis del problema, ya que es necesario un análisis previo antes de implementar la solución. El diseño que he seguido para el desarrollo de la solución, tanto a nivel de software como a nivel estética y, finalmente la implementación del proyecto.

**CAPÍTULO 2**

# **TECNOLOGÍAS WEB EMPLEADAS**

Como he mencionado antes, existen varios frameworks para el desarrollo web. Por una parte, tenemos el desarrollo en PHP, Python con frameworks como Django y C# que es la herramienta que he decidido utilizar.

La decisión de la tecnología es una de las decisiones más importantes a la hora de realizar un proyecto y por eso es necesario realizar un análisis previo de las opciones que tenemos a nuestra disposición, valorando las ventajas y las desventajas que esa tecnología nos pueda ocasionar.

|  |  |  |
| --- | --- | --- |
| Lenguaje | Pros | Contras |
| PHP | * Numerosos frameworks. * Una comunidad muy grande. * Compatible con prácticamente todos los hostings. | * Lento en comparación con el resto. * Más código para realizar la misma tarea que los demás. |
| Python | * Muy rápido. * Fácil de desarrollar. * Gestión interna de paquetes. | * Pocos hostings compatibles. * Necesidad de un framework para desarrollo web. |
| C# | * Mejor procesamiento paralelo * Totalmente orientado a objetos * Uso de Access incluido para el acceso a las bases de datos. | * Más lento que PHP * Compilación necesaria en el lado del servidor |

Mi elección final para este proyecto ha sido C# por su facilidad a la hora de la generación de las clases, el tratamiento de los elementos de la base de datos y la integración de conexiones con los servidores de Microsoft. A parte, gracias al uso de Microsoft Visual Studio puedo emplear una estructura Modelo-Vista-Controlador de una manera mucho más sencilla y natural en el proyecto gracias a esta herramienta. También hay que resaltar que el hecho de que C# se encuentre compilado en el lado del servidor, pese a ser una de las desventajas que tiene, ya que ralentiza el trabajo, supone un aumento de la seguridad, ya que lo que se almacena en el servidor es el ejecutable del proyecto.

## **2.1 HERRAMIENTAS CLOUD DE MICROSOFT (AZURE)**

De entre las herramientas que se van a utilizar para este proyecto hay que resaltar la herramienta de Azure. Obviamente, el hecho de desarrollar una aplicación web implica que debemos de tener un sitio en el cual alojar tanto la base de datos como la web en sí. En este caso al trabajar con C# y herramientas de Microsoft debemos tener un servidor con un Sistema Operativo Windows para que se pueda compilar y ejecutar correctamente en este.

Para obtener acceso a los servicios de Azure es necesario crearse una cuenta en el portal de Azure y dentro de este ya podremos crear los recursos necesarios para poder trabajar.



Ilustración . Captura del panel de Azure.

Aquí podemos ver una captura de la pantalla que nos encontramos en el panel de Azure. Como se puede observar tenemos varios recursos ya creados, el primero se trata del recurso donde se almacenará la base de datos de SQL con el nombre de *TFGDatabase*. El segundo recurso que observamos es el servidor de SQl, *wecalendar.* En el tercer recurso almacenaremos el Back-End de la web, *WCBackend,* y en el último se encontrará el Front-End, *wecalendar*.

El hecho de trabajar con la herramienta Visual Studio implica que este servicio de Azure ya se encuentra integrado dentro de esta, permitiéndome hacer subidas al servidor directamente desde esta aplicación seleccionando la dirección que desee.

Durante la mayor parte del desarrollo de la aplicación todas las pruebas serán ejecutadas en local, dándole uso a Azure para almacenar la base de datos y poder realizar los accesos. Una vez finalizada la fase de producción tanto el Back-End como el Front-End serán subidas al servidor de Azure.

**CAPÍTULO 3**

# **GESTIÓN DEL PROYECTO**

Para el desarrollo de este Proyecto he seguido una metodología SCRUM con sprints semanales e incremental. La base de esta organización ha sido la división del proyecto en tareas sencillas o de corta duración, fechando al final de estas una reunión con el tutor para evaluar el estado o porcentaje de realización de la tarea, y corrigiendo o incrementando el trabajo para la siguiente interacción.

Cada uno de los Sprints deben ir acompañados de la documentación específica de dicha tarea fragmentando la memoria final en diversos ficheros y facilitando la realización de este documento.

Para la organización de los Sprints y las reuniones con el tutor se ha utilizado la herramienta de Redmine con la cual estructuramos los objetivos, aquí llamados *peticiones,* anotando lo que se va a realizar, las horas estimadas de cada sprint y los documentos que se generaran en dicho sprint, así como la fecha tope en la cual se debe de tener terminado. Una vez terminada una petición podemos añadir las horas reales empleadas, y el porcentaje de éxito de esta tarea. También se anotarán las cosas que no han sido terminadas o deben ser corregidas para tenerlas en cuenta para el siguiente Sprint.

Para facilitar la corrección y comunicación con el tutor se me ha proporcionado también un servidor de SVN para poder almacenar ahí mi proyecto. En este servidor almacenare el proyecto dividiendo las carpetas en *Documentación*, *Bibliografía* y *Código* *Fuente*. Esta división de carpetas favorece el trabajo mediante Sprints y ayuda a la posterior realización de la memoria y control de errores, ya que se van guardando aplicaciones independientes de cada interacción.

**CAPÍTULO 4**

# **ESPECIFICACIÓN DE REQUISITOS**

## **4.1 ESPECIFICACIÓN INFORMAL**

DESCRIPCIÓN DE LOS ACTORES QUE INTERVIENEN EN EL PROYECTO Y COMO SE RELACIONEN ENTRE ELLOS.

Como he mencionado antes este proyecto consiste en la realización de un calendario Web con la posibilidad de compartir eventos. Este calendario contará con una serie de usuarios registrados en el sistema los cuales interactuarán los unos con los otros gracias a las herramientas que se les proporcionarán. Este sistema contará primero con un calendario personal en el cual el usuario podrá anotar sus eventos personales. Dichos eventos podrán ser enviados a otros usuarios del sistema que tengamos agregados como amigos. Cuando un usuario reciba un evento de un amigo tendrá la opción de aceptarlo o rechazarlo quedando, en el primer caso agregado también a su calendario personal. Los eventos generados por los usuarios podrán ser privados o públicos por los que los usuarios también podrán buscar eventos públicos para añadirlos a su calendario si desean asistir.

También se ha dispuesto de un sistema de notas con las cuales el usuario podrá anotar recordatorios que no requieran de una fecha y una hora concreta. Junto a este sistema de notas tendremos también una sala de chat por grupo en la cual los usuarios podrán hablar para acordar más detalles de los eventos. Cada grupo vendrá incorporado con un tablón de notas mediante el cual los usuarios podrán compartir notas a todos los miembros del grupo.

Cada usuario tendrá en su perfil la opción de recibir notificaciones. Estas notificaciones hacen referencia a los eventos que el usuario tendrá hoy. Las notificaciones se aplicarán a los eventos que el usuario anote como prioritarios. Este sistema de notificaciones puede ser desactivado en cualquier momento por el usuario.

Finalmente, como todo sistema web con interacción de usuarios debe de tener una serie de administradores. En este caso los administradores serán los encargados de banear a los usuarios que realicen acciones poco éticas en la página, así como de borrar los eventos peligrosos u ofensivos del sistema.

## **4.2 MODELO FUNCIONAL**

IDENTIFICAR LOS ACTORES Y PONER LOS CASOS DE USO. PRIMERO UN DIAGRAMA DE CASO DE USO GENERAL EN EL QUE SE VEN ENCAPSULADOS VARIOS CASOS DE USO Y LOS ACTORES QUE INTERVIENEN Y LUEGO UN DIAGRAMA DE CASO DE USO DE CADA UNO DE LOS SUBSISTEMAS CON UNA EXPLICACIÓN DE CADA UNO DE ELLOS Y DEBAJO DE CADA UNO TODOS LOS CASOS DE USO.

Basándonos en la especificación anterior podemos identificar a 5 actores dentro de este sistema. El **Usuario** identificado con un nombre de usuario y un correo, un **Administrador** el cual se va a encargar de controlar el uso que los usuarios dan a la aplicación, identificado por el DNI y el correo, el **Evento**, que almacenará los datos de hora lugar y descripción, la **Nota**, que se encargará de los recordatorios y el **Mensaje,** que se encargará de establecer comunicación entre uno o varios usuarios.

Teniendo estos actores en cuanta, podemos hacer una división de los casos de uso en función de su temática. En este caso la división seria: **Administración** para los perfiles de usuario y administradores, **Eventos,** para la gestión y coordinación de los eventos, **Notas**, para la gestión de los recordatorios y **Chat** para los mensajes y los grupos.

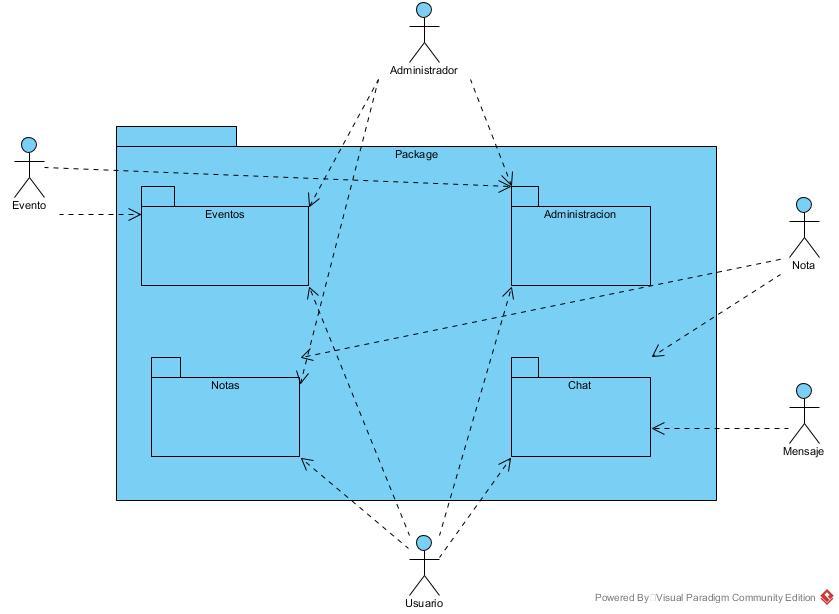


Ilustración . Diagrama de casos de uso genérico.

En esta ilustración se pueden ver las relaciones entre los actores y los 4 componentes principales de la página. El usuario al ser al actor principal es el que se encargará de interactuar con todas las componentes.

**CAPÍTULO 5**

# **ANÁLISIS**

MODELO CONCEPTUAL Y DESCRIPCIÓN DE LAS CLASES Y RELACIONES

**CAPÍTULO 6**

# **DISEÑO**

## **6.1 ARQUITECTURA**

DIFERENCIA ENTRE FRONTEND Y BACKEND. DENTRO DE BACKEND ENCONTRAMOS LA CAPA DE API, CAPA DE NEGOCIO Y DATOS.

## **6.2 DIAGRAMA ENTIDAD RELACIÓN**

PONER Y EXPLICAR EL DIAGRAMA ENTIDAD RELACIÓN QUE SE HA SEGUIDO EN LA PLATAFORMA

## **6.3 MOCKUPS**

CAPTURAS DEL ESTADO DE LA WEB Y COMO VAN QUEDANDO LAS TRANSICIONES

**CAPÍTULO 7**

# **IMPLEMENTACIÓN**

FRONEND Y BACKEND

## **7.1 SEGURIDAD**

PROTECCIÓN INTEGRIDAD ACCESO

## **7.2 USABILIDAD**

FACILIDADES DE USO PARA EL USUARIO Y NÚMEROS DE CLICKS QUE SON NECESARIOS PARA LAS TRANSICIONES

## **7.3 PERSISTENCIA**

USO DEL SERVIDOR DE AZURE Y QUE HACE PARA ASEGURAR LA PERSISTENCIA

**CAPÍTULO 8**

# **CONCLUSIONES**

CONCLUSIONES GENERALES Y ANÁLISIS DE LOS LOGROS

|  |  |  |
| --- | --- | --- |
| OBJETIVO | NIVEL DE LOGRO | OBSERVACIONES |