Gestión de Tecnologías Informáticas en las Organizaciones

EventPlanner

Memoria de Proyecto

Paul Vega

Sonia Elizondo

Sonia Elizondo

Xabier Dendarieta

Contenido

[Alcance 2](#_Toc41126941)

[Características técnicas 2](#_Toc41126942)

[Recursos 3](#_Toc41126943)

[Tiempo 3](#_Toc41126944)

[Metodología 3](#_Toc41126945)

[Anexo 5](#_Toc41126946)

[ADR Base de Datos 5](#_Toc41126947)

[Contexto 5](#_Toc41126948)

[Procesos 5](#_Toc41126949)

[Opciones consideradas 5](#_Toc41126950)

[Decisión 6](#_Toc41126951)

[Estado 6](#_Toc41126952)

[Implicaciones 6](#_Toc41126953)

[ADR Gestión de la Configuración 8](#_Toc41126954)

[Contexto 8](#_Toc41126955)

[Procesos 8](#_Toc41126956)

[Opciones consideradas 9](#_Toc41126957)

[Decisión 9](#_Toc41126958)

[Estado 10](#_Toc41126959)

[Implicaciones 10](#_Toc41126960)

# Alcance

La idea de este proyecto se basa en el desarrollo de una aplicación web para **gestionar la asistencia a eventos sociales** de pequeñas proporciones, entre los que se encuentran comidas, cenas, reuniones familiares y ocio similar.

El funcionamiento es sencillo: cada usuario puede **crear** los eventos que considere necesarios y enviar una **invitación** del evento a los usuarios que considere oportuno. Éstos, por su parte, pueden indicar si **asisten o no asisten** a cada evento, de forma que se almacenen los **asistentes en forma de listado**. De este modo, el creador de dicho evento podrá gestionar de forma muy sencilla el **número de asistentes** con los que contará y la **lista de sus nombres**.

Para ello necesitaremos tres **componentes** principales: el **cliente web**, el **servidor** y la **base de datos**. El cliente estará pensado para ejecutarse en el lado del usuario, y tanto el servidor como la base de datos, se alojarán en la máquina virtual de la que disponemos en la asignatura.

# Características técnicas

* El nombre de la aplicación web elegido es **EventPlanner**.
* Para el diseño del portal web se utilizará Bootstrap junto con HTML/CSS y JavaScript. Se han decidido estas tecnologías por las siguientes razones:
  + **Bootstrap:** es una herramienta de prototipado rápido para clientes web, por lo que nos ahorrará una cantidad considerable de tiempo a la hora de implementar nuestro cliente. Entre sus ventajas, nos permite crear proyectos receptivos y móviles de forma sencilla y completamente integrada con HTML y es de código abierto. Su uso genera resultados rápidos y visualmente elegantes.
  + **JavaScript:** es necesario en el lado del usuario para añadir funcionalidades en el cliente web sin tener que recargar la página completa. Hay un conjunto grande de librerías en este lenguaje que nos facilitarán el trabajo de implementación.
* El portal web se comunicará con el servidor web mediante una **API** (*backend*) desarrollada en el lenguaje de programación **Node.js**. Esta API será de tipo **REST**. Se ha decidido esta tecnología por lo siguiente:
  + **Node.js:** este *framework* es conocido por los desarrolladores y es una alternativa muy sencilla ante la mayoría de los lenguajes y *frameworks* disponibles. Se tuvo en consideración el uso de Python con la librería *Flask*, pero se descartó porque, a pesar de ser muy similar a la anterior, Node.js se escribe en JavaScript, al igual que nuestro cliente, y ofrece una API estable para la librería LevelDB (comentada a continuación).
* Toda la información sobre los usuarios y los eventos se gestionará mediante una base de datos llamada **LevelDB** en forma local en la máquina virtual. Se ha decidido esta tecnología por lo establecido en el ADR sobre la Base de Datos.

# Recursos

Para la implementación del sistema propuesto se hará uso de **una de las máquinas virtuales proporcionadas** para esta asignatura como principal medio de almacenamiento y despliegue. La máquina virtual citada también hará la función de servidor y contendrá la base de datos, como ya hemos comentado antes. Consideramos que son recursos suficientes y no disponemos de medios para obtener más (AWS u otras plataformas).

Además de lo anterior, necesitaremos diferentes **editores de texto** o **IDEs** para la programación del proyecto. Puesto que los lenguajes de programación que hemos propuesto son tan sencillos y populares, hay una variedad muy amplia de editores e IDEs disponibles y todos son compatibles entre sí. Con el fin de que los desarrolladores estén lo más cómodos posibles durante el desarrollo, se deja a su elección tal herramienta.

# Tiempo

Estimamos una primera demostración de funcionalidad para el día 22 de abril, miércoles. Estimamos también la finalización del proyecto para la última semana de mayo. La fecha de finalización última queda pendiente de concretarse más, mediante una estimación de tiempo más detallada.

# Metodología

Seguimos una metodología basada en la **metodología ágil** junto con **Kanban**.

Estamos siempre en contacto mediante un **servidor de Discord** exclusivamente dedicado a este proyecto. En ese servidor hemos habilitado tres chats de texto y uno de voz. De este modo, la comunicación entre los integrantes del proyecto es cercana y continua. Hemos elegido Discord por ser una plataforma fiable, sencilla y con un amplio abanico de posibilidades para comunicarse.

El **chat de texto principal** (llamado simplemente chat) es el que se utiliza para comunicación general. El **chat de texto secundario** (llamado notificaciones) se utiliza con el único fin de que cada desarrollador notifique al resto del equipo los eventos que van ocurriendo, por ejemplo, el completar una tarea importante. El **tercer chat de** texto (llamado reuniones) se utiliza exclusivamente para dejar por escrito las fechas y horas de las reuniones planificadas por el equipo (a realizar en el propio Discord). El **chat de voz** se utiliza para realizar reuniones. Es posible, en caso de necesitarlo, activar webcam o compartir pantalla.

Las **reuniones** antes citadas son solicitadas cuando al menos uno de los desarrolladores lo ve necesario, para resolver cualquier duda o incidencia. Además de las convocadas de manera más improvisada, se realiza una reunión cada dos semanas (día y hora a concretar según disponibilidad) para hacer un seguimiento global del proyecto y poner al día al equipo, con intención también de asignar a cada uno nuevas tareas de desarrollo o documentación.

También incorporamos a nuestra metodología un **tablón de tareas**, extraído de la idea de Kanban. La herramienta utilizada es **Trello**, con la que podemos crear tarjetas y organizarlas de una forma muy intuitiva. Además, dicha herramienta es ya conocida por los desarrolladores, por tanto, no es necesario que inviertan excesivo tiempo en aprender a utilizar una diferente.

Con todo esto tenemos una metodología que refuerza la comunicación entre desarrolladores y ayuda a organizar las tareas de una forma sencilla.

# Anexo

## ADR Base de Datos

### Contexto

Es necesario definir qué tipo de Base de Datos se va a utilizar para almacenar y gestionar los datos dispuestos en la aplicación.

### Procesos

1. Interacción continua entre las tres componentes de la aplicación para que la Base de Datos realice su cometido (utilizando JSON para dar estructura a los datos en los envíos):
   1. El usuario realiza una operación desde el cliente web (registrarse, identificarse, crear un nuevo evento, indicar que se asiste a un evento, etc.).
   2. El cliente web manda la acción correspondiente a la API con los datos proporcionados por el usuario.
   3. La API se comunica con la Base de Datos para indicar la petición a realizar en ella (get o put, es decir, guardar o devolver datos) con los datos disponibles.
   4. La Base de Datos almacena datos nuevos o devuelve los almacenados según haya sido realizada la petición.
      1. Los datos críticos deben estar cifrados (en este caso, únicamente la contraseña del usuario se considera relevante).
      2. No se deben perder datos debido a errores de gestión.
   5. La Base de Datos devuelve información a la API, ya sea como confirmación de éxito de una operación como de los datos requeridos.
   6. La API envía la información obtenida al cliente web.
   7. El cliente web se actualiza para mostrarse del modo que corresponda (cambio de página, confirmación de operación realizada, etc.).

Cabe resaltar las características que se requieren para el sistema, de modo que, aunque no se trata de un proceso como tal, se tengan en cuenta para la decisión: alta disponibilidad, escalabilidad y almacenamiento persistente (lo cual conlleva a que no se produzcan pérdidas por fallos internos o de la red o por cualquier otra razón).

### Opciones consideradas

Aunque hay un amplio abanico de posibilidades para la implementación de una Base de Datos, incluyendo tanto locales como en la nube, se decide acotar la comparación a dos opciones: una muy completa pero complicada de dominar y otra muy básica pero sencilla de utilizar.

1. ElasticSearch:
   1. Pros:
      1. Permite escalabilidad horizontal.
      2. Está orientada a ficheros JSON.
      3. Incluye un sistema de *master-workers* distribuido en un clúster que facilita una alta disponibilidad con alta tolerancia a fallos.
      4. Proporciona un sistema de replicación automático.
   2. Cons: el equipo no está familiarizado con su uso.
2. LevelDB:
   1. Pros:
      1. Librería de Nodejs.
      2. Uso sencillo.
      3. Más de la mitad del equipo está familiarizado con su uso.
   2. Cons:
      1. Sólo permite almacenamiento de pares clave-valor.
      2. No facilita los requerimientos del sistema.

### Decisión

Se ha escogido **LevelDB** como Base de Datos para la aplicación.

La decisión ha sido tomada en base a que la aplicación ya se está implementando en JavaScript y Nodejs. Además, algunos de los desarrolladores ya tienen experiencia tratando con la librería, por lo que no va a ser necesario dedicar horas a formación del personal antes de comenzar la programación de la Base de Datos.

Los datos que se van a utilizar en la aplicación son muy básicos y no necesitan de estructuras relacionales muy complejas, por lo que la simpleza de esta librería es suficiente.

Por último, el alcance de este proyecto es reducido debido a los recursos tecnológicos y temporales de los que se dispone, por lo que conseguir un sistema con todas las características mencionadas anteriormente no es crucial en este momento; sino que, aun siendo básico, sea funcional.

### Estado

(Tachar las que no apliquen)

1. Aceptado
2. ~~Rechazado~~
3. ~~Reemplazado~~

### Implicaciones

1. Instalación de la librería LevelDB de Nodejs en la máquina virtual utilizada para el proyecto.
2. El sistema no cumplirá, en principio, las características requeridas.

Hay que destacar que, si el contexto fuese otro, ElasticSearch sería la mejor opción para la Base de Datos, dado que permitiría alcanzar las características principales para el sistema de forma satisfactoria y medianamente automática.

#### Fecha

21 de mayo. [La decisión no había sido plasmada como ADR]

## ADR Gestión de la Configuración

### Contexto

Se requiere de una herramienta donde almacenar los archivos relacionados con el proyecto que se vayan desarrollando (código fuente, documentación, etc.), así como una organización concreta para su orden y claridad.

### Procesos

* Creación de un repositorio disponible para todos los miembros del equipo.
  + Acceso a la nube por medio de navegador, línea de comandos o interfaz gráfico.
* Organización del repositorio.
  + Utilización de árbol de directorios para ubicar rápidamente los archivos.
    - Directorio “doc”: almacenar toda la documentación referente al proyecto, por ejemplo, los propios ADRs. El directorio constará de la siguiente estructura interna:
      * Directorio “actas”: en él figurarán las actas de reunión.
      * Directorio “ADR”: para almacenaje de los ADRs, junto con la plantilla.
      * Directorio “anotaciones”: en él se guardarán apuntes y notas arbitrarias.
      * Directorio “protocolo”: en él se almacenarán los documentos sobre los protocolos de comunicación acordados entre las diferentes partes del proyecto.
      * Además de todo esto, en el directorio “doc” también va a figurar el documento de la memoria completo.
    - Directorio “src”: almacenar todo el código desarrollado.
    - No se descarta la creación de más directorios conforme las necesidades del equipo así lo requieran.
  + Utilización de ramas (*branches*) para disponer el código:
    - Rama *master*: rama principal y única.
* Política de nombres en los archivos:
  + ADRs: deben comenzar con ADR y tras un ‘\_’ especificar con sustantivo (o varios en caso de ser necesario) el objetivo principal del mismo o tema principal tratado.
  + Pudieran añadirse más normas si así se requiriese (siempre dejando constancia).
  + Actas de reunión: deben comenzar con el numero de acta que tiene asignado (secuencialmente) y debe estar representado en formato de dos cifras, seguidos de un ‘\_’, y finalizado por el día y mes de realización, ambos con formato de dos cifras también. Las actas estarán escritas en Markdown, por tanto, la extensión será *“.md”*.
  + El documento de la memoria estará en formato DOCX y llevará de nombre “memoria\_proyecto”.
* Acciones pertinentes a cambios en archivos:
  + Siempre que se realice un cambio, incluir un comentario breve con la descripción o motivo de lo realizado.
  + Realizar peticiones de cambios (*pull* *requests*), no actualizar sin control.
* Realización de pruebas sobre el código desarrollado:
  + Lanzar desde el acceso al repositorio (rama *master*) los tests pertinentes (tanto de integración como unitarios) para la comprobación del código desarrollado cada vez que se genere un cambio significativo.

### Opciones consideradas

1. GitHub:

* Pros:
  + Permite almacenar todos los tipos de archivos y de la forma establecida.
  + El equipo está familiarizado con su uso.
  + Permite realizar *pull requests*.
  + Se puede integrar con Trello (herramienta de gestión del equipo).
  + Disponemos de “Paquete Education”:
    - GitHub Pro gratuito: repositorios ilimitados, colaboradores ilimitados, etc.
  + Posibilidad de integración con Jenkins (herramienta para integración continua).
  + El equipo tiene cuentas registradas.

1. Bitbucket:

* Pros:
  + Gratuito para equipos de hasta 5 usuarios: repositorios privados ilimitados.
  + Se puede integrar con Trello (herramienta de gestión del equipo).
  + Permite realizar *pull requests*.
  + Posibilidad de integración con Jenkins (herramienta para integración continua).
* Cons:
  + El equipo no está familiarizado con su uso.
  + No se tienen cuentas ya registradas.

### Decisión

Se utiliza la herramienta **GitHub**.

No se han planteado más alternativas debido a que hay múltiples herramientas disponibles con características muy similares (Bitbucket es un ejemplo).

Se ha decidido en favor de GitHub no sólo por lo que ofrece (que, como se dice, es bastante común al resto) sino debido a que es completamente gratuito por el “Paquete Education” del que ya disfruta el equipo. Además, ya se tienen cuentas abiertas y utilizadas en la plataforma, por lo que su uso, al menos básico, es conocido.

### Estado

(Tachar las que no apliquen)

1. Aceptado
2. ~~Rechazado~~
3. ~~Reemplazado~~

### Implicaciones

Todos los miembros del equipo deben tener una cuenta registrada en la herramienta (ya se cuenta con ello) y acceso (incluyendo permisos de edición) al repositorio utilizado para el desarrollo del proyecto.

El uso de la herramienta seleccionada debe ser ágil y preciso ya que será utilizada a diario para la actualización del trabajo.

Además, se deben tener en cuenta las normas establecidas en este ADR en cuanto a la disposición del repositorio y sus componentes.

#### Fecha

27 de marzo de 2020