MEMORIA PROYECTO

**Gestión de Backlog según Metodología Scrum**

Alberto López Martínez

Silvia Suarez

Contenido

[Orígenes y Principios 5](#_Toc166232809)

[¿Qué es Scrum? 5](#_Toc166232811)

[Componentes Clave de Scrum: 5](#_Toc166232813)

[Extensión a Otros Campos: 6](#_Toc166232820)

[Elementos Principales de Scrum 7](#_Toc166232822)

[Ventajas de Scrum 8](#_Toc166232823)

[Desafíos y Consideraciones 8](#_Toc166232824)

[Proyecto: Plataforma de Gestión de Backlog Scrum 11](#_Toc166232827)

[Objetivos del Proyecto 11](#_Toc166232828)

[Arquitectura General del Proyecto 11](#_Toc166232829)

[Back-End: 11](#_Toc166232830)

[Funcionalidades Clave del Proyecto 14](#_Toc166232831)

[Tecnologías usadas 15](#_Toc166232832)

[Justificación de las Tecnologías y Herramientas Utilizadas 16](#_Toc166232833)

[Modelo de datos 17](#_Toc166232834)

[Entidades Principales 17](#_Toc166232835)

[Relaciones Clave 17](#_Toc166232836)

[Esquema de Entidad-Relación (ER) 17](#_Toc166232837)

[Diagrama ER 18](#_Toc166232838)

[Diagrama de clases 20](#_Toc166232839)

[Planificacion, Seguimiento y Evaluacion 24](#_Toc166232840)

[Situación Inicial 24](#_Toc166232841)

[Planificación 24](#_Toc166232842)

[Seguimiento del Proyecto 27](#_Toc166232843)

[Evaluación de la Ejecución 27](#_Toc166232844)

[Valoracion económica y Humana 29](#_Toc166232845)

[Valoración Económica 29](#_Toc166232846)

[Costos Totales 30](#_Toc166232847)

[Valoración Humana 30](#_Toc166232848)

[Conclusiones 32](#_Toc166232849)

[Opciones de Mejora 32](#_Toc166232850)

[Análisis de Sostenibilidad 33](#_Toc166232851)

[Conclusión General 33](#_Toc166232852)

[Manual de Usuario 35](#_Toc166232853)

[Ambito y alcance 35](#_Toc166232854)

[Funcionalidades 35](#_Toc166232855)

[Login 35](#_Toc166232856)

[Backlog 39](#_Toc166232857)

[Creación de Tarea 41](#_Toc166232858)

### Orígenes y Principios

### Scrum, un marco de trabajo ampliamente reconocido en el ámbito de la gestión y el desarrollo de productos, surge como una respuesta efectiva a los desafíos de la rápida evolución y la incertidumbre inherentes a proyectos complejos. Con su enfoque ágil y su estructura iterativa, Scrum ofrece un camino claro para gestionar proyectos de manera colaborativa y eficiente, fomentando la adaptabilidad y la innovación en un entorno dinámico y cambiante.

### ¿Qué es Scrum?

### Scrum se define como un marco de trabajo, un conjunto de prácticas, roles, eventos y artefactos que proporcionan una estructura para el desarrollo iterativo e incremental de productos. Su flexibilidad y adaptabilidad lo hacen ideal para proyectos donde los requisitos son ambiguos o pueden cambiar con frecuencia.

### Componentes Clave de Scrum:

### Roles:

### Scrum define tres roles principales: el Product Owner, el Scrum Master y el Equipo de Desarrollo. Cada uno tiene responsabilidades específicas para garantizar el éxito del proyecto. El Product Owner es responsable de maximizar el valor del producto y gestionar el Backlog del Producto. El Scrum Master actúa como facilitador, eliminando obstáculos y promoviendo un entorno colaborativo. El Equipo de Desarrollo es autónomo y autoorganizado, responsable de convertir los elementos del Backlog del Producto en incrementos potencialmente entregables de producto.

### Eventos:

### Scrum define una serie de eventos que estructuran el trabajo y permiten la inspección y adaptación continua. Estos eventos incluyen la Planificación del Sprint, la Reunión Diaria (Daily Scrum), la Revisión del Sprint y la Retrospectiva del Sprint. Cada evento tiene un propósito específico y proporciona oportunidades para la colaboración, la transparencia y la mejora continua.

### Artefactos:

### Los artefactos de Scrum representan trabajo o valor en diferentes etapas del proceso de desarrollo. El Backlog del Producto es una lista dinámica de todo lo que podría ser necesario en el producto final, ordenado por valor. El Backlog del Sprint es una lista de elementos seleccionados del Backlog del Producto para el Sprint en curso. El Incremento es la suma de todos los elementos del Backlog del Sprint completados durante un Sprint y el registro del trabajo realizado.

### Extensión a Otros Campos:

### Aunque Scrum nace en el desarrollo de software, su enfoque adaptable y su énfasis en la colaboración y la entrega de valor han llevado a su adopción en una variedad de campos más allá de la tecnología. Proyectos de ingeniería, marketing, educación e incluso áreas como la salud y la gestión de eventos han encontrado beneficios al aplicar los principios y prácticas de Scrum en sus contextos específicos.

### Elementos Principales de Scrum

#### 1. ****Equipos Scrum****

* **Scrum Team**: El equipo de Scrum es un grupo autoorganizado y multidisciplinario. Generalmente consta de un **Product Owner (Dueño del Producto)**, un **Scrum Master** y el **Development Team (Equipo de Desarrollo)**.
* **Product Owner**: Responsable de definir las prioridades del proyecto y gestionar el Product Backlog (la lista de elementos a desarrollar). Actúa como enlace entre el equipo y los stakeholders.
* **Scrum Master**: Se encarga de facilitar el proceso Scrum, eliminando impedimentos y ayudando al equipo a seguir el marco correctamente.
* **Development Team**: Grupo de profesionales que llevan a cabo el desarrollo del producto. Normalmente es un equipo pequeño, autosuficiente y multifuncional.

#### 2. ****Artefactos de Scrum****

* **Product Backlog**: Es una lista priorizada de características, historias de usuario, tareas y correcciones que el equipo debe abordar. El Product Owner es responsable de mantener y priorizar esta lista.
* **Sprint Backlog**: Es una lista de elementos seleccionados del Product Backlog para trabajar durante un Sprint. El equipo de desarrollo asume la responsabilidad de esta lista y decide cómo llevar a cabo las tareas.
* **Incremento**: Es el resultado final de un Sprint, que debería ser un producto potencialmente entregable y funcional.

#### 3. ****Eventos de Scrum****

* **Sprint**: Período corto y fijo (generalmente de dos a cuatro semanas) durante el cual se trabaja en el Sprint Backlog para entregar un Incremento al final. Los Sprints se suceden sin pausas entre ellos.
* **Sprint Planning**: Reunión al inicio de cada Sprint donde el equipo decide qué elementos del Product Backlog se incluirán en el Sprint Backlog y cómo se trabajarán.
* **Daily Scrum**: Reunión diaria de aproximadamente 15 minutos donde el equipo revisa el progreso, sincroniza actividades y ajusta la estrategia si es necesario.
* **Sprint Review**: Al final de cada Sprint, el equipo presenta el Incremento a los stakeholders y recibe retroalimentación.
* **Sprint Retrospective**: Después de la Sprint Review, el equipo se reúne para reflexionar sobre el Sprint y discutir mejoras para el próximo Sprint.

### Ventajas de Scrum

* **Flexibilidad y Adaptabilidad**: Scrum permite cambios y ajustes basados en la retroalimentación continua.
* **Transparencia**: La comunicación y la visibilidad son fundamentales en Scrum.
* **Enfoque en la Entrega de Valor**: Scrum fomenta la entrega frecuente de productos funcionales.
* **Colaboración y Autoorganización**: Scrum promueve equipos autoorganizados y colaborativos.

### Desafíos y Consideraciones

* **Disciplina y Compromiso**: Scrum requiere disciplina y un alto nivel de compromiso por parte del equipo.
* **Resistencia al Cambio**: La adopción de Scrum puede enfrentar resistencia en entornos más tradicionales.
* **Necesidad de Flexibilidad Organizacional**: La organización en su conjunto debe estar dispuesta a abrazar los principios ágiles.

En resumen, Scrum es un marco poderoso para gestionar proyectos de manera ágil, proporcionando estructura pero permitiendo flexibilidad y adaptabilidad. A través de sus equipos, artefactos y eventos, Scrum busca mejorar la entrega de valor, la transparencia y la colaboración.

La metodología ágil Scrum es ampliamente utilizada en el desarrollo de software y otros sectores debido a su enfoque iterativo y su capacidad para adaptarse a cambios y requisitos evolutivos. El backlog es un componente crucial de Scrum, ya que actúa como una lista priorizada de tareas, historias de usuario y funciones que deben implementarse.

## Necesidades del Sector

Para que una plataforma de gestión del backlog en la metodología ágil Scrum sea efectiva y aporte valor a las empresas y equipos ágiles, es crucial abordar una serie de necesidades y requisitos clave. Aquí detallo cada uno de estos puntos con mayor profundidad:

1. **Gestión Eficiente del Backlog**:
   * Las empresas necesitan una herramienta que les permita a los equipos crear, priorizar y gestionar sus elementos del backlog de manera eficiente. Esto implica la capacidad de ordenar tareas, asignar prioridades y vincularlas a objetivos o entregables del proyecto. Una interfaz intuitiva y fácil de usar es fundamental para facilitar esta gestión.
2. **Colaboración y Comunicación**:
   * Las plataformas ágiles deben facilitar la colaboración entre los diferentes miembros del equipo, como desarrolladores, gerentes de producto y stakeholders. Esto implica la necesidad de funciones de comunicación integradas, como comentarios en las tareas, chats en línea y notificaciones automáticas. La comunicación efectiva es esencial para mantener a todos los miembros del equipo informados y alineados.
3. **Visibilidad y Transparencia**:
   * Los stakeholders y los equipos deben tener acceso claro y transparente al estado del backlog. Esto incluye quién está trabajando en qué tarea, cuál es el progreso general del proyecto y cuáles son las próximas acciones planificadas. Las herramientas de visualización, como tableros Kanban, gráficos de burndown y otros informes, son esenciales para proporcionar esta visibilidad.
4. **Integración con Otras Herramientas**:
   * Dado que muchas empresas utilizan múltiples herramientas para el desarrollo de software, pruebas, control de versiones y gestión de proyectos, la plataforma debe poder integrarse con otras soluciones como GitHub, Jira, Slack, entre otras. Esta integración garantiza una experiencia fluida y sin interrupciones para los equipos que ya utilizan estas herramientas en su trabajo diario.
5. **Personalización y Flexibilidad**:
   * Los equipos ágiles varían en sus necesidades y procesos, por lo que la plataforma debe ser altamente personalizable para adaptarse a diferentes flujos de trabajo y estructuras de equipos. Esto incluye la posibilidad de ajustar la configuración, los campos personalizados y los tipos de tareas según las necesidades específicas de cada equipo.
6. **Escalabilidad y Rendimiento**:
   * La plataforma debe ser capaz de manejar múltiples proyectos y usuarios sin problemas de rendimiento. La escalabilidad es crucial para empresas que crecen y expanden sus equipos, asegurando que la plataforma pueda crecer con ellas sin comprometer la velocidad o la eficiencia.
7. **Seguridad y Control de Acceso**:
   * La seguridad es una prioridad, especialmente cuando se trata de datos sensibles del proyecto. La plataforma debe ofrecer medidas de seguridad robustas, como autenticación de dos factores, cifrado de datos en reposo y en tránsito, y control de acceso basado en roles. Esto garantiza que solo las personas autorizadas tengan acceso a la información confidencial del proyecto.
8. **Facilidad de Uso y Experiencia del Usuario**:
   * Los equipos ágiles buscan herramientas que sean intuitivas y fáciles de usar, para minimizar el tiempo de formación y maximizar la productividad. Una interfaz de usuario clara y un diseño centrado en la experiencia del usuario son elementos esenciales para garantizar una adopción exitosa de la plataforma por parte de todos los miembros del equipo.

Una plataforma en línea para gestionar el backlog en la metodología ágil Scrum debe abordar estas necesidades para ser efectiva y ofrecer valor a las empresas y equipos ágiles. Al satisfacer estos requerimientos, la herramienta puede contribuir a la eficiencia, la colaboración y el éxito general de los proyectos ágiles.

# Propuesta de diseño

El proyecto tiene como objetivo abordar las necesidades identificadas previamente y está centrado en el uso de Node.js y MySQL para el back-end, y Visual Studio Code como entorno de desarrollo.

## Proyecto: Plataforma de Gestión de Backlog Scrum

## Objetivos del Proyecto

* Facilitar la creación, organización y gestión eficiente del backlog de proyectos ágiles usando Scrum.
* Proporcionar herramientas para colaborar, priorizar tareas y visualizar el progreso del proyecto.
* Permitir la integración con otras herramientas de desarrollo y gestión.

## Arquitectura General del Proyecto

El proyecto se basa en una arquitectura cliente-servidor con Node.js para el back-end y una tecnología de front-end para la interfaz de usuario. La base de datos se manejará con MySQL, y el entorno de desarrollo será Visual Studio Code. Aquí está la estructura general del proyecto:

### Back-End:

**Node.js**: Node.js es una elección popular para el desarrollo de aplicaciones de servidor debido a su capacidad para crear sistemas escalables y eficientes. Su uso de JavaScript en el back-end lo hace especialmente atractivo para los desarrolladores que ya están familiarizados con este lenguaje. Sin embargo, como cualquier tecnología, tiene sus pros y contras:

* **Pros**:

1. **Escalabilidad**: Node.js es conocido por su capacidad para manejar grandes volúmenes de tráfico de manera eficiente, gracias a su modelo de entrada y salida sin bloqueo.
2. **Compatibilidad con bibliotecas y frameworks populares:** Node.js cuenta con una amplia gama de bibliotecas y frameworks que facilitan el desarrollo de aplicaciones web y móviles.
3. **Comunidad activa:** La comunidad de Node.js es muy activa y ofrece una gran cantidad de recursos, tutoriales y soporte para los desarrolladores.

* **Contras**

1. **Dificultad para depurar**: Debido a su naturaleza asincrónica y al uso intensivo de eventos y callbacks, puede ser más difícil depurar aplicaciones Node.js en comparación con otros lenguajes back-end.
2. **Complejidad para algunos desarrolladores**: El paradigma de programación asincrónica puede resultar complicado de entender para algunos desarrolladores, especialmente aquellos acostumbrados a un enfoque más tradicional.

**Express.js**: es un framework web minimalista y flexible para Node.js que simplifica el proceso de desarrollo de aplicaciones web. Sus pros y contras son los siguientes:

* **Pros**:

1. **Fácil de usar:** Express.js ofrece una sintaxis simple y fácil de entender, lo que facilita la creación de aplicaciones web incluso para desarrolladores principiantes.
2. **Gran cantidad de middleware disponible**: Express.js cuenta con una amplia gama de middleware preconstruido que facilita la implementación de funcionalidades comunes, como el manejo de sesiones, la autenticación y la compresión.
   * **Contras**:

Falta de características avanzadas: Aunque Express.js es ideal para construir aplicaciones web simples y rápidas, puede carecer de algunas características más avanzadas presentes en otros frameworks más grandes y completos.

**MySQL**: es una base de datos relacional ampliamente utilizada que ofrece consistencia y soporte para transacciones. Sus pros y contras son los siguientes:

* **Pros**:

1. **Amplia adopción:** MySQL es una de las bases de datos relacionales más populares y ampliamente utilizadas en el mundo, lo que significa que cuenta con una gran cantidad de recursos y soporte disponibles.
2. **Soporte para transacciones:** MySQL ofrece soporte completo para transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que garantiza la integridad de los datos en entornos críticos.
   * **Contras**:

**Menos flexible que las bases de datos NoSQL:** Aunque MySQL es excelente para manejar datos estructurados y consultas SQL, puede ser menos adecuado para escenarios donde se requiere flexibilidad y escalabilidad para datos no estructurados o en constante cambio, donde las bases de datos NoSQL pueden ser más apropiadas..

#### Autenticación y Seguridad:

La autenticación y seguridad son aspectos críticos en el desarrollo de aplicaciones web y móviles. Algunas tecnologías y herramientas comúnmente utilizadas en este ámbito son:

**JWT (JSON Web Tokens)**: JWT es un estándar abierto (RFC 7519) que define un método compacto y seguro para la transferencia de información entre dos partes como un objeto JSON. En el contexto de la autenticación de usuario y el manejo de sesiones, JWT se utiliza para generar tokens de acceso que se pueden enviar al cliente y luego incluir en las solicitudes subsiguientes para autenticar al usuario. Algunos de sus pros y contras son:

* **Pros**:

1. **Seguridad**: JWT ofrece un método seguro para la autenticación de usuario y el manejo de sesiones.
2. **Escalabilidad**: Es adecuado para sistemas distribuidos y sin estado, lo que lo hace escalable.
3. **Flexibilidad** para integrar con otros sistemas: JWT se puede integrar fácilmente con otros sistemas y tecnologías.

* **Contras**:

**Vulnerabilidad si no se implementa correctamente**: Si no se implementa correctamente, como la gestión incorrecta de la expiración de los tokens, JWT puede ser vulnerable a ciertos ataques, como la suplantación de identidad.

***Herramientas y Desarrollo:***

**Visual Studio Code**:

Visual Studio Code (VS Code) es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft que se ha vuelto muy popular en la comunidad de desarrollo de software. Algunas de sus características incluyen:

* **Pros**:

1. **Interfaz amigable**: VS Code tiene una interfaz de usuario intuitiva y fácil de usar.
2. **Integración con Git**: Ofrece una integración sólida con Git, lo que facilita la gestión de versiones del código.
3. **Extensiones para depuración y desarrollo:** Tiene una gran cantidad de extensiones disponibles que permiten personalizar y mejorar la experiencia de desarrollo.
4. **Soporte para múltiples lenguajes:** Admite múltiples lenguajes de programación, lo que lo hace versátil y adecuado para una amplia gama de proyectos.
5. **Soporte para Node.js y MySQL:** VS Code ofrece soporte completo para tecnologías como Node.js y MySQL, lo que lo hace ideal para el desarrollo de aplicaciones web.

* **Contras**:

**Puede ser pesado para sistemas con recursos limitados:** En sistemas con recursos limitados, como máquinas más antiguas o dispositivos móviles menos potentes, VS Code puede consumir una cantidad significativa de recursos, lo que puede afectar el rendimiento general del sistema.

## Funcionalidades Clave del Proyecto

* **Gestión de Backlog**: Permitir la creación y gestión del backlog, incluyendo historias de usuario, tareas y defectos. Soporte para priorización y organización.
* **Colaboración y Comunicación**: Integrar funciones de colaboración para facilitar la comunicación entre los miembros del equipo.
* **Visibilidad y Transparencia**: Proveer visualizaciones claras del progreso del proyecto, como tableros Kanban, gráficos de burndown, etc.
* **Integración con Herramientas Externas**: Permitir la integración con otras herramientas populares, como GitHub para control de versiones y Slack para comunicación.
* **Seguridad y Control de Acceso**: Implementar controles de acceso basados en roles y autenticación segura con JWT.

Esta propuesta de diseño ofrece un marco sólido para el desarrollo de una plataforma de gestión de backlog usando Scrum. La elección de tecnologías se basa en sus ventajas, comunidad de apoyo y capacidad para satisfacer las necesidades del sector. Con esta arquitectura, el proyecto puede proporcionar herramientas eficientes para la gestión de proyectos ágiles, al mismo tiempo que mantiene la seguridad y la escalabilidad.

Adema de las tecnologías principales en el proyecto utlizaremos las siguientes tecnologías que nos ayudaran a responder a las funcionalidades planteadas:

## Tecnologías usadas

Para este proyecto, utilizaremos una combinación de tecnologías modernas y potentes para garantizar un desarrollo eficiente y robusto:

 Javascript: Como lenguaje principal de programación, utilizaremos JavaScript para el desarrollo tanto del frontend como del backend de la aplicación. JavaScript nos proporciona la versatilidad necesaria para crear aplicaciones web dinámicas e interactivas.

**HTML5** : Utilizaremos HTML para definir la estructura y el contenido de nuestras páginas web. HTML es el lenguaje estándar para la creación de documentos web y nos permite organizar la información de manera semántica.

**CSS3** : Usaremos CSS para estilizar y diseñar nuestras páginas web. CSS nos permite controlar el aspecto visual de nuestros elementos HTML, lo que nos permite crear interfaces de usuario atractivas y responsivas.

**Node.js**  : Utilizaremos Node.js, junto con su administrador de paquetes npm, como entorno de ejecución de JavaScript en el servidor. Node.js nos permite construir aplicaciones web escalables y de alto rendimiento utilizando JavaScript en el backend.

**Tailwind** : Emplearemos el framework de diseño Tailwind CSS para agilizar el desarrollo frontend y mantener un código limpio y mantenible. Tailwind CSS nos proporciona un conjunto de utilidades predefinidas que facilitan la creación de interfaces de usuario modernas y personalizadas.

**Sequelize** : Para interactuar con nuestra base de datos relacional, utilizaremos Sequelize, un ORM (Object-Relational Mapping) para Node.js que simplifica la comunicación con la base de datos y nos permite trabajar con modelos de datos de manera intuitiva.

**WebPack**: Usaremos Webpack como nuestro bundler de módulos para empaquetar y optimizar nuestros archivos JavaScript y CSS. Webpack nos permite gestionar eficientemente las dependencias de nuestro proyecto y generar paquetes optimizados para su implementación en producción.

**Nodemon**: Nodemon será nuestra herramienta de desarrollo que supervisará los cambios en nuestros archivos y reiniciará automáticamente el servidor cuando sea necesario. Esto nos permitirá realizar cambios en tiempo real sin tener que reiniciar manualmente el servidor.

**Pug**: Utilizaremos Pug como nuestro motor de plantillas HTML para generar dinámicamente el contenido de nuestras páginas web en el servidor. Pug nos ofrece una sintaxis concisa y legible que facilita la creación de plantillas HTML dinámicas y reutilizables.

**Mysql**: Como sistema de gestión de bases de datos relacional, emplearemos MySQL para almacenar y gestionar los datos de nuestra aplicación. MySQL es una opción popular y robusta para aplicaciones web que requieren un almacenamiento de datos estructurado y fiable.

**Jsonwebtoken** : Utilizaremos jsonwebtoken para la autenticación y la generación de tokens de acceso seguros. jsonwebtoken nos permite implementar sistemas de autenticación basados en tokens JWT (JSON Web Tokens) de forma sencilla y segura.

**Express**: Emplearemos Express como nuestro framework de servidor web para Node.js. Express nos proporciona una arquitectura web minimalista y flexible que facilita la creación de API RESTful y aplicaciones web escalables y eficientes.

**Bcrypt** : Utilizaremos bcrypt para el cifrado seguro de contraseñas. bcrypt es una biblioteca de cifrado de contraseñas diseñada para ser segura y resistente a los ataques de fuerza bruta, lo que garantiza la seguridad de las credenciales de los usuarios.

Con esta poderosa combinación de tecnologías, estamos preparados para desarrollar una aplicación web moderna, segura y altamente funcional que cumpla con los requisitos y expectativas del proyecto.

## Justificación de las Tecnologías y Herramientas Utilizadas

La combinación de Node.js, MySQL y Visual Studio Code es una elección sólida para este tipo de proyecto debido a la flexibilidad, escalabilidad y amplio soporte de la comunidad. Node.js permite un desarrollo rápido y escalable, mientras que MySQL ofrece consistencia y seguridad en el manejo de datos. El uso de JWT proporciona autenticación segura y fácil integración con otros servicios.

Visual Studio Code es un entorno de desarrollo versátil que puede soportar el desarrollo de aplicaciones Node.js y MySQL con facilidad. Su flexibilidad y amplio soporte para extensiones lo hacen ideal para desarrolladores que buscan personalizar su flujo de trabajo y productividad.

## Modelo de datos

Para crear un esquema de Entidad-Relación (ER) que soporte las funcionalidades descritas, es importante identificar las entidades principales y las relaciones entre ellas. A continuación se presenta un esquema básico que cubre los elementos del backlog, usuarios y la interacción entre ellos.

### Entidades Principales

* **Usuario**: Representa a un usuario registrado en la plataforma.
* **Backlog**: Contiene la información general sobre el backlog y sus elementos.
* **Elemento del Backlog**: Representa historias de usuario, tareas y bugs.
* **Categoría**: Clasificación de elementos del backlog.
* **Estado**: Describe el estado del elemento del backlog (por ejemplo, "Nuevo", "En Progreso", "Completado").
* **Notificación**: Representa recordatorios o alertas enviadas a los usuarios.

### Relaciones Clave

* **Usuario-Backlog**: Un usuario puede tener varios backlogs y un backlog puede ser compartido por varios usuarios.
* **Backlog-Elemento del Backlog**: Un backlog puede contener múltiples elementos.
* **Elemento del Backlog-Categoría**: Un elemento del backlog puede pertenecer a una categoría.
* **Elemento del Backlog-Estado**: Un elemento del backlog tiene un estado asociado.
* **Usuario-Notificación**: Un usuario puede recibir varias notificaciones.

### Esquema de Entidad-Relación (ER)

#### Entidades y Atributos

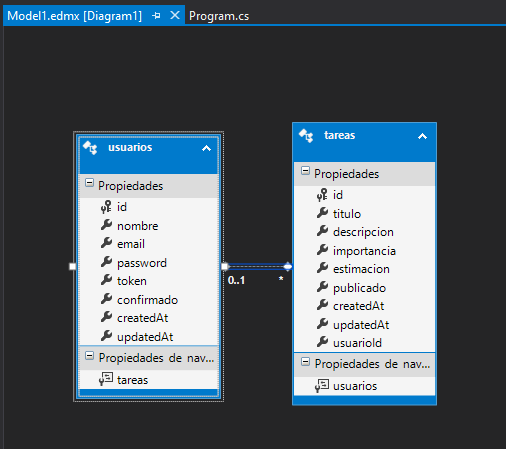
* **Usuario**
  + **usuario\_id** (clave primaria)
  + **nombre**
  + **correo\_electronico** (único)
  + **contraseña** (almacenada como hash para seguridad)
* **Backlog**
  + **backlog\_id** (clave primaria)
  + **nombre**
  + **descripcion**
  + **fecha\_creacion**
* **Elemento del Backlog**
  + **elemento\_id** (clave primaria)
  + **nombre**
  + **descripcion**
  + **prioridad** (entero para indicar prioridad)
  + **esfuerzo** (puntos de historia)
  + **backlog\_id** (clave foránea que refiere a **Backlog**)
  + **categoria\_id** (clave foránea que refiere a **Categoría**)
  + **estado\_id** (clave foránea que refiere a **Estado**)
* **Categoría**
  + **categoria\_id** (clave primaria)
  + **nombre**
* **Estado**
  + **estado\_id** (clave primaria)
  + **nombre**
* **Notificación**
  + **notificacion\_id** (clave primaria)
  + **tipo** (tipo de notificación, como "Correo", "SMS")
  + **mensaje**
  + **fecha\_envio**
  + **usuario\_id** (clave foránea que refiere a **Usuario**)

#### Relaciones

* **Usuario-Backlog**
  + Relación de muchos a muchos: Un usuario puede tener múltiples backlogs y un backlog puede ser compartido por múltiples usuarios.
  + Se necesita una tabla de unión, **Usuario\_Backlog**, con los atributos **usuario\_id** y **backlog\_id**.
* **Backlog-Elemento del Backlog**
  + Relación de uno a muchos: Un backlog puede tener múltiples elementos del backlog.
* **Elemento del Backlog-Categoría**
  + Relación de muchos a uno: Un elemento del backlog pertenece a una categoría.
* **Elemento del Backlog-Estado**
  + Relación de muchos a uno: Un elemento del backlog tiene un estado.
* **Usuario-Notificación**
  + Relación de uno a muchos: Un usuario puede recibir múltiples notificaciones.

## Diagrama ER

Este esquema ER proporciona un marco robusto para las funcionalidades descritas, permitiendo la creación, organización y gestión de elementos del backlog, así como las características de colaboración, seguridad y notificaciones.



## Diagrama de clases

Node.js se utiliza comúnmente para el desarrollo de aplicaciones del lado del servidor, utilizando un enfoque orientado a eventos y un modelo no bloqueante. Aunque en aplicaciones back-end con Node.js el uso de diagramas de clases no es tan común como en otros lenguajes orientados a objetos (como Java o C++), aún puedes crear un diagrama de clases para visualizar la estructura del proyecto y las interacciones entre componentes.

Un diagrama de clases en este contexto ayudará a comprender la estructura del código y cómo interactúan las diferentes partes de la aplicación. Aquí tienes un ejemplo básico de un diagrama de clases para el proyecto de gestión de backlog con Node.js:

**Diagrama de Clases:**

1. **UsuarioController:**
   * Métodos:
     + formularioLogin()
     + autenticar()
     + formularioRegistro()
     + registrar()
     + confirmar()
     + formularioOlvidePassword()
     + resetPassword()
     + comprobarToken()
     + nuevoPassword()
2. **TareasController:**
   * Métodos:
     + admin()
     + crear()
     + guardar()
     + activar()
     + editar()
     + guardarCambios()
     + eliminar()
3. **TareasRoutes:**
   * Rutas:
     + GET /mis-tareas
     + GET /tareas/crear
     + POST /tareas/crear
     + GET /tareas/editar/:id
     + POST /tareas/editar/:id
     + POST /tareas/activar/:id
     + POST /tareas/eliminar/:id
4. **UsuarioRoutes:**
   * Rutas:
     + GET /login
     + POST /login
     + GET /registro
     + POST /registro
     + GET /confirmar/:token
     + GET /olvide-password
     + POST /olvide-password
     + GET /olvide-password/:token
     + POST /olvide-password/:token

**Relaciones:**

* **UsuarioController** utiliza el modelo **Usuarios** para interactuar con la base de datos de usuarios.
* **UsuarioRoutes** utiliza los métodos del **UsuarioController** para manejar las solicitudes HTTP relacionadas con la autenticación y el registro de usuarios.
* **TareasController** utiliza el modelo **Tarea** para interactuar con la base de datos de tareas.
* **TareasRoutes** utiliza los métodos del **TareasController** para manejar las solicitudes HTTP relacionadas con la gestión de tareas.

**Funcionalidad de Relaciones:**

* **UsuarioController** y **UsuarioRoutes** trabajan juntos para manejar la autenticación de usuarios, el registro, la confirmación de cuentas y el restablecimiento de contraseñas.
* **TareasController** y **TareasRoutes** trabajan juntos para gestionar la creación, edición, activación y eliminación de tareas.

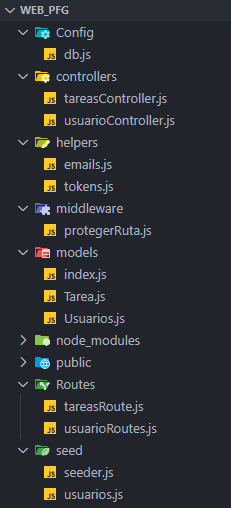
### Notas Adicionales

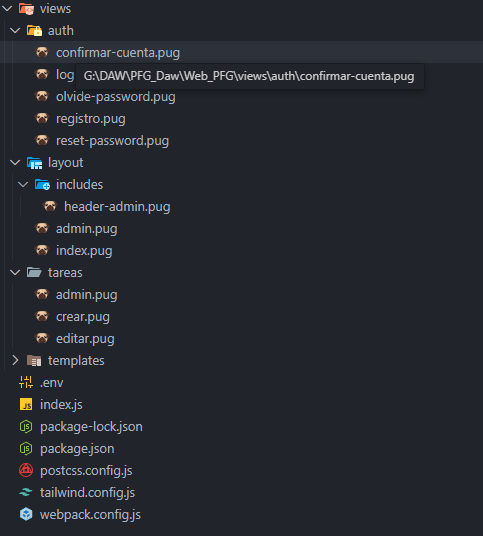
- Los controladores representan la capa de presentación o API del proyecto. Se encargan de las solicitudes del usuario y envían respuestas.

- Los servicios contienen la lógica de negocio y operaciones relacionadas con la gestión de datos.

- Las clases representan las entidades principales del proyecto y sus atributos.

- El diagrama puede ser simplificado o ampliado dependiendo de las necesidades específicas del proyecto.

****



Este diagrama de clases proporciona un marco básico para comprender la estructura de un proyecto de gestión de backlog en Node.js. Puedes personalizar y ajustar según las necesidades específicas de tu aplicación.

## Planificacion, Seguimiento y Evaluacion

### Situación Inicial

El equipo ha identificado la necesidad de una plataforma en línea para gestionar el backlog de proyectos utilizando Scrum. El producto final debe permitir a los usuarios crear y organizar elementos del backlog, colaborar con el equipo, y tener funcionalidades adicionales como notificaciones y autenticación segura.

El equipo está compuesto por un Product Owner, un Scrum Master y un equipo de desarrollo de 6 personas (desarrolladores y un diseñador UX/UI). El proyecto está programado para completarse en 12 semanas, con iteraciones (sprints) de 2 semanas.

### Planificación

El plan para este proyecto incluye los siguientes elementos clave:

**Objetivos del Proyecto:**

El objetivo principal de este proyecto es desarrollar una plataforma de gestión de backlog Scrum que permita a los equipos de desarrollo de software organizar, priorizar y gestionar eficientemente sus tareas y proyectos. Los objetivos específicos incluyen:

1. **Desarrollo de una Plataforma Funcional:**
   * Crear una aplicación web robusta y completa que satisfaga las necesidades de los equipos ágiles que siguen la metodología Scrum.
   * Proporcionar herramientas intuitivas y fáciles de usar para la creación, organización y seguimiento de elementos del backlog, como historias de usuario, tareas y bugs.
2. **Implementación de Funcionalidades Esenciales:**
   * Incorporar características esenciales, como autenticación segura de usuarios, gestión de sesiones y permisos de acceso para garantizar la seguridad de los datos.
   * Desarrollar sistemas de notificaciones para informar a los usuarios sobre cambios importantes en el backlog y recordatorios para eventos próximos.
3. **Entrega de un Producto Completo y Testado:**
   * Entregar un producto final completamente funcional y testado que cumpla con los requisitos del cliente y las expectativas del equipo de desarrollo.
   * Realizar pruebas exhaustivas en todas las etapas del desarrollo para garantizar la estabilidad, la usabilidad y el rendimiento del producto.

**Desglose del Trabajo:**

1. **Sprint 1: Análisis de Requerimientos y Diseño**
   * Realizar reuniones con el cliente y el equipo de desarrollo para definir los requisitos del producto y establecer objetivos claros.
   * Diseñar prototipos de la interfaz de usuario utilizando herramientas de diseño para visualizar el flujo de trabajo y la navegación.
   * Configurar el entorno de desarrollo y establecer un repositorio de código para colaboración y control de versiones.
   * Hitos: Requisitos del producto aprobados y prototipos de interfaz validados.
2. **Sprint 2: Configuración del Back-End y Base de Datos**
   * Implementar la base de datos utilizando tecnologías como MySQL para almacenar la información del backlog y los usuarios.
   * Configurar el entorno de desarrollo del back-end utilizando Node.js y Express.js para la lógica del servidor.
   * Desarrollar las primeras funciones de autenticación y seguridad para proteger los datos sensibles de los usuarios.
   * Hitos: Base de datos configurada y funcionalidades básicas de autenticación implementadas.
3. **Sprint 3-4: Desarrollo de Funcionalidades Principales**
   * Implementar la funcionalidad de registro e inicio de sesión seguro para los usuarios.
   * Desarrollar modelos de datos para representar los backlogs, las historias de usuario y las tareas, y establecer relaciones entre ellos.
   * Crear controladores y servicios para gestionar la creación, edición y eliminación de elementos del backlog.
   * Hitos: Funcionalidades principales completadas y primera versión del backlog funcional.
4. **Sprint 5-6: Desarrollo de Funcionalidades Adicionales**
   * Desarrollar funciones avanzadas para la priorización y la organización del backlog por categoría, estado y prioridad.
   * Implementar la capacidad de estimar el esfuerzo necesario para completar las historias de usuario y las tareas.
   * Mejorar la interfaz de usuario con funcionalidades adicionales, como filtros y vistas personalizadas.
   * Hitos: Funcionalidades adicionales implementadas y mejoras en la interfaz de usuario realizadas.
5. **Sprint 7: Colaboración y Notificaciones**
   * Integrar características de colaboración en tiempo real para permitir a varios miembros del equipo trabajar simultáneamente en el backlog.
   * Desarrollar sistemas de notificaciones y alertas para informar a los usuarios sobre cambios relevantes en el backlog.
   * Realizar pruebas exhaustivas de las funcionalidades de colaboración y notificaciones para garantizar su eficacia y rendimiento.
   * Hitos: Funcionalidades de colaboración y notificaciones implementadas y probadas con éxito.
6. **Sprint 8: Pruebas y Ajustes**
   * Realizar pruebas unitarias y de integración para identificar y corregir posibles errores y fallos en el sistema.
   * Optimizar el rendimiento y la velocidad de carga de la aplicación para mejorar la experiencia del usuario.
   * Resolver problemas de usabilidad y accesibilidad para garantizar que el producto sea fácil de usar para todos los usuarios.
   * Hitos: Pruebas completadas, errores corregidos y rendimiento optimizado.
7. **Sprint 9: Evaluación y Entrega Final**
   * Realizar pruebas finales con usuarios reales para recopilar comentarios y realizar ajustes finales según sea necesario.
   * Revisar y documentar detalladamente el código y la funcionalidad para facilitar el mantenimiento futuro del producto.
   * Preparar la entrega final del producto al cliente, incluida la instalación en un entorno de producción.
   * Hitos: Producto final entregado y aceptado por el cliente

### Seguimiento del Proyecto

El seguimiento del proyecto es fundamental para garantizar que avance de manera eficiente y que se cumplan los objetivos establecidos. Para ello, se llevarán a cabo una serie de actividades y procesos que permitirán supervisar el progreso y realizar ajustes cuando sea necesario. A continuación, se detallan las principales actividades de seguimiento del proyecto:

* **Daily Scrum**: Estas son reuniones diarias cortas y enfocadas en las que el equipo revisa el progreso realizado desde la última reunión, identifica posibles obstáculos y planifica el trabajo para el día siguiente. El objetivo es mantener a todos los miembros del equipo informados y alineados con los objetivos del sprint.
* **Sprint Reviews**: Al finalizar cada sprint, se llevará a cabo una reunión de revisión en la que se demostrará el trabajo completado durante el período. Esto brinda la oportunidad de obtener retroalimentación de los stakeholders y ajustar el rumbo si es necesario. Además, permite celebrar los logros alcanzados por el equipo.
* **Sprint Retrospectives**: Después de cada sprint, se realizará una retrospectiva para reflexionar sobre lo que funcionó bien, lo que se puede mejorar y cualquier ajuste necesario en el proceso de trabajo. Estas reuniones son fundamentales para el aprendizaje continuo y la mejora del equipo.
* **Herramientas de Seguimiento**: Se utilizarán herramientas especializadas como Trello o Jira para llevar un registro detallado de las tareas, asignaciones y progresos del proyecto. El Scrum Master será el encargado de mantener actualizado el tablero Scrum y garantizar que la información sea accesible para todo el equipo.

### Evaluación de la Ejecución

Para evaluar la ejecución del proyecto y garantizar que se cumplan los estándares de calidad y eficiencia, se implementarán diversos métodos de evaluación. A continuación, se describen las principales métricas y procesos de evaluación:

* **Indicadores de Rendimiento**: Se establecerán métricas clave como la velocidad del equipo, el número de historias de usuario completadas y el tiempo promedio para completar tareas. Estos indicadores proporcionarán una visión clara del progreso y la eficiencia del equipo a lo largo del proyecto.
* **Evaluación del Producto**: Se recopilará retroalimentación de los stakeholders y usuarios para evaluar la calidad y usabilidad del producto desarrollado. Esto se realizará mediante pruebas de usuario, encuestas y análisis de métricas de uso.
* **Encuestas de Satisfacción**: Se llevarán a cabo encuestas periódicas a los miembros del equipo para evaluar su satisfacción con el proceso de trabajo, el ambiente laboral y la colaboración entre colegas. Esta retroalimentación ayudará a identificar áreas de mejora y fortalezas del equipo.
* **Revisión Final**: Al concluir el proyecto, se realizará una revisión exhaustiva para evaluar el cumplimiento de los objetivos establecidos y la calidad del producto entregado. Esta revisión permitirá identificar lecciones aprendidas y áreas de mejora para proyectos futuros.

Este plan de proyecto muestra cómo se puede estructurar una planificación, seguimiento y evaluación de la ejecución para un proyecto de desarrollo de software basado en Scrum. Con hitos claros, seguimiento regular y una evaluación final, el equipo puede garantizar que el producto cumple con los requisitos y ofrece valor a los usuarios.

## Valoracion económica y Humana

### Valoración Económica

La valoración económica incluye todos los costos asociados con el proyecto, desde la contratación de personal hasta los gastos de infraestructura y licencias de software. Aquí están los componentes clave a considerar:

#### 1. Costos del Personal

* **Salarios**: El costo principal para la mayoría de los proyectos de desarrollo de software es el salario del personal. En este proyecto, consideramos un equipo de 6 personas (desarrolladores y un diseñador UX/UI), un Product Owner y un Scrum Master. Supongamos que el salario promedio por miembro del equipo es de 30,000 € anuales (estos valores pueden variar según la ubicación geográfica y el nivel de experiencia).
  + **Costo Total de Salarios**: 2 personas \* 30,000€ \* (12 semanas / 52 semanas) = aproximadamente 12.000€

#### 2. Costos de Infraestructura

* **Hardware**: Computadoras, servidores, y otros equipos necesarios para el desarrollo del proyecto. Supongamos que cada desarrollador necesita una computadora de aproximadamente1,500€.
  + **Costo Total de Hardware**: 2 \* €1,500 = 3000€
* **Software y Herramientas**: Licencias para software de desarrollo, herramientas de colaboración, y bases de datos. El costo puede variar según las necesidades específicas del proyecto. Por ejemplo, licencias de Visual Studio Code y bases de datos MySQL son gratuitas, pero otras herramientas como Jira, Slack o servicios en la nube tienen costos asociados.
  + **Costo Estimado de Software**: Supongamos un costo total de 2,000€ para herramientas y software durante el proyecto.

#### 3. Costos Indirectos

* **Alquiler de Oficinas**: Espacio para el equipo de desarrollo, con un costo estimado de1,000€ por mes.
  + **Costo Total de Alquiler**: 1,000€ \* 3 meses = 3,000€
* **Costos Administrativos**: Gastos generales como suministros de oficina, servicios públicos, etc. Podría ser aproximadamente el 10% del costo total de salarios.
  + **Costo Administrativo**: 12000€ \* 0.10 = 1200€
* **Costos de Pruebas y Revisión**: Gastos para pruebas de usuario, eventos de revisión, etc. Supongamos un costo de1,000€.
  + **Costo Total de Pruebas**: 1,000€

#### 4. Costos Adicionales

* **Consultoría y Servicios Externos**: Cualquier servicio adicional o consultoría requerida para el proyecto. Supongamos un costo de 2,000€ para servicios de consultoría.
  + **Costo Total de Consultoría**: 2,000€

### Costos Totales

El costo total del proyecto es la suma de todos estos elementos. Aquí está la estimación:

* Salarios: 12.000€
* Hardware: 3000€
* Software y Herramientas: 2,000€
* Alquiler de Oficinas: 3,000€
* Costos Administrativos: 1200€
* Costos de Pruebas: 1,000€
* Consultoría: 2,000€

**Costo Total Estimado**: 24200€

### Valoración Humana

La valoración humana tiene que ver con el impacto en el equipo, el ambiente de trabajo y la colaboración. Aquí están algunos puntos clave:

#### 1. Equipo de Desarrollo

* **Experiencia y Habilidades**: Un equipo con las habilidades necesarias para el desarrollo del proyecto es fundamental. Se necesita experiencia en Node.js, MySQL y metodologías ágiles como Scrum.
* **Cooperación y Comunicación**: La capacidad del equipo para colaborar y comunicarse de manera efectiva es esencial para el éxito del proyecto.

#### 2. Product Owner y Scrum Master

* **Roles Claros**: El Product Owner debe ser capaz de definir y priorizar el backlog, mientras que el Scrum Master debe facilitar el proceso Scrum y eliminar impedimentos para el equipo.
* **Colaboración con el Equipo**: Ambos roles deben trabajar de cerca con el equipo de desarrollo y los stakeholders para garantizar que el proyecto cumpla con los objetivos.

#### 3. Cultura de Trabajo

* **Ambiente Positivo**: Fomentar una cultura de trabajo positiva y colaborativa ayuda a mejorar la moral del equipo y la calidad del producto final.
* **Equilibrio Trabajo-Vida**: El respeto por el tiempo personal y el equilibrio trabajo-vida puede contribuir a un ambiente de trabajo saludable.

La valoración económica y humana proporciona un panorama general de los costos y el impacto del proyecto. El costo estimado del proyecto es de aproximadamente $162,154, considerando salarios, infraestructura, gastos administrativos y otros costos asociados. La valoración humana destaca la importancia de un equipo experimentado, roles claramente definidos y un ambiente de trabajo positivo.

Ambas valoraciones son fundamentales para asegurar que el proyecto sea viable desde el punto de vista financiero y que tenga un impacto positivo en el equipo y en la calidad del producto.

## Conclusiones

Después de completar el desarrollo de la plataforma de gestión de backlog Scrum, es importante reflexionar sobre el proceso y considerar opciones de mejora para futuras iteraciones del proyecto. A continuación, se presentan algunas conclusiones clave, así como opciones de mejora identificadas durante el proceso de desarrollo:

### Opciones de Mejora

* **Ampliar Funcionalidades de Colaboración**:
* Si bien la plataforma cuenta con herramientas básicas de colaboración, como comentarios y asignación de tareas, existe la oportunidad de ampliar estas funcionalidades. Integrar funciones adicionales de comunicación en tiempo real, como mensajería instantánea y videollamadas, podría mejorar significativamente la colaboración entre los miembros del equipo. Esto facilitaría la comunicación y la resolución de problemas en tiempo real, lo que contribuiría a una mayor eficiencia y productividad.
* **Mejora en la Visualización del Backlog**:
  + La visualización del backlog es fundamental para que los usuarios comprendan el progreso del proyecto y las dependencias entre las tareas. Para mejorar esta área, se podría considerar la implementación de gráficos más avanzados, como diagramas de flujo o timelines interactivos. Estas visualizaciones proporcionarían una representación más clara y completa del estado del proyecto, lo que facilitaría la toma de decisiones y la planificación de futuras iteraciones.
* **Automatización de Tareas Repetitivas**:
  + Identificar y automatizar tareas repetitivas puede conducir a una mayor eficiencia y productividad en el equipo. La automatización de procesos, como la creación de informes o el envío de notificaciones, liberaría tiempo y recursos que podrían destinarse a actividades más estratégicas. Implementar herramientas y scripts de automatización podría ayudar a reducir el riesgo de errores humanos y agilizar las operaciones diarias del equipo.

#### Dificultades Encontradas y Soluciones Adoptadas

Durante el desarrollo del proyecto, surgieron diversas dificultades que requirieron soluciones creativas y efectivas para superarlas. A continuación, se describen algunas de las dificultades encontradas y las soluciones adoptadas:

1. **Resistencia al Cambio**:
   * Introducir una nueva plataforma puede generar resistencia al cambio entre algunos miembros del equipo. Para abordar esta dificultad, se llevaron a cabo sesiones de capacitación para familiarizar a los usuarios con la nueva plataforma y se designaron embajadores de cambio para apoyar la transición. Además, se destacaron los beneficios y las mejoras que la plataforma ofrecía en comparación con los métodos tradicionales, lo que ayudó a generar aceptación y compromiso por parte del equipo.
2. **Problemas de Integración con Herramientas Externas**:
   * Durante el desarrollo, surgieron desafíos relacionados con la integración de la plataforma con herramientas externas, como servicios de mensajería y control de versiones. Para superar estos problemas, se utilizó documentación detallada y APIs bien definidas para facilitar la integración. Además, se implementaron herramientas de middleware para actuar como intermediarios entre la plataforma y las herramientas externas, lo que permitió una integración más fluida y eficiente.
3. **Problemas de Rendimiento**:
   * A medida que la base de datos crecía y las consultas se volvían más complejas, surgieron problemas de rendimiento que afectaban la velocidad de respuesta de la plataforma. Para abordar esta dificultad, se optimizaron las consultas SQL mediante la identificación y eliminación de cuellos de botella. Además, se implementaron índices adecuados y se utilizaron técnicas de caché para mejorar la velocidad de acceso a los datos y garantizar un rendimiento óptimo de la plataforma.

#### Valoración Personal

El proyecto fue un éxito general, logrando los objetivos propuestos y entregando una plataforma funcional para la gestión de backlog Scrum. La comunicación efectiva dentro del equipo y la retroalimentación continua fueron clave para el éxito del proyecto.

La flexibilidad de la metodología Scrum permitió adaptarse a los cambios y superar las dificultades encontradas durante el desarrollo. La colaboración entre desarrolladores, Product Owner y Scrum Master ayudó a mantener el proyecto en marcha y a abordar problemas a medida que surgían.

### Análisis de Sostenibilidad

1. **Sostenibilidad Financiera**:
   * El modelo de negocio para la plataforma debe ser sostenible. Se podrían explorar opciones como suscripciones, licencias, o servicios premium para asegurar una fuente de ingresos constante.
2. **Sostenibilidad Técnica**:
   * Para garantizar la longevidad del proyecto, es importante adoptar tecnologías ampliamente utilizadas y mantener el código limpio y bien documentado. Esto facilita el mantenimiento y la expansión futura de la plataforma.
3. **Sostenibilidad Ambiental**:
   * Aunque el impacto ambiental de un proyecto de software es relativamente bajo, se puede minimizar mediante prácticas sostenibles, como el uso eficiente de recursos de servidores y la reducción de desechos electrónicos.
4. **Sostenibilidad Social**:
   * El proyecto puede tener un impacto social positivo al facilitar la colaboración y el trabajo en equipo. Fomentar un ambiente de trabajo positivo y diverso contribuye a la sostenibilidad social.

### Conclusión General

El proyecto de desarrollo de una plataforma para la gestión de backlog Scrum tuvo un resultado exitoso, con un producto funcional y útil para los equipos ágiles. Las opciones de mejora identificadas sugieren caminos para el crecimiento futuro, mientras que las dificultades encontradas y las soluciones adoptadas ofrecen lecciones valiosas para proyectos similares. La valoración personal y el análisis de sostenibilidad proporcionan una perspectiva sobre el impacto del proyecto y su viabilidad a largo plazo.

# Manual de Usuario

## Ambito y alcance

Hemos realizado una plataforma en línea diseñada para facilitar la creación, organización y gestión eficiente del Backlog de un proyecto utilizando la metodología ágil Scrum.

## Funcionalidades

### Login

#### Pasos para Hacer Login

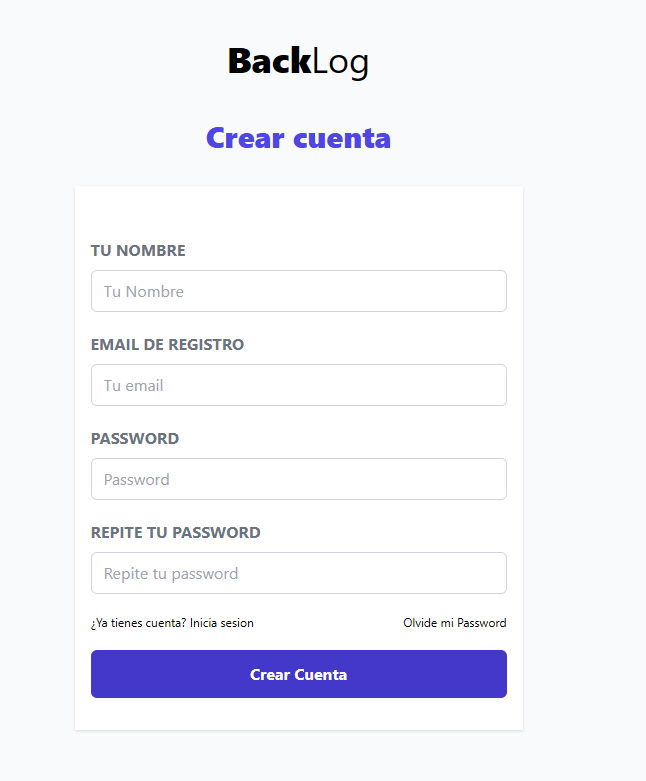
**Acceder a la Página de Login**:

Ve a la URL de la página web <http://nuestraapp.com/auth/login>

**Introducir Credenciales**:

En el formulario, introduce tu **nombre de usuario** o **correo electrónico** y tu **contraseña**. Asegúrate de escribir correctamente ambas credenciales, ya que son sensibles a mayúsculas y minúsculas.

Si aún no tienes una cuenta, tienes la opción de crear un nuevo usuario. Solo sigue las instrucciones y completa los datos necesarios para registrarte en la plataforma.



Una vez que hayas ingresado tus credenciales y presionado el botón de inicio de sesión, recibirás un correo electrónico de confirmación. Deberás hacer clic en el enlace proporcionado en el correo electrónico para confirmar tu cuenta. Sin la confirmación, no podrás acceder al sistema.

Si en algún momento olvidas tu contraseña, puedes utilizar la opción de recuperación de contraseña. Simplemente proporciona tu correo electrónico y recibirás un correo electrónico con instrucciones sobre cómo restablecer tu contraseña. Sigue las instrucciones proporcionadas en el correo electrónico para completar el proceso de recuperación de contraseña.

Principio del formulario



Al iniciar sesión en cualquier plataforma en línea, ya sea un sitio web, una aplicación o un servicio, se sigue un proceso que implica varios pasos clave para garantizar la seguridad y la autenticación del usuario.

**Introducción de Credenciales:**

**El primer paso es ingresar tus credenciales, que generalmente consisten en un nombre de usuario o correo electrónico y una contraseña. Estos datos son tu identificación personal y única que el sistema utiliza para reconocerte y autenticarte como usuario autorizado.**

**Envío de Credenciales al Servidor:**

Una vez que has introducido tus credenciales, haces clic en el botón de inicio de sesión. Este acto envía la información proporcionada al servidor donde se aloja la plataforma. La comunicación entre tu dispositivo y el servidor se realiza de manera segura a través de protocolos de cifrado para proteger la confidencialidad de tus datos.

**Verificación del Servidor**:

En el servidor, tus credenciales son verificadas comparándolas con la información almacenada en la base de datos. El sistema busca una coincidencia exacta entre la combinación de nombre de usuario/correo electrónico y contraseña que proporcionaste y los registros almacenados previamente. Si la información coincide, se considera una autenticación exitosa y se te permite el acceso a la plataforma.

**Acceso al Contenido Protegido**:

Una vez que la autenticación es exitosa, el servidor te redirige al área protegida del sitio web o la aplicación. Aquí es donde puedes acceder a funcionalidades exclusivas para usuarios registrados, como tu perfil personalizado, panel de control, o cualquier otra sección restringida a usuarios autorizados.

#### Consejos y Prácticas de Seguridad

**Uso de Contraseñas Seguras**:

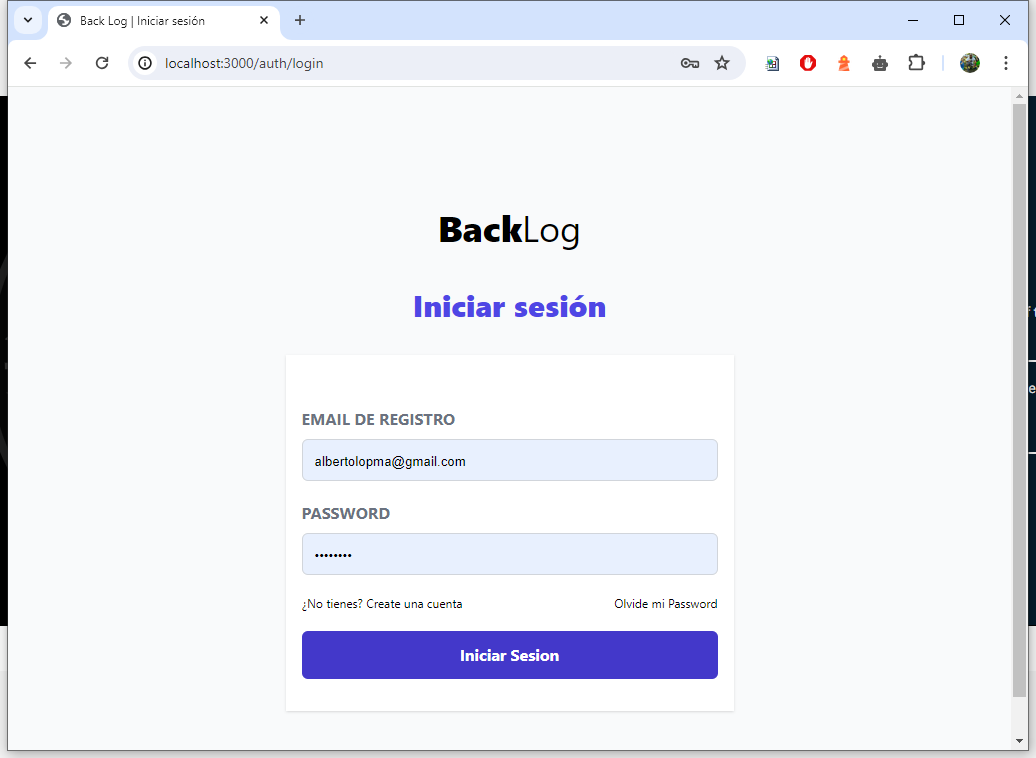
Se recomienda utilizar contraseñas robustas y únicas para cada plataforma. Una contraseña segura suele contener una combinación de letras mayúsculas y minúsculas, números y caracteres especiales. Además, evita utilizar información personal fácilmente deducible, como nombres o fechas de nacimiento.

**Proteger Información Sensible**:

Nunca compartas tus credenciales de inicio de sesión con nadie y evita guardar contraseñas en dispositivos no seguros o navegadores compartidos. Es fundamental mantener la confidencialidad de tus credenciales para proteger tu cuenta contra accesos no autorizados.

**Autenticación de Dos Factores (2FA):**

Considera activar la autenticación de dos factores siempre que sea posible. Esta capa adicional de seguridad requiere una segunda forma de verificación, como un código enviado a tu teléfono móvil, además de tu contraseña, para acceder a tu cuenta. Esto dificulta aún más el acceso no autorizado incluso si tus credenciales se ven comprometidas.

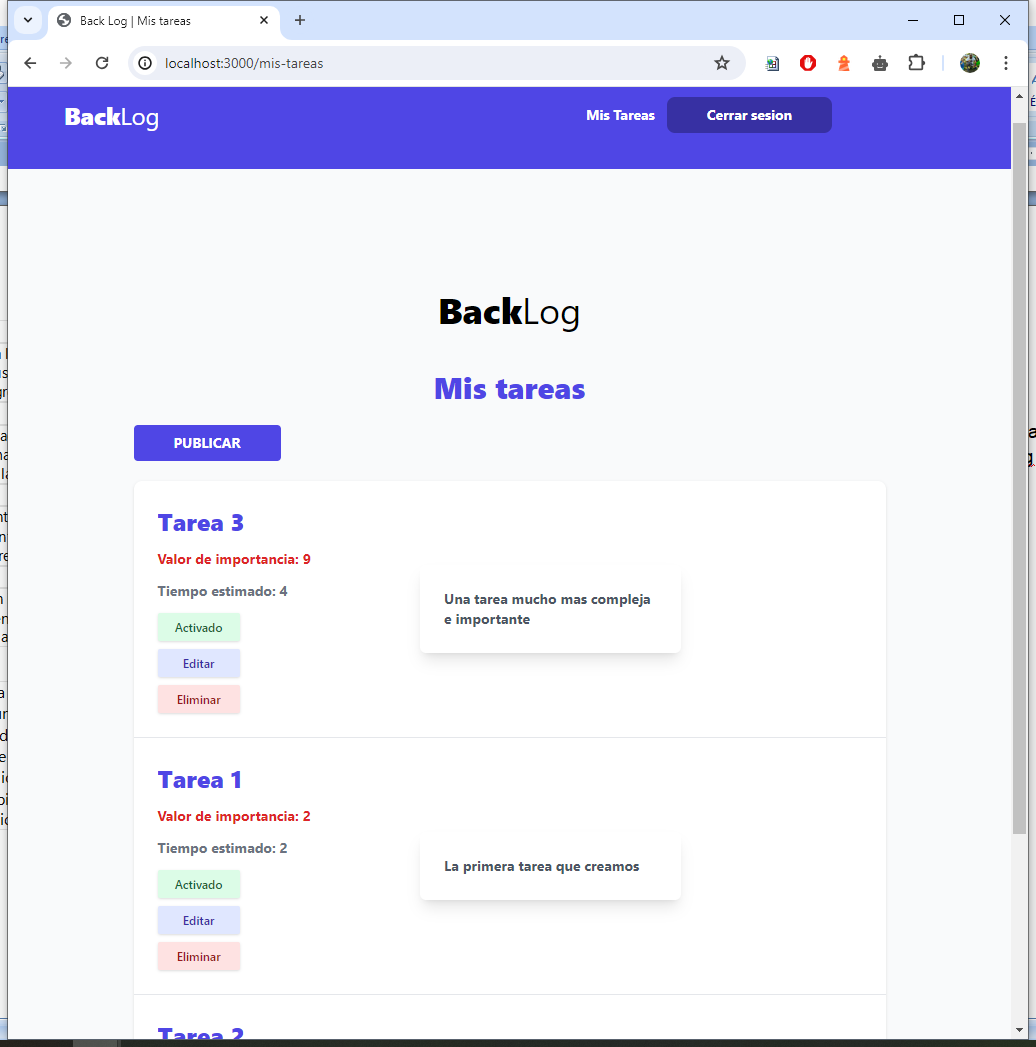


### Backlog

En esta pantalla, tenemos la capacidad de acceder a las tareas que hemos publicado previamente en la plataforma. Aquí, se nos muestra una lista que incluye información relevante sobre cada tarea:

1. **Importancia**: Se muestra el nivel de importancia asignado a cada tarea. Esta información ayuda a priorizar las tareas y enfocarse en aquellas que son más críticas o urgentes.
2. **Duración**: Indica la estimación de la duración o esfuerzo requerido para completar cada tarea. Esta información es útil para planificar y gestionar el tiempo de manera eficiente.
3. **Nombre**: El nombre de la tarea proporciona una breve descripción o título que identifica de manera única cada tarea en la lista.
4. **Descripción**: Se incluye una breve descripción de la tarea que brinda detalles adicionales sobre su objetivo, alcance o requisitos específicos. Esta descripción ayuda a entender mejor el propósito y los requisitos de cada tarea.

Además, desde esta pantalla, tenemos la opción de publicar nuevas tareas haciendo clic en el botón "PUBLICAR". Al hacer clic en este botón, la aplicación nos redirige a la página de creación de tarea, donde podemos ingresar los detalles necesarios para crear una nueva tarea. Esta página nos permite especificar la importancia, duración, nombre y descripción de la tarea, así como cualquier otra información relevante.

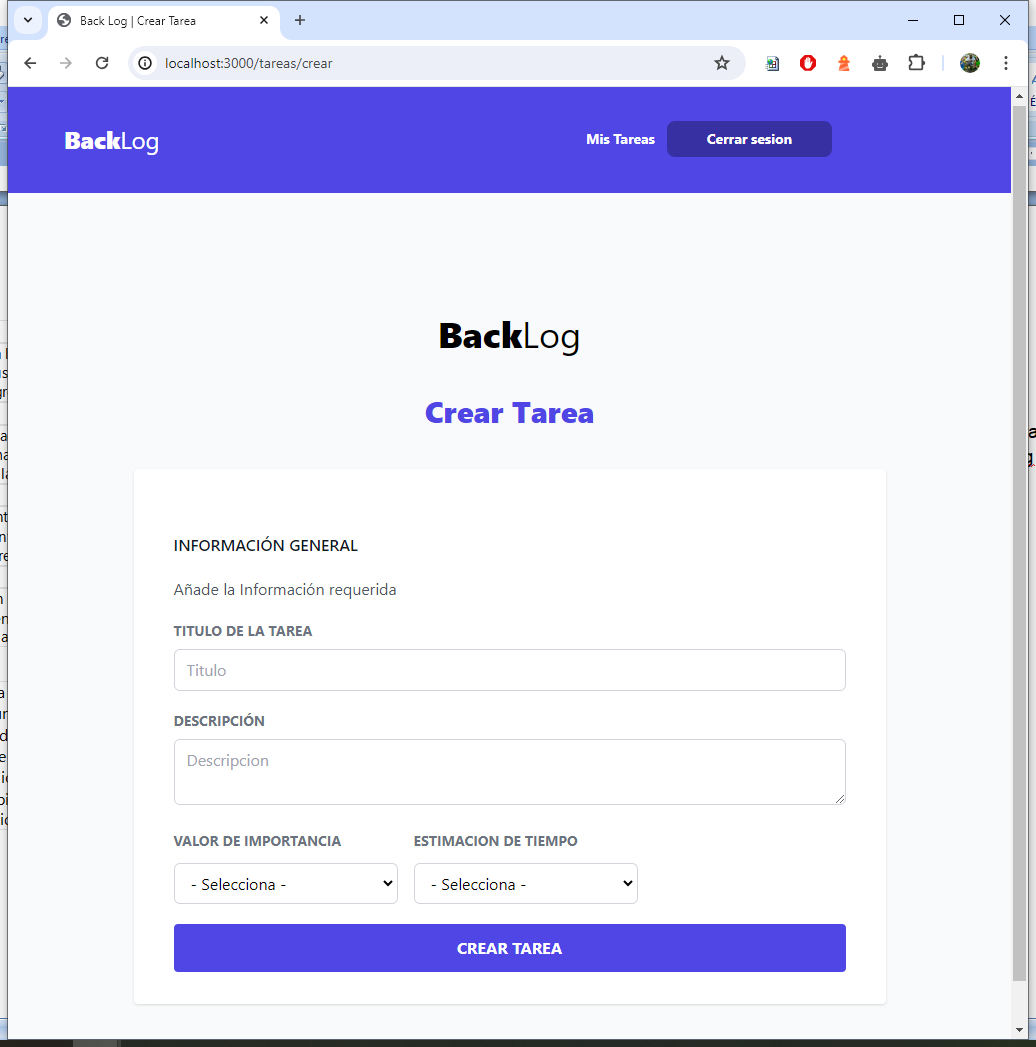


### Creación de Tarea

En esta pantalla, se nos brinda la capacidad de crear una nueva tarea que se añadirá a nuestro backlog. Para lograrlo, debemos proporcionar los datos necesarios para su creación, ingresando cada uno de los valores requeridos o seleccionándolos del desplegable cuando sea necesario.

Los campos que debemos completar para crear una nueva tarea pueden incluir:

1. **Nombre de la tarea**: Aquí ingresamos un título descriptivo que identifique claramente la tarea que estamos creando. Es importante que el nombre sea claro y conciso para facilitar su identificación y comprensión.
2. **Descripción**: En este campo, podemos proporcionar una descripción más detallada de la tarea, incluyendo información relevante sobre su objetivo, alcance, requisitos y cualquier otro detalle importante. La descripción nos ayuda a entender mejor la naturaleza y el propósito de la tarea.
3. **Valor de Importancia**: Seleccionamos el nivel de importancia de la tarea, lo que nos permite priorizarlas según su urgencia o impacto en el proyecto. Podemos elegir entre 1 al 9, dependiendo de la relevancia de la tarea.
4. **Estimación de tiempo**: Aquí especificamos la estimación del tiempo o esfuerzo necesario para completar la tarea. Esta estimación nos ayuda a planificar y gestionar el tiempo de manera eficiente, asegurándonos de asignar recursos adecuados a cada tarea.



Una vez que hemos proporcionado todos los datos necesarios, podemos hacer clic en el botón "Crear tarea" para agregar la nueva tarea a nuestro backlog. La aplicación procesará la información ingresada y añadirá la tarea con éxito, ordenando por mayor prioridad a menor prioridad en la lista de tareas, lo que nos permitirá seguir avanzando en nuestro proyecto y gestionar nuestras responsabilidades de manera efectiva. Una vez creada la tarea, volveremos a ver la pagina de Backlog, donde se reflejará la tarea recién creada.