UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL DE CÓRDOBA

# Ingeniería en Sistemas de Información

## Proyecto Final - 5K2

2024

# UTN ClassMate

## Sprint 0

**INTEGRANTES**

Ortiz Reverte, Santiago Javier - 85244

Pagés, Juan Ignacio - 78314

Rojo Birán, Agustín - 85681

**DOCENTES**

Iris Gastañaga

María Natalia Jaime

# TABLA DE CONTENIDOS

## INTRODUCCIÓN

El Sprint 0 representa la fase inicial donde se definen y estructuran los aspectos fundamentales que guiarán el desarrollo del proyecto. Desde la configuración del equipo de proyecto hasta la selección de herramientas y tecnologías involucradas, cada determinación en la presente entrega sentará inicialmente los pilares sobre los cuales se dará origen al Sistema de Información deseado.

Se tratarán definiciones tomadas sobre el presunto proyecto tales como la definición del equipo, el listado de historias de usuario inicial, la definición inicial del product backlog, la definición de un story map, determinación de la herramienta de software para la gestión del proyecto, la técnica de estimación a utilizar, la definición de la tecnología a utilizar en el desarrollo, pautas de codificación y testing, métricas de proyecto, SCM y un borrador de la arquitectura elegida.

**LISTADO DE HISTORIAS DE USUARIO (inicial)**

| Temas | User Stories |
|---|---|
| Gestión de foros | - Crear foro<br>- Buscar foro por nombre<br>- Ver foros<br>- Eliminar foro propio<br>- Modificar foro propio<br>- Ajustar privacidad de foro |
| Gestión de posts | - Crear post dentro de un foro<br>- Buscar post<br>- Eliminar post propio<br>- Modificar post propio<br>- Ver posts de un foro<br>- Indicar valoración a post<br>- Ver comentarAdjuntar imagen/es a post<br>- Adjuntar documento a post<br>- Adjuntar link a post<br>- Agregar mención hacia un usuario en post |
| Gestión de comentarios | - Agregar comentario a post<br>- Eliminar comentario a post<br>- Modificar comentario a post<br>- Ver comentarios a post<br>- Indicar valoración a comentario<br>- Adjuntar imagen/es a comentario<br>- Adjuntar documento a comentario<br>- Adjuntar link a comentario<br>- Agregar mención hacia un usuario en comentario |
| Gestión de chats | - Enviar mensaje privado a usuario específico(épica)<br>- Adjuntar documento en mensaje<br>- Adjuntar link en mensaje<br>- Adjuntar imágen en mensaje |
| Gestión de Calendario | - Ver calendario<br>- Agregar evento<br>- Modificar evento<br>- Eliminar evento |
| Gestión de Notificaciones | - Recibir notificación de evento<br>- Recibir notificación de mensaje chat<br>- Recibir notificación de comentario a post propio<br>- Recibir notificación de respuesta a comentario |

| | |
|---|---|
| | - Recibir notificación de valoración a post<br>- Recibir notificación de valoración a comentario<br>- Ajustar preferencias de notificaciones |
| Gestión de usuario | - Registrar cuenta<br>- Iniciar sesión<br>- Cerrar sesión<br>- Crear perfil<br>- Modificar perfil |

# DOCUMENTACIÓN DE GESTIÓN DE PROYECTO

## Definiciones Generales del Proyecto (Sprint 0)

### DEFINICIÓN DEL EQUIPO

| | |
|---|---|
| Juan Ignacio Pagés | Product Owner<br>Desarrollador FullStack |
| Santiago Ortiz Reverte | Scrum Master<br>Desarrollador FullStack |
| Agustín Rojo Birán | Scrum Master (rotación)<br>Desarrollador FullStack |

Tal como se muestra, el rol de Scrum Master se irá rotando de Sprint a Sprint.

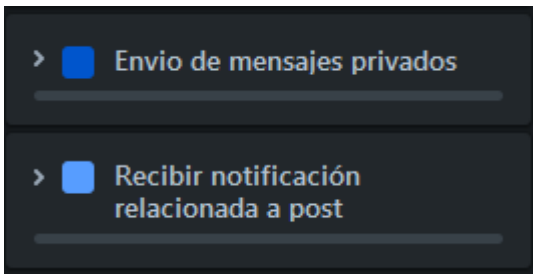
En un inicio, todos los representantes del equipo desempeñaremos el rol de desarrolladores, con el fin de que cada uno pueda ser capaz de construir porciones verticales de Software que den valor, sin depender verticalmente (valga la redundancia), de otro.

El PO escribirá los criterios de aceptación y pruebas de usuario de las User Stories al principio de cada sprint, además, al comienzo de cada sprint, debe priorizar el Product Backlog, de forma que refleje correctamente las necesidades actuales del equipo.

El SM documentará lo discutido en las diferentes ceremonias de scrum, se encargará de que el equipo asista a las mismas, y al finalizar el sprint analizará los resultados y métricas obtenidas, comunicándolas al equipo y documentándolas en el proceso.

## PRODUCT BACKLOG INICIAL


## Épicas:



**STORY MAP**

Link a Story Map*(6/4/2024)*

**HERRAMIENTA DE SOFTWARE PARA LA GESTIÓN DE PROYECTO**

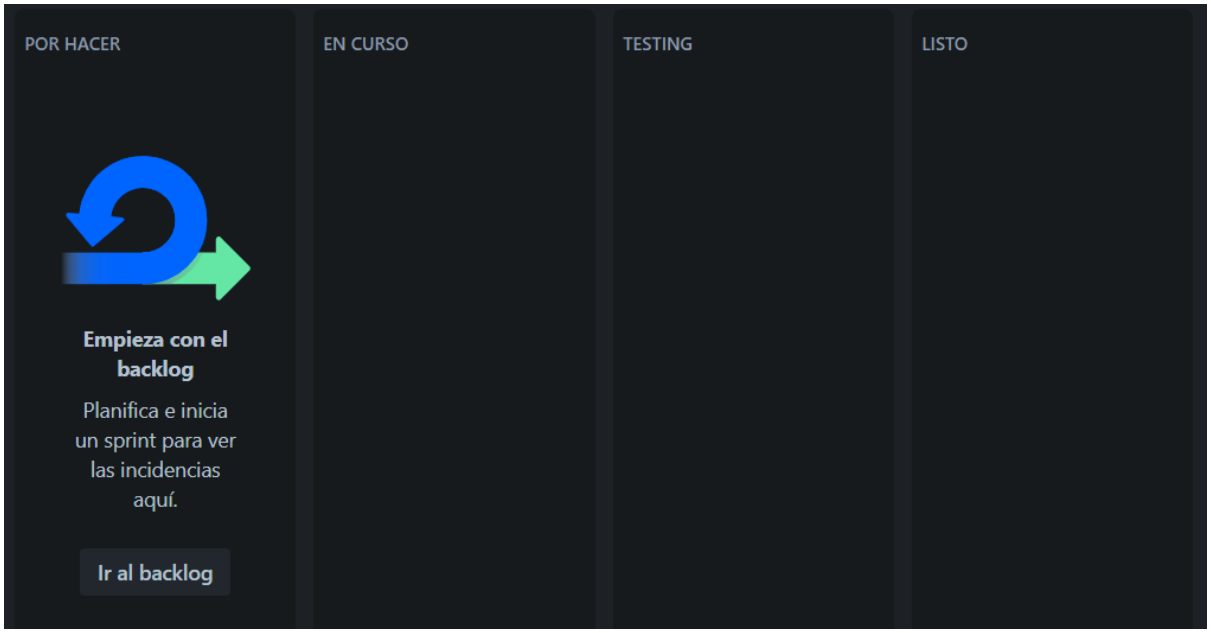
La herramienta Jira nos permitirá organizar y hacer un seguimiento ordenado del proyecto. Con ésta el Product Owner modificará el product backlog, agregando épicas, historias de usuarios, tareas, bugs, Spikes y mejoras. Además, esta herramienta otorga transparencia al workflow del equipo, a través del uso de una tabla **Kanban**.

Dada la situación recién mencionada, se remarca que se utilizará el framework Kanban, con el fin de lograr una mejora en la precisión y en el desarrollo de manera continua.

Vamos a contar con los siguientes ítems del Product Backlog:

- User Stories
- Historias técnicas
- Tareas
- Bugs
- Mejoras
- Spikes

Las historias de usuarios, historias técnicas, tareas, bugs y mejoras tendrán un ciclo de vida que empieza en **TO-DO,** esta puede ser tomada por cualquier miembro, pasándola a **IN PROGRESS**, al terminar de desarrollar el artefacto pasa a **TESTING**, y finalmente a DONE**.**




Los Spikes seguirán el mismo ciclo de vida que el resto de los ítems del Product Backlog, a excepción de la etapa de Testing, puesto que la misma no se aplica en este caso.

Se define a continuacion la Definition of Ready y Definition of Done para cada uno de los posibles ítems del Product Backlog.

Para US, tareas, bugs y mejoras se tiene:

**DEFINITION OF READY(DoR)**

Para la Definición de Preparado, se utiliza el criterio INVEST:

- **I**ndependent: La existencia e implementación de la User Story debe ser independiente del resto de historias.
- **N**egotiable: Las US deben ser descripciones cortas de funcionalidad cuyos detalles deben poder ser negociados entre desarrolladores y equipos de clientes.
- **V**aluable: Cada US debe otorgar valor de negocio al cliente.
- **E**stimable: Los desarrolladores deben poder estimar el tamaño de cada Story o la cantidad de tiempo que puede llevar implementarla.
- **S**mall: La historia debe ser de tamaño justo como para ser desarrollada e implementada en un Sprint.
- **T**estable: Las historias deben ser redactadas de forma que puedan ser probadas.

**DEFINITION OF DONE (DoD)**

Para la Definición de Listo, se utiliza el siguiente criterio:

- Pruebas de usuario pasadas en su totalidad.
- Casos de pruebas relacionados pasados.
- Criterios de aceptación cumplidos.
- Historia de Usuario aprobada por el PO.
- Funcionalidad incorporada a producción satisfactoriamente.

Para Spikes se tiene:

**DEFINITION OF READY(DoR)**

Una Spike se considera lista para ser trabajada cuando:

- Se han definido claramente los objetivos de investigación.
- Se ha establecido un límite de tiempo para la investigación.
- Se ha definido los criterios de éxito.

**DEFINITION OF DONE(DoD)**

Una Spike se considera completada cuando:

- Se han alcanzado los objetivos de investigación.
- Se han comunicado los hallazgos al equipo de desarrollo de manera clara y relevante.

Para historias técnicas se tiene:

**DEFINITION OF READY(DoR)**

Una historia técnica se considera lista para ser trabajada cuando:

- Se han definido claramente los requisitos técnicos.
- Se ha realizado un análisis de impacto adecuado.

**DEFINITION OF DONE(DoD)**

Una historia técnica se considera completada cuando:

- Se han implementado todos los cambios requeridos.
- Se ha realizado una revisión de código estricta.

El trabajo se coordinará permitiéndole a los miembros del equipo tomar items del product backlog del **to-do** en el tablero Kanban, habiendo definido en un principio el alcance del trabajo para todo el sprint. La comunicación se dará a través de un canal de slack, donde habrá recordatorios automáticos de todas las ceremonias de scrum (el scrum master se encargará de definirlas y asegurarse que estas sean usadas correctamente en el momento).

**TÉCNICA DE ESTIMACIÓN A UTILIZAR**

Para la estimación de las user stories, utilizaremos **Poker Estimation**, otorgándole un número 1, 2, 3, 5, 8, o 13 en base al esfuerzo en horas, complejidad e incertidumbre técnica y de negocio que conlleve la misma con relación a la user story canónica.

**DEFINICIÓN DE LA TECNOLOGÍA A UTILIZAR EN EL DESARROLLO**

En términos de lenguaje de programación, utilizaremos aquellos más conocidos por los miembros del grupo, **Java** para el back end y **Typescript** para el front end, además de lenguajes de etiquetado básicos como HTML y CSS.

La selección de **frameworks** se basó en buscar aquellos más robustos, que nos permitan una escalabilidad estable, que tengan soluciones ya planteadas para problemas comunes y una comunidad establecida. Además, tuvimos en cuenta que se usen en la industria para después poder desenvolvernos en un futuro empleo utilizando los mismos.

Framework para el back end: **Spring Boot**

Framework para el front end: **AngularJS**

Para la gestión de versionado se utilizará **Github**, ya que el equipo puede utilizarlo con facilidad.

**PAUTAS DE CODIFICACIÓN Y TESTING**

Para la codificación seguiremos los principios SOLID, y nos basaremos en los libros Clean Code y Clean Architecture para desarrollar un código legible y mantenible con la mínima cantidad de recursos posibles. *(6/4/2024)*

Además, en términos del análisis se emplearán los patrones GRASP, y en materia de diseño se aplicarán los principios SOLID y patrones de diseño desde un principio, haciendo de esta forma a la mantenibilidad y escalabilidad del proyecto en materia de Software.

En términos de Testing, se emplearán métodos de caja negra divididos en dos pasos:

- Identificación de clases de equivalencia
- Confección y ejecución de casos de prueba.

Plantilla de casos de prueba a utilizar, con ejemplos a manera ilustrativa *(6/4/2024)*

Los casos de prueba serán ejecutados por el PO al haber terminado cada US en forma completa (backend y frontend), generando en dicho proceso los datos de prueba, y registrando los resultados de ejecución en la plantilla mostrada.

Además, en el backend se programarán pruebas unitarias simulando las pruebas de usuario utilizando 2 frameworks complementarios, Junit y Mockito, esto será llevado a cabo por el desarrollador backend que esté trabajando en la US específica. Desarrollar los tests de esta manera servirá para determinar si el backend cumple con la funcionalidad esperada de cada US, luego de agregar cambios, ya que una vez programados los tests unitarios, pueden ser corridos automáticamente para detectar problemas. Esto simplemente quedará comentado de manera formal en el código y no se verá en la plantilla de casos de prueba, ya que esta solo reflejará pruebas en el momento que funcione el front end y el backend de manera conjunta.

**MÉTRICAS DE PROYECTO**

En base al peso de las user stories que vayamos terminando, Jira generará automáticamente la **velocidad** a la que vamos trabajando, y en base a esto podremos determinar la **capacidad** para trabajar en los próximos sprints.

En esta materia, emplearemos dos métricas de proyecto:

- **Velocity(velocidad):** Definida por la cantidad de Story Points completados por Sprint.
- **Capacity:** Definida como la cantidad de trabajo que puede completar el equipo en un Sprint determinado. A esta métrica la tomaremos de entrada para distribuir el trabajo al inicio de un Sprint

## GESTIÓN DE CONFIGURACIÓN DEL PROYECTO

A medida que el proyecto crece en tamaño, iremos actualizando la estructura física del repositorio en GitHub y las reglas de nombrado de los ítems de configuración.

Adjuntamos una definición inicial de ambos aspectos.

# ClassMate

Producto de Software que brinde una ayuda para la gestión y comunicación en ámbitos académicos para los alumnos de la UTN FRC.

## Estructura

La estructura física definida para el repositorio es la siguiente:

```
├── ClassMate
├── Documentacion
|  ├── Templates
|  ├── Entregas
|  |  ├── Editables
|  |  ├── Definitivas
|  ├── SprintX
|  |  ├── DiagramasDeArquitectura
|  |  ├── DiagramasDeClase
|  |  ├── DiagramasDeSecuencia
|  |  ├── ModeladoDeProcesos
├── LICENSE
├── Readme
```

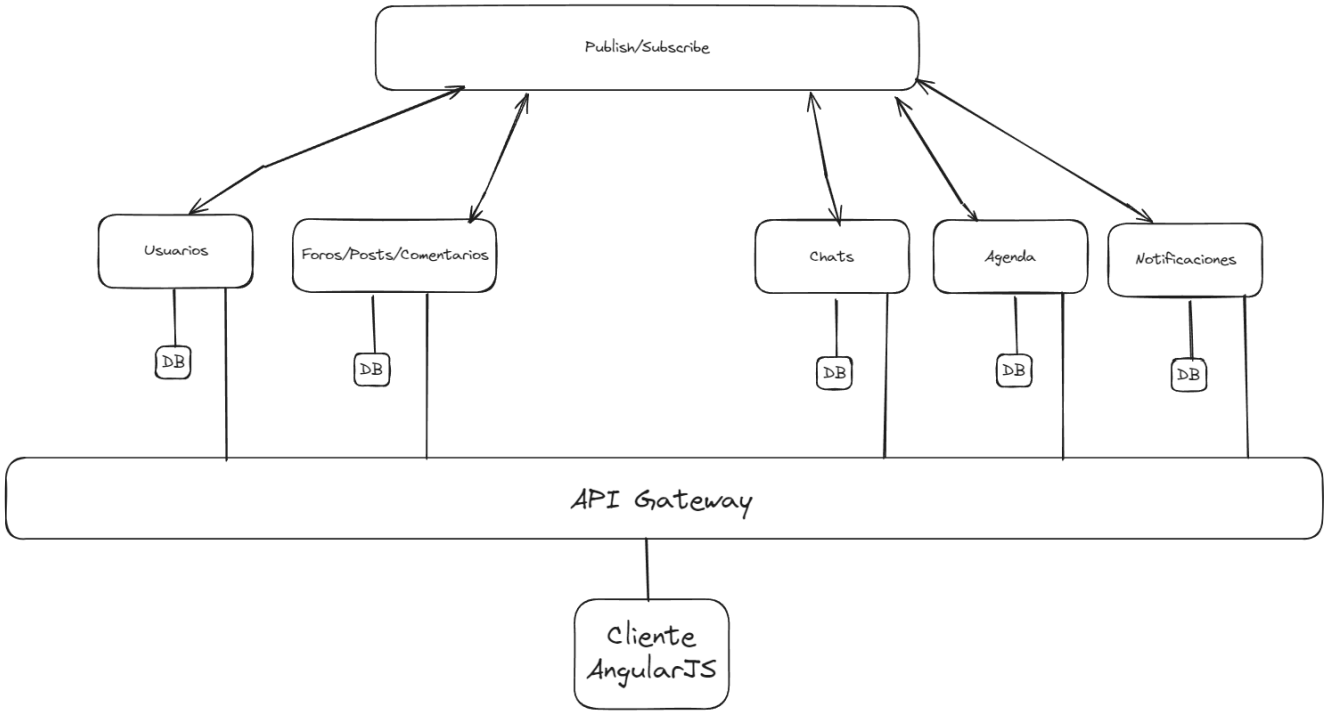
## Reglas de Nombrado de los Ítems de Configuración

| Nombre del Ítem de Configuración | Reglas de Nombrado | Ubicación |
|---|---|---|
| Diagrama de Arquitectura | DARQ-*.xml | /Documentacion/SprintX/DiagramasDeArquitectura |
| Diagrama de Clase | DC-*.eap | /Documentacion/SprintX/DiagramasDeClase |
| Diagrama de Secuencia | DSEC-*.eap | /Documentacion/SprintX/DiagramasDeSecuencia |
| Modelo de Proceso de Negocio | proceso_*.bpmn | /Documentacion/SprintX/ModeladoDeProcesos |
| Entrega editable | ENTREGA-X_*.docx | /Documentacion/Entregas/Editables |
| Entrega definitiva | ENTREGA-X_*.docx | /Documentacion/Entregas/Definitivas |
| Template de Documentación | TEMPLATE_*.docx ó TEMPLATE_*.xslx* | /Documentación/Templates |

## ARQUITECTURA

Incluimos un borrador inicial de la arquitectura basada en microservicios y diseñada mediante Domain Driven Design (DDD).

La comunicación entre los microservicios está pensada mediante el uso de eventos, siguiendo una arquitectura de tipo Event Driven Architecture.



## VERSIONADO

| Versión | Fecha | Descripción |
|---|---|---|
| 1.0.0 | 5/7/2024 | Primera Versión |
| 1.0.1 | 1/6/2024 | Espacios Arreglados |
| 1.0.2 | 4/6/2024 | Correcciones visuales, e información agregada relacionada al testing |