Acta de inicio Proyecto SGIV:

Índice

Metodología, Estándares y Estructura del proyecto 1
Canales de comunicación y herramientas: 1
Daily Meeting: 1
Code Style: 2
Commit Standards: 2
Configuración y Backlog: 3
JIRA 3
Base de Datos – PostgreSQL 4
Avance en Requerimientos R1, R2 y Testing 4
R1 – Gestión de Usuarios y Roles 4
R2 – Catalogación Dinámica de Destinos y Servicios 5
Testing – R1 y R2 5
Enlaces 5

Metodología, Estándares y Estructura del proyecto

Canales de comunicación y herramientas:

- *WhatsApp* (chat)
- *Discord* (meetings)
- *Google meet* (meetings, 2nd option)
- *Miro* (daily meeting notes)
- *Jira* (Project tracking software)

Daily Meeting:

Cada día se debe hacer una pequeña “ceremonia” en horas de la mañana. La **daily meeting** será de máximo 12 minutos, donde se converse y se escriba: Qué se hizo ayer, en que se trabajará hoy y qué obstáculos tiene. Se espera todo el equipo esté conectado vía Discord o Google meet (la hora de la reunión se definirá por WhatsApp). Para comodidad en las daily usaremos este MIRO: https://miro.com/app/board/uXjVKcRfNqs=/

Importante: Las reuniones de trabajo cooperativo o solución de bugs, etc., se llevarán a cabo en otro espacio y tiempo. Las daily son estrictamente para ver el proceso del desarrollo y soluciones puntuales

Code Style:

Para el código:

- Siempre deberán ir en inglés (Se hará refactoring con aquellas que estén en español).
- UpperCamelCase, para el nombrado de clases. Ejemplo: “TipoDestino”
- lowerCamelCase para las variables. Ejemplo: “idClient”
- SCREAMING_SNAKE_CASE para las constantes. Ejemplo: “MAX_RESERVATIONS”
- Se hará lo posible para que el nombre esté completo. Ejemplo “*modificationDate*”, en lugar de “*modifDate*”
- Se utilizará el estilo de identado K&R. Ejemplo:

```
while (x == y) {
    foo();
    bar();
}
```

Para las bases de datos:

- También siempre en inglés
- SCREAMING_SNAKE_CASE para los nombres de las **tablas**. Siempre MAYÚSCULAS y en plural.
- snake_case para los nombres de las **columnas**. Siempre minúscula y en singular.

Nombrado de documentos y carpetas:

- Se utilizará kebab-case

Commit Standards:

El commit puede/deberá incluir. *Todo en minúsculas*:

- **feat** – a new feature is introduced with the changes
- **fix** – a bug fix has occurred
- **chore** – changes that do not relate to a fix or feature and don't modify src or test files (for example updating dependencies)
- **refactor** – refactored code that neither fixes a bug nor adds a feature
- **docs** – updates to documentation such as a the README or other markdown files
- **style** – changes that do not affect the meaning of the code, likely related to code formatting such as white-space, missing semi-colons, and so on.

- **test** – including new or correcting previous tests
- **perf** – performance improvements
- **ci** – continuous integration related
- **build** – changes that affect the build system or external dependencies
- **revert** – reverts a previous commit

El cuerpo opcional del commit, debe utilizarse para proporcionar más detalles que no quepan dentro de las limitaciones de caracteres de la descripción de la línea de asunto.

Ejemplo commit convencional completo:

```
fix: fix foo to enable bar

This fixes the broken behavior of the component by doing xyz.
```

También puede ser importante indicar en que archivo o modulo se hizo el cambio principalmente, por lo que puede indicarse entre paréntesis. Ejemplo

```
feat(lang): add Polish language
```

Configuración y Backlog:

JIRA

Enlace al Jira Backlog y Board:

- https://icesi-viajes-app.atlassian.net/jira/software/projects/SGIV/boards/1/backlog
- https://icesi-viajes-app.atlassian.net/jira/software/projects/SGIV/boards/1

Responsable: *J. Lora*

Martes: Identificación a manera *superficial* de las **Historias de Usuario (HU)**. Tratar de que sean atómicas (no-divisibles en términos de funcionalidad). Siguen el formato RX, respecto a los requerimientos.

> Ejemplo: El requerimiento R2

Miércoles: Identificación total de las historias de usuario. Estas se guían por el siguiente estándar: Feature y Criterios de aceptación

Feature, que a su vez se divide en:

- **Como**: Persona que expresa una necesidad
- **Quiero/Necesito**: La necesidad requerida

- **Para**: Valor obtenido, contexto de la necesidad

Criterios de aceptación (máximo 4), que se dividen en:

- **Escenario** (Scenario): Es un ejemplo concreto que contiene una regla de negocio.
- **Dado** (Given): Se utiliza para describir el *contexto inicial del sistema*: la escena del escenario. Su objetivo es poner el sistema en un estado concreto antes de que el usuario (o sistema externo) comience a interactuar con el sistema (en los WHEN).
- **Cuando** (When): Se utiliza para describir un evento o una acción, desde una persona que interactúa con el sistema o un evento desencadenado por otro sistema.
- **Entonces** (Then): Se utiliza para describir el resultado esperado. La definición de un THEN debe usar una aserción para comparar el resultado real (lo que el sistema realmente hace) con el resultado esperado (lo que se supone que debe hacer el sistema).

Las **Historias de Usuario** deben ser *concisas*. **Ejemplo**:

> **Feature**: Búsqueda en Google
> ***Como*** usuario web, ***quiero*** buscar en Google ***para*** responder mis dudas.
> **Escenario**: Pantalla de búsqueda simple en Google
> **Dado**: Un navegador web en la página de Google
> **Cuando**: Se introduce la palabra de búsqueda “agile”
> **Entonces**: Se muestra el resultado de “agile”

Base de Datos – PostgreSQL

Responsable: *J. Lora*

La base de datos que se usará será PostgreSQL.

Avance en Requerimientos R1, R2 y Testing

Mientras se configura formalmente el Jira y las HU, se comenzará a trabajar para definir correctamente la base del proyecto en términos de código. Por lo cual se trabajará en los siguientes apartados:

R1 – Gestión de Usuarios y Roles

Responsable: *J. Brawn*

Gestión de Usuarios y Roles: Creación, modificación, y eliminación de usuarios y roles.

En esta etapa aún *no se trabajará* en la asignación de permisos específicos.

R2 – Catalogación Dinámica de Destinos y Servicios

Responsable: *J. Lora*

Catalogación Dinámica de Destinos y Servicios: Herramientas para añadir y categorizar destinos y servicios asociados, como alojamientos y transporte.

Testing – R1 y R2

Responsable: *C. Del Castillo*

Pruebas unitarias de los requerimientos R1 y R2 en el módulo de test. También cerciorarse de que estos se estén haciendo desde los test y no en el main

Enlaces

Enlace al **Jira** Backlog y Board:

https://icesi-viajes-app.atlassian.net/jira/software/projects/SGIV/boards/1/backlog

https://icesi-viajes-app.atlassian.net/jira/software/projects/SGIV/boards/1

Miro:

https://miro.com/app/board/uXjVKcRfNqs=/