**Logotipo

Descripción generada automáticamente**

|  |  |  |
| --- | --- | --- |
| **Nombre:** Marisol Orozco Orozco | | **Matrícula:** T02986821 |
| **Nombre del curso:**  Taller de productividad basada en herramientas tecnológicas | **Nombre del profesor:**  José Alfredo Jiménez Hernández | |
| **Módulo:**  **3ra Semana** | **Actividad:**  Fase 3 ejecucion | |
| **Fecha:** 25/04/2022 | | |
| **Bibliografía:** | | |

Fase III. Ejecución

**Descripción**

Basado en los requisitos del software previamente definido, crearás un cronograma de actividades para su diseño, desarrollo, pruebas e implementación.

**Objetivo**

Diseñar un plan de acción para el diseño, desarrollo, pruebas e implementación de los requisitos definidos dentro de la solución.

**Requerimientos**

Retroalimentación de la fase II, seleccionar una herramienta de administración de proyectos (gitlo, zube) e integrarla con el repositorio que contendrá el código de la integración (github o bitbucket).

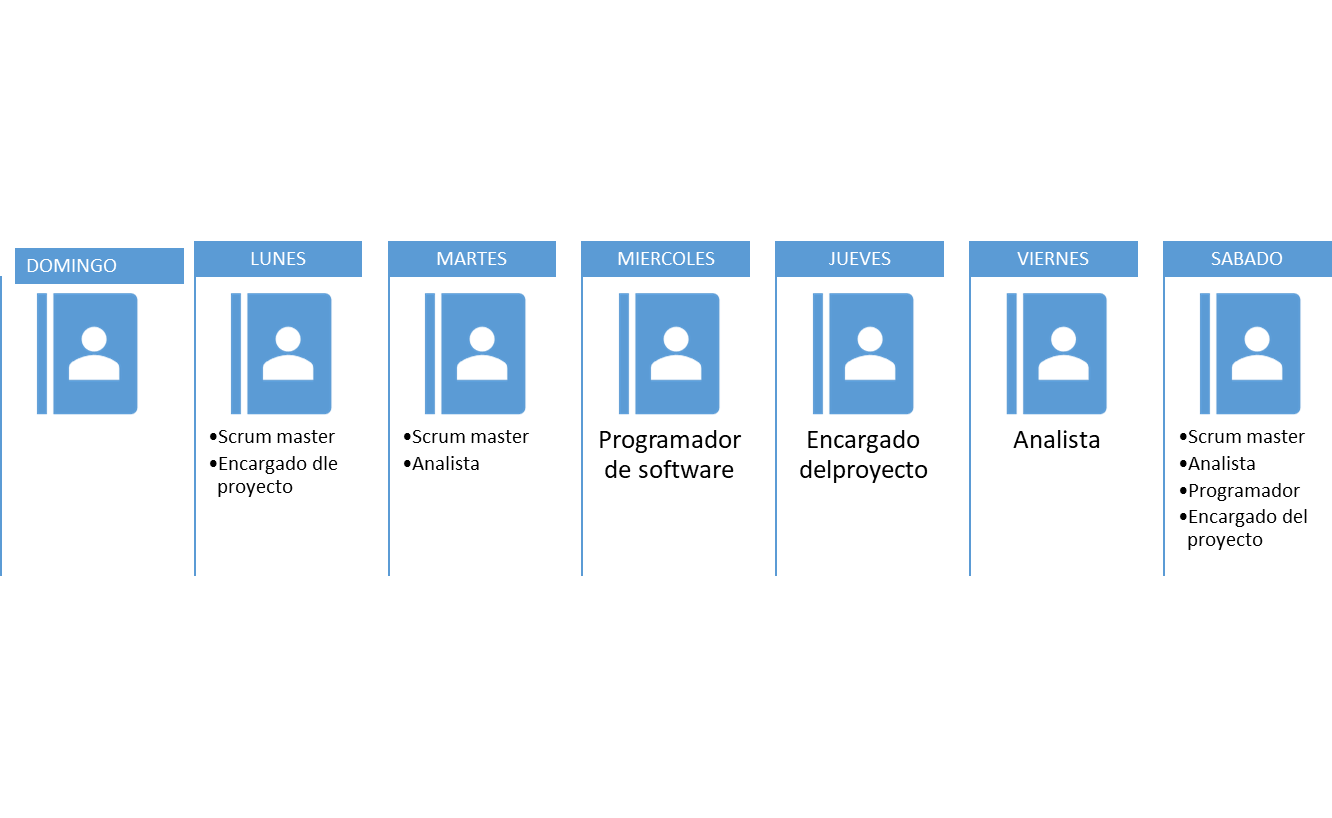
**Desarrollo**

Después de completar los requerimientos establecidos, realiza lo siguiente:

1. Mediante la integración del administrador de código (github) y el administrador de proyectos (gitlo/zube), crea las actividades a llevar a cabo, categorízalas con etiquetas y establece un tiempo estimado, esto con base en tu análisis previamente realizado. En trello, es importante agregar todo el detalle que sea posible, esto facilitará el análisis, desarrollo y solución del requerimiento, justificación, etc.





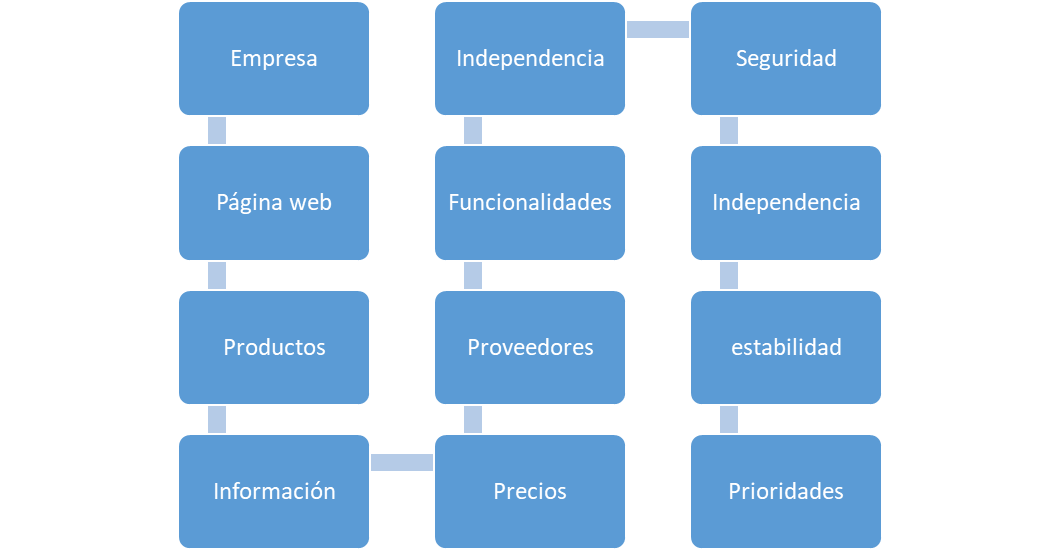


Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente



<https://github.com/momis19/fase-3-taller-.git>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Nota**: En estas tareas, es importante establecer lo que no se cubrirá y en la primera versión y categorizarla para futura referencia.

1. Agrupar las tareas en etapas (milestones), las cuales definirán la madurez del producto. Dentro del proyecto integrador se manejarán dos etapas: **Beta y General Availability** (**GA**). Beta se utilizará para agrupar todo el código que sea de desarrollo y GA, será el código final. Esto te ayudará a validar cuál es el progreso de tu trabajo dentro de la solución.

1. Hasta este momento con las tareas y las etapas definidas, cuentas con el programa.

1. De igual manera que durante el certificado, el código deberá ser administrado con dos branches principales: **develop** y **master.**

1. Cuando estés agregando el código de una tarea o requerimiento, es importante que generes un branch único y cuando esté listo lo agregues al branch **develop** por medio de **pull requests.**

**Nota**: Si los términos branch, issue, pull request, git, commit y otros no te son familiares, es recomendable revisar los recursos relacionados con git y la documentación de gitlab.

1. Seleccionarás un sistema de integración continua (travis-ci, drone) y conectarás tu repositorio para ejecutar al menos una prueba (JUnit) de tu código.

1. Arquitectura de la aplicación, la cual incluye los componentes necesarios, esto es servidor(es) de aplicación, servidor(es) web, repositorios, etc.

**Criterios de evaluación**

|  |  |
| --- | --- |
| **Criterios** | **Puntaje** |
| 1. Repositorio en GitHub o BitBucket con branches, requisitos como issues y en estos, la descripción, análisis y solución; etapas beta y rc como milestones. | 30 |
| 2. Integración de repositorio con Gitlo o Zube. | 25 |
| 3. Integración de repositorio con travis-ci o drone. | 25 |
| 4. Diagrama de la arquitectura de la solución. | 20 |
| **Puntaje total** | **100** |

**Entregables**

Documento con los siguientes elementos:

 Acceso al repositorio GitHub para validar contenido solicitado o capturas de pantalla de los issues con descripción, análisis y solución; milestones previamente definidos y branches. Código en el repositorio es esperado en esta etapa.

 Acceso al dashboard de administración de proyectos Gitlo (Trello) o Zube para validar el diagrama

 Acceso al dashboard del sistema de integración continua Travis-Ci o Drone con al menos una prueba JUnit.

 Diagrama de la arquitectura de la solución.