Análisis dinámico de una aplicación monolítica (ADAM)

Equipo:
Diego Alejandro Mateus Cruz
Federico Torres Mojica
William Andres Baquero Rojas

PLAN DE PROYECTO DE SOFTWARE

Departamento de ingeniería de sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
Noviembre 13, 2020

1. Historial de cambios

Cuadro 1: Historial de cambios del documento.

Fecha de cambio	Descripción del cambio	Responsable
09/11/20	Creación del documento	Diego Mateus
11/11/20	Desarrollo de la subsección de inicio del proyecto y la referenciación del modelo de ciclo de vida	William Baquero
12/11/20	Desarrollo de la subsección de administración de configuración y documentación	William Baquero
12/11/20	Desarrollo del Glosario	Diego Mateus
13/11/20	Desarrollo de la subsección de control de calidad	William Baquero
13/11/20	Desarrollo de la subsección métodos y herramientas de estimación, planes de trabajo del proyecto y referenciación de documentos externos	Diego Mateus
13/11/20	Desarrollo de la subsección supuestos y restricciones, organización del documento y organización, lenguajes y herramientas y monitoreo y control del proyecto	Federico Torres

2. Índice

Índice de cuadros

3. Índice de figuras

4. Vista general del proyecto

4.1. Visión del producto

Ver documento “Versión final de la propuesta de proyecto”, Sección 4.1 .Antecedentes, problema y solución propuesta”

4.2. Propósito, alcance y objetivos

Ver documento “Versión final de la propuesta de proyecto”, Sección 4.2 ”Descripción general del proyecto”

4.3. Supuestos y restricciones

Las restricciones generales del proyecto se enmarcan en el macroproyecto ”Towards Optimal Parameterized Partitioning of Microservices”desarrollado por los profesores Jaime Pavlich, Mariela Curiel y Carlos Parra. En este macroproyecto el análisis dinámico de aplicaciones monolíticas será uno de los cinco módulos que permitirán seleccionar las funcionalidades del sistema que serán candidatos a microservicios y que contarán con sus correspondientes interfaces de entrada y salida para comunicarse entre sí. La primera entrada será un grafo de particionamiento que será la salida del módulo de análisis estático y la aplicación monolítica que será ejecutada mediante el simulador de carga GEIST, WAGON o Bodik et al.

Los supuestos del proyecto es que la aplicación monolítica de entrada esté desarrollada en Spring Framework de tal forma que podamos hacer uso del correspondiente entorno de ejecución, y podamos rastrear adecuadamente las trazas de ejecución. Se tendrán en cuenta características del este entorno de desarrollo enfocado en aplicaciones empresariales modernas basadas en Java, como son las Core Technologies: inyección de dependencias, manejo de eventos, manejo de recursos, métodos de validación, enlace de datos, tipos de conversiones, entre otras. También se tendrá en cuenta acceso a datos por medio de transacciones, JDBC, ORM y Marshalling XML. Incluye también soporte para aplicaciones desarrolladas en Spring MVC y Spring WebFlux, ambos frameworks web.

4.4. Entregables

Ver documento “Versión final de la propuesta de proyecto”, Sección 6 ”Fases metodológicas”

4.5. Glosario

Ver anexo ”Glosario”.

5. Contexto del proyecto

5.1. Modelo de ciclo de vida

Ver documento ”Versión final de la propuesta de proyecto”, Sección 6 ”Fases metodológicas” [1]

5.2. Lenguajes y herramientas

Los lenguajes de programación soportados serán Java y Python por ser dos de los lenguajes de programación más usados en el mundo y que tienen una gran comunidad de programadores en que apoyarse en caso de que surjan errores o dificultades en el desarrollo. Adicionalmente, ofrecen diferentes enfoques y herramientas propias del lenguaje que en determinados casos ofrecerán distintos beneficios a la hora de solucionar problemas particulares. A continuación, se exponen las principales características de cada uno de los lenguajes:

- Tanto Java como Python son lenguajes orientados a objetos.
- Java es un lenguaje de programación de propósito general mientras que Python es un lenguaje de alto nivel usado en desarrollo web, inteligencia artificial, aprendizaje automático, automatización y otras aplicaciones de ciencia de datos.
- Tanto Java como Python son lenguajes de código libre.
- Java es un lenguaje de plataforma independiente mientras que python es de plataforma dependiente.
- Java es un lenguaje compilado y Python es un lenguaje interpretado.
- Java arroja errores a nivel de comilación como de ejecución mientras que python solo arroja errores de ejecución.
- Java es de tipado estático y python de tipado dinámico.
- Java soporta procesamiento multi hilo y python tiene un bloqueo de interprete global por lo que permite un solo hilo de ejecución a la vez.

5.3. Organización del proyecto y comunicación

En esta sección se definirán las principales entidades que influirán directa e indirectamente en el desarrollo del proyecto de software.

5.3.1. Organigrama

Figura 1: Organigrama

Cuadro 2: Roles del proyecto

Nombre de la entidad	Descripción	Respons.	Contacto
Diego Mateus	Desarrollador	Diseñar, documentar, desarrollar y probar	diego.mateus@javeriana.edu.co
Federico Torres Mojica	Desarrollador	Diseñar, documentar, desarrollar y probar	torres.federico@javeriana.edu.co
William Baquero	Desarrollador	Diseñar, documentar, desarrollar y probar	w.baquero@javeriana.edu.co

Jaime Pavlich	Gerente del Proyecto	Guiar a los desarrollado-res en la ejecución general del proyecto.	jpavlich@javeriana.edu.co
Mariela Curiel	Simuladores de carga	Guiar a los desarrollado-res en las herramientas a utilizar para la simulación de carga.	mcuriel@javeriana.edu.co
Carlos Parra	Parametrización de requerimientos	Determinar las restricciones de los datos de salida y que serán los datos de entrada de su módulo correspon-diente	caparraa@javeriana.edu.co

6. Administración del proyecto

6.1. Métodos y herramientas de estimación

Teniendo en cuenta que la principal metodología que se va a usar para la fase de desarrollo e implementaciones es XP, es prudente realizar una técnica de estimación ágil que permita estimar el proceso para descomponer en tareas más pequeñas según el esfuerzo para terminar cada una. El método de estimación que se adapta a estas características, y en el cual los miembros del equipo ya tienen experiencia es el Planning Poker [2].

Teniendo en cuenta que dada la situación en la cual se va a trabajar el proyecto, siendo esta la de la virtualidad, y dado que la herramienta de Trello se está usando para la organización de las tareas, se utilizara la funcionalidad que simula el Planning Poker en la misma [3].

Gracias a esta técnica se asegura que todos los miembros del equipo participen para dar su opinión y tener una idea más aproximada de las capacidades que se tienen con respecto a la realización del proyecto. Los pasos que se van a seguir según [2] son la discusión sobre las características de los springs y realización de la estimación, y Acordar entre todos la estimación del spring.

6.2. Inicio del proyecto

6.2.1. Entrenamiento personal

Para el inicio del proyecto es indispensable que los integrantes responsables de los módulos a desarrollar tengan conocimientos de las herramientas y lenguajes a utilizar para su respectivo desarrollo. Sin embargo, si algún integrante no cuenta con los conocimientos requeridos se le asignará un tiempo en el cual se procederá hacer su respectiva capacitación. Es por este motivo que las capacitaciones de los integrantes tendrán una gran prioridad ya que esto tendrá mejores resultados en cuanto a productividad y eficiencia en el desarrollo del software.

Para asegurar el buen uso del sistema por parte de los usuarios, este contara con un manual de usuario en el cual se mostrará el paso a paso para la ejecución de software y sus respectivos documentos de entrada los cuales son requeridos para su ejecución, con esto se lograra la capacitación por parte de los usuarios.

6.3. Planes de trabajo del proyecto

Para representar las distintas tareas que se realizaran a lo largo del proyecto se decidió organizarlas por distintas fases las cuales esta descritas en la sección 6 de [1], las tareas están visualizadas en la figura 2 por orden de procedencia. También, cabe recalcar que las fases están en orden de procedencia. Cumpliendo con este orden se espera que el resultado del proyecto sea exitoso.

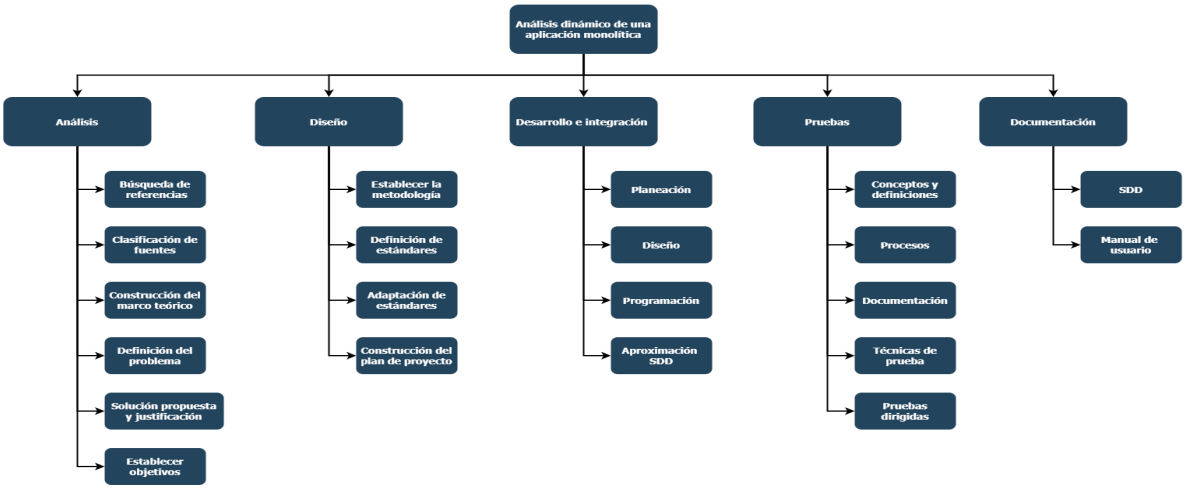

Figura 2: WBS general

Para expandir y tener un poco más de entendimiento acerca de las tareas de la fase de desarrollo e integración, se expandieron las tareas principales, como se muestra en la figura 3, haciendo énfasis y adaptando al proyecto las tareas que propone la metodología XP [4].

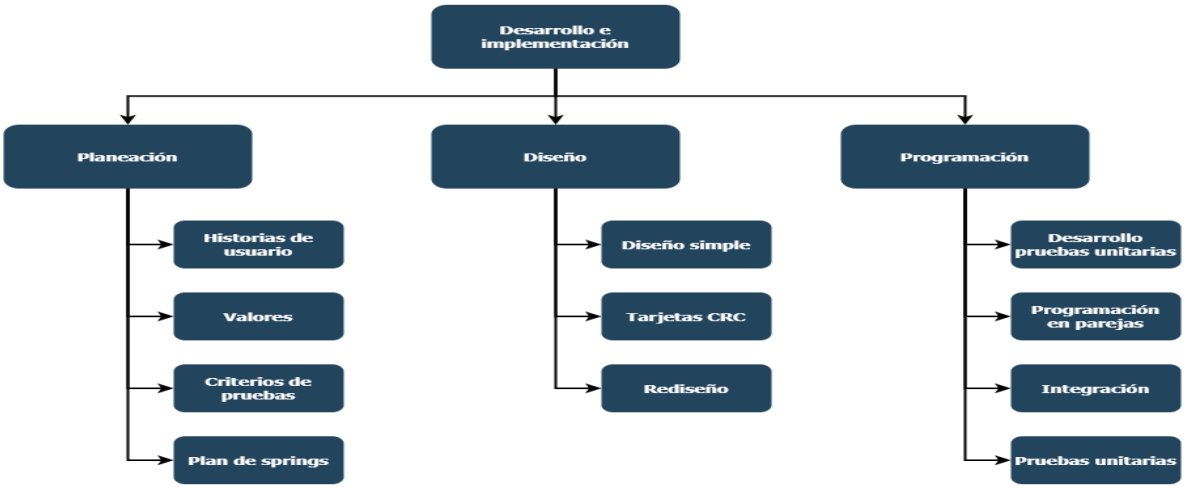

Figura 3: WBS desarrollo

En el anexo Çalendario"se puede identificar de una manera más clara y concisa las distintas fechas, tanto de inicio y finalización, de cada actividad presentadas anteriormente. Cabe que recalcar que las actividades de la fase de análisis y diseño ya se han realizado por lo que las fechas expuestas son verídicas. Por otro lado, para las futuras fases se realizó una estimación para cada una de ellas. Pudiendo ser modificadas. Para la parte de la fase de desarrollo e integración solo se visualiza una aproximación a lo que se espera de una ejecución de un spring, sabiendo que estos pueden demorar más o menos lo mismo en todas sus iteraciones.

7. Monitoreo y control del proyecto

Para el monitoreo y control general del proyecto se usará la metodología kanban enfocada en este aspecto para ofrecer metodologías basadas en el aprendizaje amplificado para realizar entregas sistemáticas del producto en la modalidad ”Justo a Tiempo”.

7.1. Administración de requisitos

Para detectar, reportar y controlar cambios en los requerimientos del producto se hará uso de las herramientas del modelo de metodologías ágiles de kanban en donde se hace del aprendizaje amplificado, que consiste en un proceso de aprendizaje continuo por medio de iteraciones. Esto quiere decir que el diseño de software se llevará a cabo mediante un proceso de solución de problemas que involucra a los desarrolladores escribiendo código, construyendo prototipos específicos del producto y verificando con el cliente que el producto cumpla las expectativas. De esta manera, se identifican rápidamente los cambios que pueda haber con respecto a la primera elicitación de requerimientos para hacer los cambios respectivos en lo que será el producto final.

Adicionalmente, se planea usar la estratégia ”Decidir tan tarde como sea posible”para manejar los problemas de incertidumbre asociados con el proyecto, que no permiten estimar desde un principio todos los cambios que puedan llegar a ocurrir en un momento dado. Para esto, lo que se busca es tomar decisiones basadas en hechos y no en suposiciones o predicciones. Se espera que entre mas complejo sea un producto, mayor capacidad de cambio debe tener, de tal forma que al demorar la toma de decisiones importantes y cruciales se pueden contemplar soluciones que abarquen la mayor cantidad de requisitos adicionales.

7.2. Monitoreo y control de progreso

Como se mencionó en la sección introductoria, se hará uso del modelo kanban para realizar pruebas sistemáticas en periodos de iteraciones semanales que permitan verificar la correcta integración del sistema con el macro proyecto y así garantizar que los requerimientos de integración con el mismo se vayan cumpliendo a medida que avanza el desarrollo general del producto.

Para llevar a cabo el monitoreo y control de avances de proyecto se tendrá en cuenta las metas programadas en el tablero de asignaciones de kanban y se verificará semanalmente que se esté cumpliendo con los objetivos propuestos.

8. Proceso de soporte

8.1. Ambiente de trabajo

Ver documento Reglas y acuerdos de trabajo de grado" [5]

8.2. Administración de configuración y documentación

Esta sección es un componente importante para garantizar la calidad de nuestro proyecto, ya que es el responsable de controlar los cambios propuestos tanto en el software como en los documentos de diseño. El proceso de la administración de configuración se va a definir en tres tareas, las cuales son:

- **Identificación de la Configuración:** Establecer estándares de documentación y un esquema de entendimiento de documentos.
- **Control de versiones:** Gestión de cambios que se realiza sobre los elementos de algún documento o código de software.
- **Control de cambios:** Evaluación y registro de todos los cambios que se realicen a la configuración del software tanto a los documentos como al código.
- **Artefactos de Documentación y Código**

8.2.1. Identificación de la Configuración

Para la configuración se tiene los siguientes objetivos: definir de forma clara la estructura de los documentos, relacionar los cambios con "quién, qué, cuándo, porqué, cómo" para facilitar el control de estos y establecer métodos para la revisión y control de los posibles cambios.

Asimismo, los entregables cuentan con una configuración ya establecida como cualquier aspecto asociado con un proyecto de software el cual está bajo unos lineamientos, como son los requerimientos del sistema, los cuales se administrarán y controlaran tanto en los cambios como en las versiones.

8.2.2. Control de versiones

Con la intención de hacer seguimiento a las diferentes versiones de los componentes de software. Se definirá un formato de referenciación para el desarrollo de los componentes del sistema. El cual tiene el siguiente formato **ADAM-MMCC-NN-VV** donde:

- **ADAM**: Identificador del proyecto.
- **MM**: Identificador del módulo.
- **CC**: Identificador del componente.

- **NN**: Identificador del integrante responsable del desarrollo del componente.
    - **WB**: William Baquero
    - **DM**: Diego Mateus
    - **FT**: Federico Torres
- **VV**: Identificador de la versión correspondiente.

8.2.3. Control de cambios

En esta sección se regulará que los cambios se apliquen al sistema de forma controlada, priorizando los cambios necesarios más urgentes y de mayor impacto para el desarrollo del sistema.

El control de cambios es una herramienta para la evaluación y aprobación de los cambios hechos a elementos de la configuración software durante el ciclo de vida. Estos pueden establecerse de dos distintas formas, las cuales son:

- **Control individual:** El cual se hace por parte del integrante responsable de ese cambio.
- **Control grupal:** El cual se requiere revisión y aprobación del cambio por parte de otro integrante del grupo el cual no haya participado en el desarrollo de ese cambio.

8.2.4. Artefactos de Documentación y Código

Cuadro 3: Artefactos de Documentación y Código

Nombre	Descripción
LaTeX	Sistema para crear textos principalmente estructurados y/o con fórmulas matemáticas [6].
Herramientas	La administración de la documentación se llevará a cabo utilizando papeeria, overleaf y Microsoft teams.
GitHub	Git es un software de control de versiones distribuido. El control de versiones es una forma de guardar cambios a lo largo del tiempo sin sobrescribir versiones anteriores. [7].

8.3. Control de calidad

Cuadro 4: Tareas de control de calidad

Nombre	Fase metodológica	Integrante responsable	Responsable de control de calidad
Documentacion	Durante todas las fases metodologicas se desarrollara los diferentes documentos los cuales se regiran por los diferentes estandares ya establecidos	Todos los integrantes	Director de proyecto
Componentes de desarrollo	Estos procesos se van a desarrollar durante la fase de desarrollo e integración y la fase de pruebas	Integrante asignado	Integrante que no haya participado en el desarrollo del componente

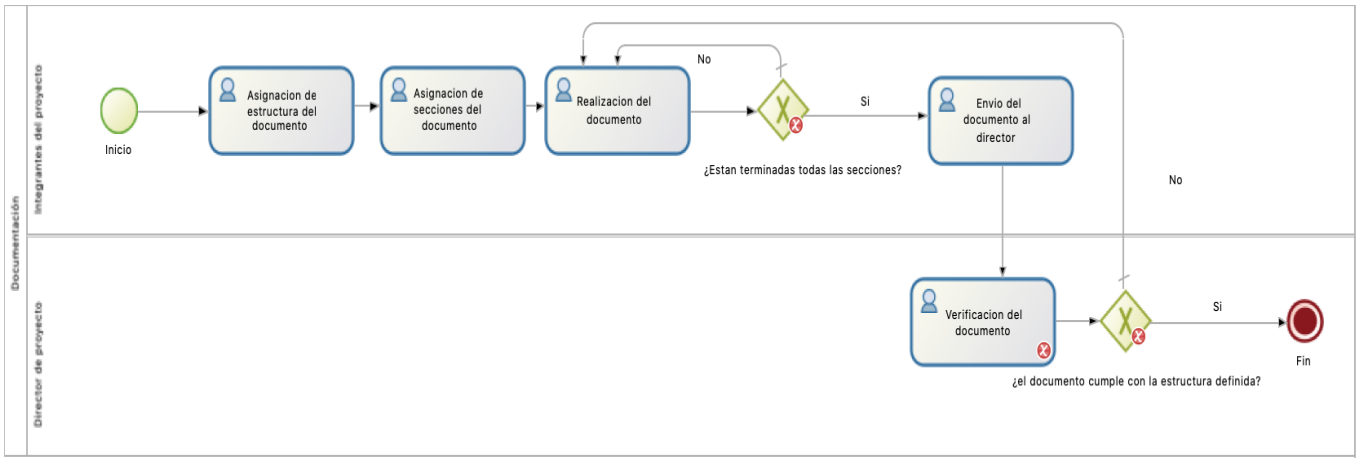

Figura 4: Diagrama BPMN de documentacion

El proceso comienza por definición, asignación y realización del documento, luego se verifica su contenido este presentado con el formato de la plantilla previamente establecida. y Si todo cumple se procede a enviar esa versión al director del proyecto el cual aprobara o recomendara hacerle los cambios correspondientes para la finalización del documento.

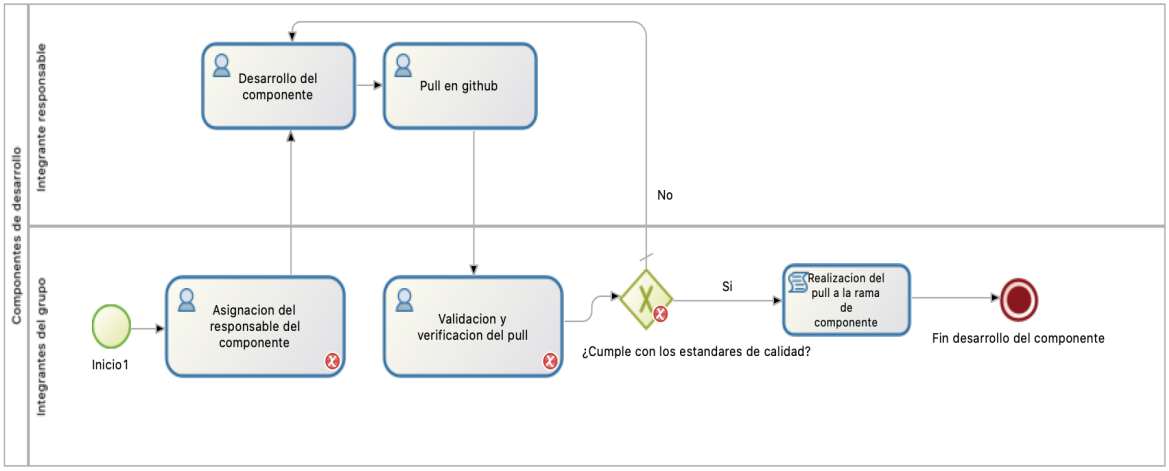

Figura 5: Diagrama BPMN de desarrollo de componentes

Para este proceso se inicia por la asignación del integrante responsable a desarrollar el componente. Sin embargo, para la finalización y cierre del proceso de desarrollo, este debe ser previamente revisado por un integrante del grupo que no haya participado en el desarrollo, verificando y validando las buenas prácticas de programación.

9. Referencias

[1] F. T. William Baquero, Diego Mateus, “Versión final de la propuesta de proyecto,” Noviembre 2020.

[2] “PlanningPoker.com - Estimates Made Easy. Sprints Made Simple.” [Online]. Available: https://www.planningpoker.com/

[3] “Power-Ups | Trello.” [Online]. Available: https://trello.com/power-ups/597cbecff4fe5f1d91d4b614/planning-poker

[4] “Funcionamiento - Metodología XP.” [Online]. Available: https://sites.google.com/site/xpmetodologia/marco-teorico/funcionamiento

[5] F. T. William Baquero, Diego Mateus, “Reglas y acuerdos de trabajo de grado,” Agosto 2020.

[6] “Usa LaTeX para tus informes,” 2003-05. [Online]. Available: https://users.dcc.uchile.cl/~jbarrios/latex/

[7] “What is Git?” 2020. [Online]. Available: https://github.com/git-guides