Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

Flujo de trabajo para pequeños equipos de desarrollo utilizando FDD y GitHub

Antonio David Ruiz Diaz Medina[1]
Universidad Nacional de Canindeyú - Paraguay
Nazario Luis Ayala Frasnelli[2]
Universidad Nacional de Canindeyú - Paraguay
Angel Gustavo Heimann Fernández[3]
Universidad Nacional de Canindeyú - Paraguay
Fátima Aidee Cáceres Urdapilleta[4]
Universidad Nacional de Canindeyú - Paraguay
Carlos Cesar Antonio Golin Galeano[5]
Universidad Nacional de Canindeyú - Paraguay
Alicia López Villalba[6]
Universidad Nacional de Canindeyú - Paraguay

Recibido: 21/12/2021
Aprobado: 15/01/2022

Resumen

Una de las características de las metodologías ágiles que resulta más atrayente para los equipos de desarrollo es su naturaleza iterativa y flexible ante cambios. Aceptar que los requerimientos cambian permite un mayor refinamiento, pero a su vez también trae consigo el aumento de la complejidad en la gestión y planificación de los proyectos. La metodología desarrollo orientado por características *(FDD)* surge como una posible

[1] Lic. en Análisis de Sistemas. Universidad Nacional de Canindeyú – Facultad de Ciencias y Tecnología. davidruizdiaz@facitec.edu.py.
[2] Lic. en Análisis de Sistemas. Universidad Nacional de Canindeyú – Facultad de Ciencias y Tecnología. nazarioayala@facitec.edu.py.
[3] Lic. en Análisis de Sistemas. Universidad Nacional de Canindeyú – Facultad de Ciencias y Tecnología. angelheimann@facitec.edu.py.
[4] Lic. en Análisis de Sistemas. Universidad Nacional de Canindeyú – Facultad de Ciencias y Tecnología. fatimacaceres@facitec.edu.py.
[5] Lic. en Análisis de Sistemas. Universidad Nacional de Canindeyú – Facultad de Ciencias y Tecnología. carlosgolin@unican.edu.py.
[6] Lic. en Análisis de Sistemas. Universidad Nacional de Canindeyú – Facultad de Ciencias y Tecnología. aliciavillalba593@gmail.com.

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

solución a la problemática con planificaciones a corto plazo orientadas a características específicas a ser implementadas. Entonces nace la pregunta de cómo integrar esta metodología a los procesos de una manera práctica. Es aquí donde aparecen las *feature branches* que permiten enfocar las características como ramas de un repositorio *Git*. Partiendo del concepto de *feature branches*, se propone un flujo de trabajo para aplicar *FDD* con *GitHub*, orientado a pequeños equipos de desarrollo. El trabajo cuenta con tres fases: primeramente, se describe cómo integrar *GitHub* con la metodología *FDD*; seguidamente, se diseñó un flujo de trabajo que permite aplicar *FDD* utilizando las funcionalidades de *GitHub*; y, por último, a fin de validar el flujo propuesto, se describe un caso de estudio en el que se presenta un escenario en el que un número equipo pequeño está a cargo de un proyecto de desarrollo de software. Según los resultados presentados, algunas de las funcionalidades de *GitHub* dan soporte en la aplicación fases de la metodología *FDD* abordadas siguiendo los procesos propuestos en el flujo de trabajo.

Palabras clave: Informática y desarrollo-Ingeniería de software-Metodologías ágiles-Desarrollo orientado por características.

Ñe'êmbyky

Peteĩ mba'e oguerekóva *metodologías ágiles*, ha umi *software*-apohára oiporuse hağua hembiapópe, ha'e ikatuha omboja'o hikuái hembiapo ambue michivevape ha ohechakuaágui aveí upe *cliente* oipotávaekue kuehe ikatu ko árape ndaha'evéima upeichaite. Ojehechakuáramo *cliente* oipotáva ikatuha okambia ikatu umi software ojejapóva osẽ porãve ha ojapo porãve hembiaporã, upéicha avei, upevarupi gestión de proyecto apohára rembiapo ijetu'uve há hasyve. Metodología de desarrollo orientado por características *(FDD)* ikatu oipytyvõ oñemomichĩve ha ojegueropu'aka porãve hağua upe *problema*, upeva ikatu *metodología FDD* omboja'órupi tembiapo guasu tembiapo michivévape. Umi tembiapo ojejapo ojehupyty hağua peteĩ mba'e cliente oipotáva *software* ojapo. Upéicha avei, oĩ *feature branches* he'íva umi *cliente* oipotáva software ojapo ikatuha ojejapo umi *ramas*, *repositório Git*-pegua, rupive. Ko tembiapópe ojehechauka mba'éichapa ikatu ojeiporu *metodología FDD* ha *feature branches*, *software*-apope, *GitHub* rupive. Ñepyrurãitépe, ojeheka kuri mba'e mba'épa oguereko

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

GitHub ikatúva ojeiporu *metodología FDD*-ndive. Upevaerã ojehesa'ỹijo haipyre omombe'uva mba'épa oguereko ha mba'éichapa ojeiporu *GitHub*. Upe rire, ojeiporu upe ojejuhúvaekue ojejapo hağua ta'anga *flujo de trabajo* rehegua ohechaukáva mba'éichapa ikatu ojeiporu *FDD* oñembaapo hağua, *GitHub* rupive, oñemotenondévo umi *proyecto de software*. Ipahaitépe, ojehechauka hağua oimerõ oiko ko'ã mba'e ko tembiapo ryepype ojehechaukáva, ojejapo peteĩ *caso de estudio* oje'eha ikatuhağuáicha opavave oipotáva oha'ã hağua ikatúpa ojeiporú térã ou porãtapa hembiapokuéra ryepýpe. Ko tembiapópe ojehechauka ikatuha, upe *flujo de trabajo* rupive, GitHub oipytyvõ umi *software*-apoharápe oiporu hağua *FDD* hembiapope.

Ñe'e tekotevéva: Informática y desarrollo-Ingeniería de software-Metodologías ágiles-Desarrollo orientado por características.

INTRODUCCIÓN

La gestión y la planeación en proyectos ágiles por su propia naturaleza cambiante, adaptativa e incremental, se vuelve extremadamente difícil. La variabilidad de la realidad es un aspecto que gana gran importancia en los paradigmas ágiles y esto provoca la incorporación de una característica no pequeña que es el dinamismo en los requisitos. Si bien permite que los resultados sean más acordes a las necesidades reales, su gestión se vuelve una pesadilla [1].

En las metodologías tradicionales como la *Waterfall*, al menos desde un punto de vista de gestión del proyecto, la planificación de las actividades es mucho más sencilla. Al retirar todos los aspectos variables del proyecto se vuelven más claros ciertos aspectos como qué ya se hizo, quién lo hizo, por qué lo hizo y qué se debe hacer. Pero, la realidad cambia los seres humanos no son infalibles, por lo que, o bien los requisitos pueden cambiar a lo largo del proyecto o pudieron cometerse errores en su especificación. Esto puede ocasionar serios problemas que solo se detectarían al final de la ejecución [1].

Es en este sentido se halla una dicotomía en la cual o se sacrifica la visión dinámica de los requisitos o, por el contrario, se mantiene aumentando, de este modo, la complejidad de los proyectos.

CIDUNAE
Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)
UNIVERSIDAD AUTÓNOMA DE ENCARNACIÓN

Una solución que puede ayudar a reducir la problemática es el desarrollo orientado a características. El FDD, por sus siglas en inglés *Feature Driven Development*, es un proceso para ayudar a los equipos a producir resultados de trabajo frecuentes y tangibles. Además, organiza el trabajo en iteraciones cortas para construir pequeñas funcionalidades que agregan valor al cliente [2].

El proceso de FDD consta de cinco fases que son realizadas de forma secuencial e iterativa para crear software de manera incremental. Estas fases son [3]:

1. Desarrollar un modelo general *(Develop an overall model)*: consiste en la definición del contexto y el alcance del proyecto a desarrollar.
2. Crear una lista de características *(Build a features list)*: se trata de una lista que incluye todas las características o funcionalidades que deben ser implementadas. Las características deben estar agrupadas según el dominio específico de cada una.
3. Planificar por característica *(Plan by feature)*: en esta fase, las prioridades se asignan a las características teniendo en cuenta las dependencias de las características, el riesgo involucrado, la complejidad y la carga de trabajo del equipo.
4. Diseño por característica *(Design by feature)*: consiste en producir iterativamente diseños de clases y diagramas de secuencias en base a las características que serán desarrolladas.
5. Construir la característica *(Build by feature)*: fase iterativa en la que se implementan y prueban las características diseñadas. Una vez probadas las características desarrolladas se integran a la compilación principal e inicia una nueva iteración.

La Figura 1 ilustra el proceso completo de FDD.

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

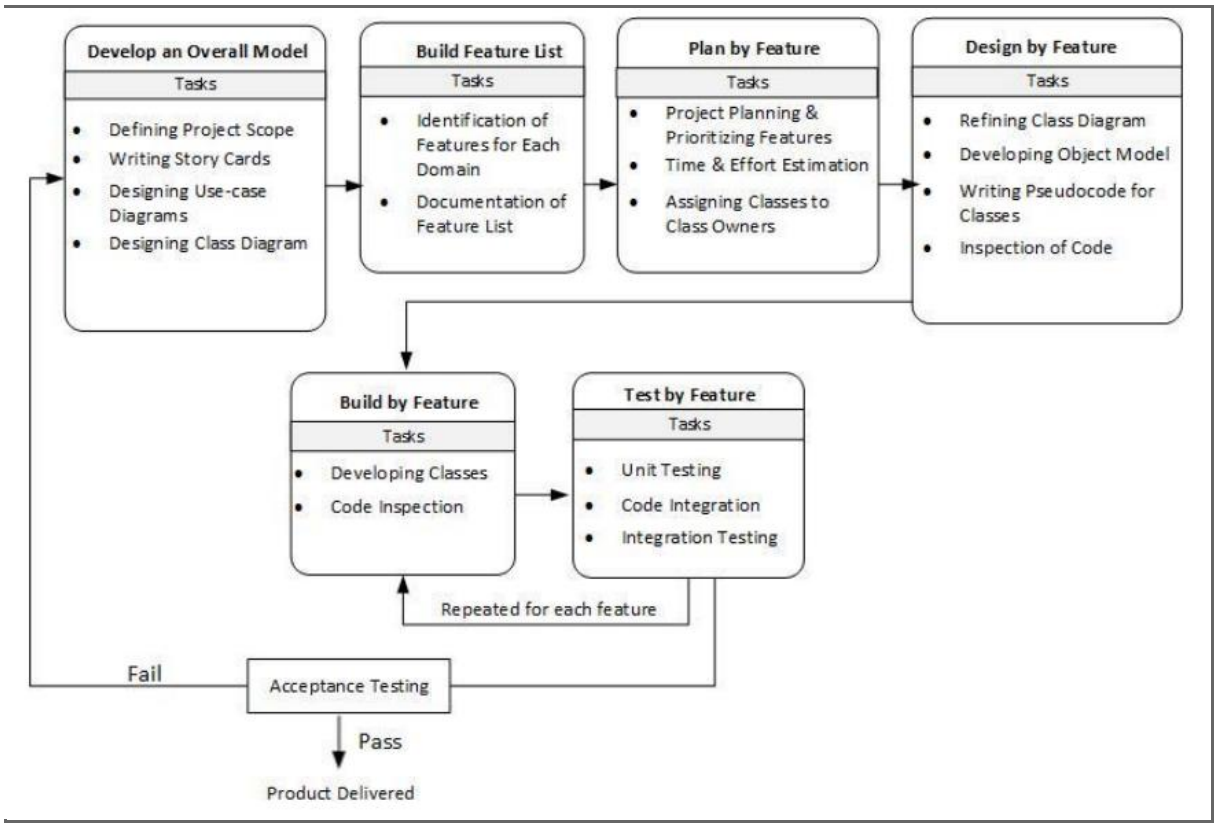


Figura 1. Proceso del modelo FDD [4].

Además de utilizar FDD cómo estrategia de desarrollo ágil, se busca integrar el modelo FDD a una técnica de desarrollo más empírica que también permite el desarrollo incremental de características a un software principal. La técnica de la que se habla es la *Feature Branches*, que básicamente se trata de crear una copia del código base principal para que un individuo o equipo de desarrolladores de software pueda trabajar en una nueva característica hasta que esté completa para, luego, integrar esta característica al código fuente original [5].

También, se busca integrar al proceso herramientas que ya existen y son de libre uso para que den soporte a ciertas etapas del modelo FDD.

ANTECEDENTES

La literatura muestra que existen varios estudios comparativos entre modelos de desarrollos que se realizan con la finalidad de conocer características de cada modelo e

CIDUNAE
Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)
UNAE UNIVERSIDAD AUTÓNOMA DE ENCARNACIÓN

intentar adaptar o integrar sus características. Tal es el caso del estudio realizado por A. F. Chowdhury y M. N. Huda [6], en que se proponen analizar las características del modelo de desarrollo adaptativo (ASD) y del modelo FDD. Por una parte, sus resultados muestran que el ASD no posee directrices claras y técnicas que ayuden a los desarrolladores en sus procesos. También mencionan que el objetivo principal del modelo ASD es la obtención del producto, dejando en segundo plano los detalles del proceso. Por otra parte, mencionan que el FDD tiene ventajas sobre el método anterior por contar con prácticas y técnicas recomendadas para cada fase de su proceso.

Cabe mencionar casos de estudios donde se describe la implementación exitosa del modelo FDD, como en el trabajo realizado por A. Firdaus, I. Ghani, y N. I. M. Yasin en [7]. El trabajo describe la implementación experimental del modelo FDD en un curso de desarrollo desde la perspectiva de seguridad. Los resultados obtenidos de las observaciones realizadas revelan que FDD puede ser aplicado a procesos de desarrollo de software seguros, pero también se discute la carencia que tiene FDD en cuanto a la definición clara de los procesos.

Sumado a lo anterior, M. Rychlý y P. Tichá [8] realizan el análisis, diseño e implementación de una aplicación web para dar soporte a los procesos de FDD. La aplicación cubría los cinco procesos de FDD, además de ofrecer un conjunto de informes útiles para conocer el progreso del proyecto.
Z. Nawaz [4] propone un proceso simplificado de FDD para pequeños y medianos equipos de desarrollo e identifica una serie de debilidades detectadas en las fases del clásico FDD. A partir de eso, propuso y detalló un modelo mejorado y adaptado, al cual denomina *Simplified Feature Driven Development* (SFDD), con fases más simples y utilizando técnicas de las metodologías ágiles.

DEFINICIÓN DEL PROBLEMA

Una de las características de las metodologías ágiles que resulta más atrayente para los equipos de desarrollos es su naturaleza iterativa y flexible ante los cambios. Esto permite un mayor refinamiento de los requerimientos, pero por otro lado esta misma naturaleza

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

cambiante aumenta la complejidad de la gestión de los proyectos y torna la planificación, el análisis de los riesgos y de otros aspectos de estimación todo un desafío [1], [9].

Por otro lado, es necesario analizar y actualizar algunos aspectos de las metodologías ágiles a la realidad actual de los proyectos de desarrollo de software. Uno de los principales factores a considerar en este sentido es la evolución de herramientas de gestión de procesos vinculados a los proyectos de desarrollo [9]. En ese sentido, es necesario analizar cómo integrar estas herramientas a los procesos ágiles para mejorarlos y/o adaptarlos a ciertos contextos de desarrollo de proyectos de software.

En este proyecto se busca integrar de forma coherente y lógica algunas de las fases del modelo FDD con prácticas y técnicas como la utilización de control de versiones con Git y las herramientas de gestión de proyectos de GitHub para obtener un material de apoyo al desarrollo de software de los pequeños equipos de desarrollo.

OBJETIVOS

- Proponer un flujo de trabajo orientado a pequeños equipos de desarrollo utilizando un modelo de desarrollo orientado por características aplicando control de versiones.
- Determinar funcionalidades de Github que puedan ser útiles en los procesos de FDD
- Obtener un diseño del flujo de trabajo que integre la metodología FDD con las herramientas ofrecidas por Github.
- Validar la propuesta a través de un caso de estudio.

METODOLOGÍA

Este trabajo se puede enmarcar como una investigación descriptiva de alcance exploratorio en la que se pretende realizar una descripción sobre la integración de la metodología FDD con procesos de control de versiones.

Módulos y funcionalidades de Github prácticos para FDD

CIDUNAE
Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)
UNIVERSIDAD AUTÓNOMA DE ENCARNACIÓN

A fin de determinar qué módulos y funcionalidades de GitHub son convenientes usar como apoyo a ciertos procesos de FDD se realizaron las siguientes acciones:

- Primero se realizó una revisión de la documentación de GitHub [10] con lo cual se pudo crear una lista de módulos y funcionalidades de la aplicación.
- Luego, por medio de un análisis empírico se relacionaron funcionalidades de Github con los procesos de FDD. Con esto se pudo obtener una lista de funcionalidades que pueden servir como herramienta de apoyo en ciertos procesos de FDD.

Proceso para diseño del flujo de trabajo

La revisión de la literatura sobre la metodología FDD ayudó a entender sus procesos y fases. Además, se analizaron otras propuestas de aplicación de la metodología que ayudaron a tener un mejor panorama de qué aspectos aprovechar para el mejoramiento de la gestión de proyectos de desarrollo de software.

Entendiendo el proceso y obtenida la lista de funcionalidades que potencialmente puedan ser útiles para resolver la problemática planteada en el trabajo, se procedió a diseñar un flujo de trabajo que permita aplicar la metodología FDD integrando las herramientas de gestión de proyectos de software que proporciona la plataforma GitHub.

Validación del flujo de trabajo

Para la validación del flujo de trabajo aplicado a equipos pequeños se planteó un escenario de desarrollo de software con un número reducido de involucrados. Los roles de la metodología FDD [3] seleccionados se enfocan a los procesos abordados por este trabajo y cabe resaltar que la adopción es hasta cierto punto flexible con relación a las características particulares de los equipos de desarrollo.

A continuación, se detallan los roles seleccionados para la confección del caso de estudio.

Tabla 1. Roles seleccionados para el caso de estudio.

CANT.	ROL	DESCRIPCIÓN

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

1	*Development Manager*	Supervisa la ejecución de las actividades de desarrollo diarias.
1	*Chief Programmer*	Lidera las actividades de construcción del software. Selecciona un conjunto de características que se deben construir y las asigna a los *class owners*.
3	*Class Owner*	Es el encargado de implementar el conjunto de características que se le fueron asignadas.

Tomando como base los roles seleccionados y el flujo de trabajo planteado, se describe un caso de estudio en cual se utilizan las herramientas proporcionadas por GitHub para la gestión de los procesos de un proyecto desarrollo de software, desde la planeación de las características a ser construidas hasta su incorporación al producto de software en producción.

RESULTADOS

Características seleccionadas de GitHub

En la Tabla 2 se presenta, de manera general, una lista de funcionalidades de GitHub que pueden ser integrados a los procesos de FDD. Sin embargo, cabe aclarar que esta lista no es definitiva y que existe flexibilidad en la elección de cuáles características de GitHub pueden ser útiles para su aplicación en la metodología.

Tabla 2. Lista de características de GitHub útiles en la metodología FDD

FUNCIONALIDAD	DESCRIPCIÓN	RELACIÓN CON FDD
Organizaciones y equipos	Módulo de gestión de organizaciones. Administración de equipos, accesos a repositorios y proyectos.	• *Develop a overall model*

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

Proyectos	Herramienta para la planificación y seguimiento de actividades de un proyecto.	• *Develop a overall model* • *Build a features list* • *Plan by feature*
Tablero de proyecto	Módulo que permite tener una visión general del estado de las actividades del proyecto.	• *Build a features list* • *Plan by feature*
Issues	Permite el seguimiento de ideas, tareas y resolución de errores.	• *Plan by feature* • *Build by feature*
Ramas	Permite a los desarrolladores crear copias de todo el código del repositorio para trabajar en un ambiente aislado.	• *Build by feature*
Pull request	Funcionalidad que permite a los desarrolladores solicitar la inclusión de cambios realizados a la rama principal del proyecto.	• *Build by feature*
Enlace a *Issues*	Permite que los cambios realizados estén enlazados a un *issue*. Esto se realiza utilizando las palabras clave y el código del *issue*.	• *Build by feature*

Diseño del flujo de trabajo propuesto

Utilizando algunas de las funcionalidades ofrecidas por la plataforma GitHub que se describen en la Tabla 2 es posible implementar algunos de los procesos de la metodología

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

FDD en un equipo de desarrollo. Estas funcionalidades permiten, entre otras cosas, definir y gestionar las listas de características que se deben implementar, asignar características para su implementación a los miembros de un equipo de desarrollo e integrar las características finalizadas al producto.

Mediante la aplicación de conceptos de *feature branches* y aprovechando las herramientas ofrecidas por la plataforma GitHub, en la Figura 2 se propone un flujo de trabajo para la aplicación de ciertos procesos de la metodología FDD a proyectos de desarrollo de software. Cabe mencionar que, la aplicación del esquema propuesto puede ser mejorado o adaptado a las necesidades de los equipos de desarrollo.

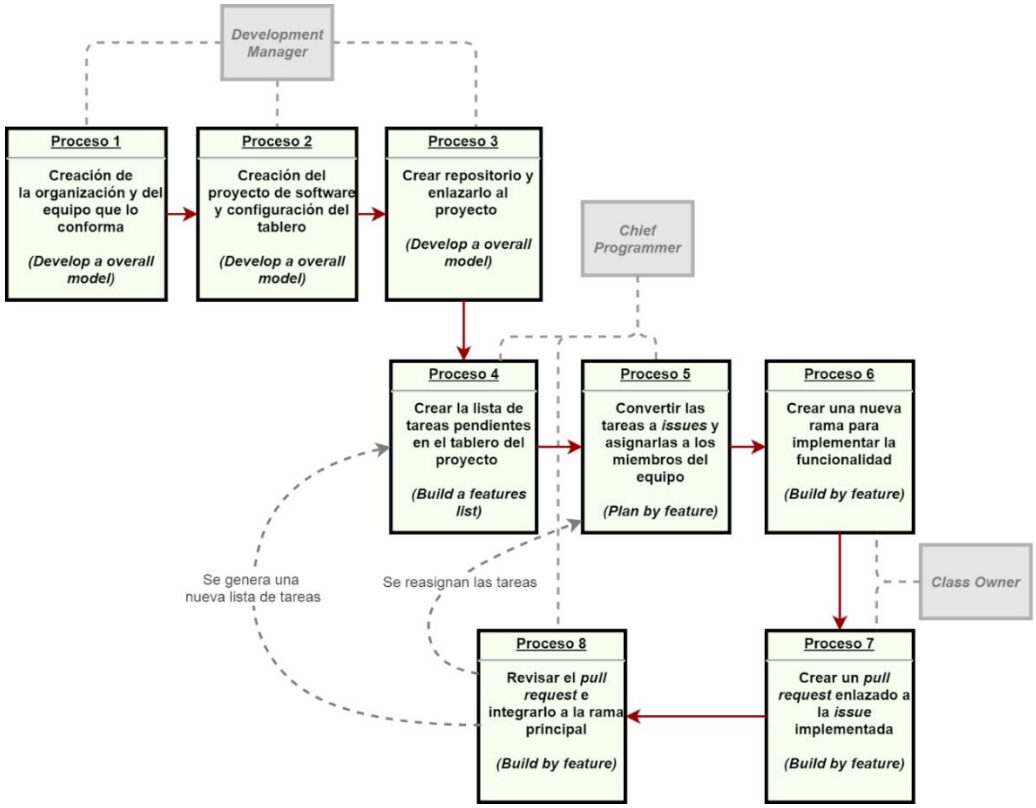


Figura 2. Flujo de trabajo basado en la metodología FDD y control de versiones con GitHub

El modelo de trabajo planteado parte de la definición de un equipo de desarrollo y continúa con la creación del proyecto y los repositorios asociados. Una vez establecidos estos aspectos, que pueden considerarse como aspectos más generales, comienzan los procesos orientados a la planificación, construcción e integración de características (también llamadas funcionalidades).

CIDUNAE
Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)
UNIVERSIDAD AUTÓNOMA DE ENCARNACIÓN

Dentro de un proyecto de desarrollo de software, los procesos desde el número 4 (cuatro) se ejecutan de manera iterativa como es habitual en las metodologías de desarrollo ágil. Una vez implementadas las características asignadas, presentadas en el flujo de trabajo propuesto como *issues*, y luego de que sean integradas a la rama principal del proyecto, es necesario realizar nuevas asignaciones a los desarrolladores o agregar nuevos elementos a la lista de características pendientes.

Caso de estudio

Como una forma de validar el flujo de trabajo propuesto se plantea un caso de estudio, el cual tiene como objetivo el desarrollo de una API REST de hola mundo con NodeJS.

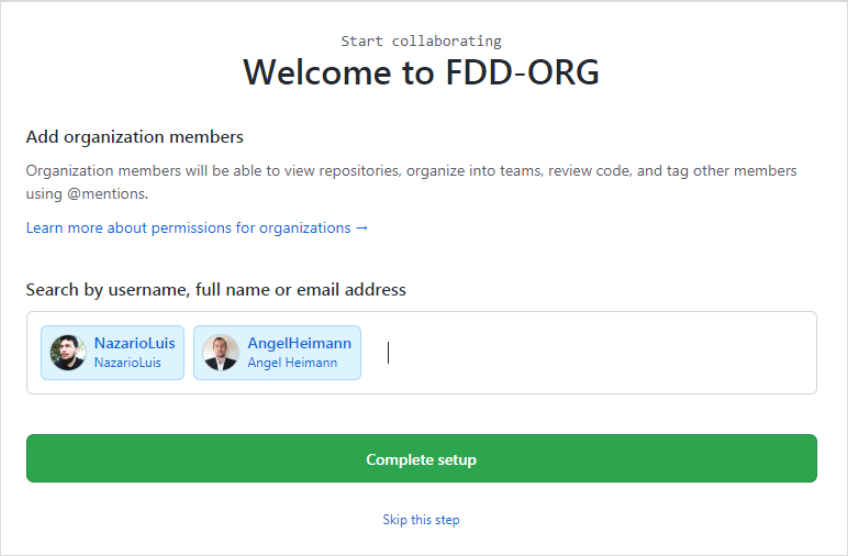


Figura 3. Inclusión de miembro en la organización

El caso de estudio inicia con la creación de una organización en Github a la que se denominó FDD-ORG. Además, se agregaron a los miembros de la organización como se visualiza en la Figura 3 y se creó un equipo de trabajo conformado por los miembros incluidos anteriormente, tal como se muestra en la Figura 4. Con estas tres acciones, se logra tener un espacio de trabajo integrado por todos los miembros de la organización y donde se pueden crear proyectos de desarrollo específicos para cada grupo de trabajo.

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

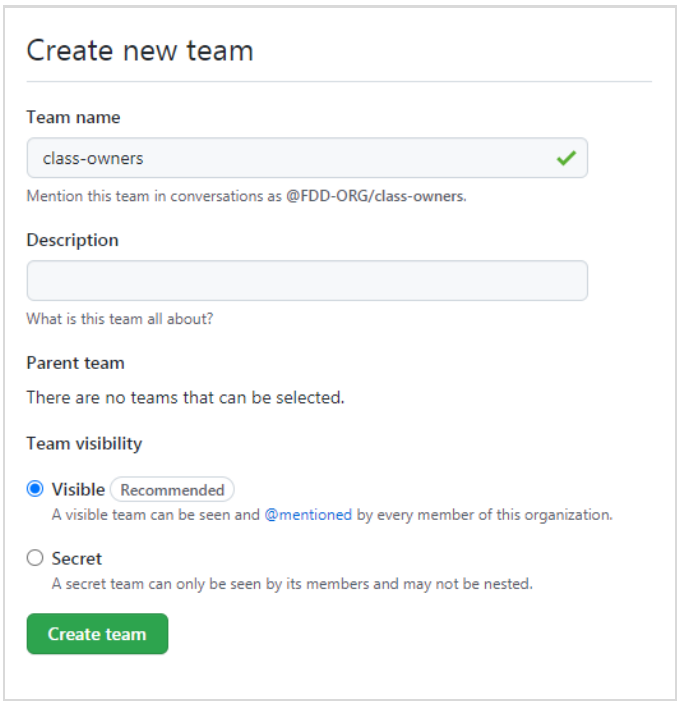


Figura 4. Creación del equipo de trabajo

Siguiendo el segundo proceso del flujo de trabajo, se procede a la creación de un nuevo proyecto como se visualiza en la Figura 5. El rol que se encarga de concretar esta acción es el *Development manager* quien también es el encargado de la planificación y el seguimiento de las actividades de desarrollo.

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

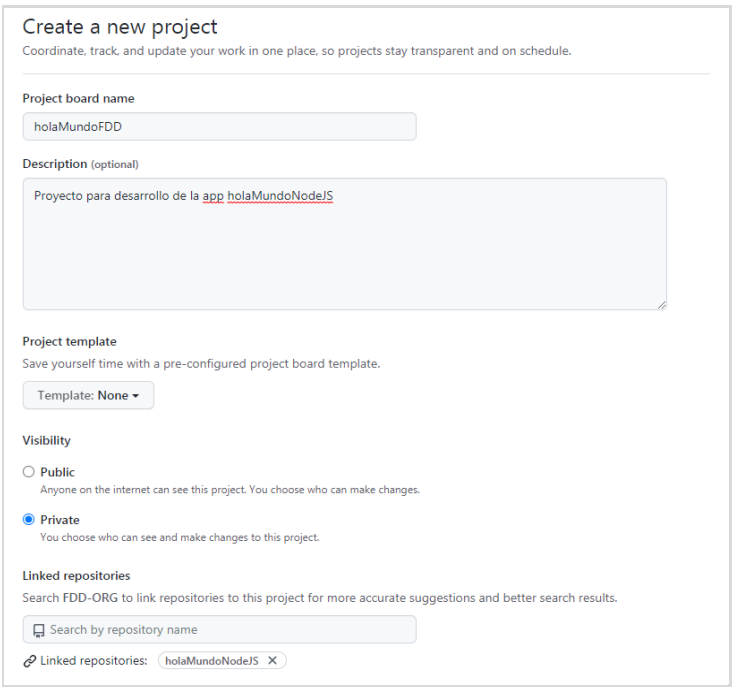


Figura 5. Creación de un nuevo proyecto

De la misma forma, el *Development manager* debe configurar el tablero kanban como se puede observar en la Figura 6. El tablero será utilizado por el equipo de desarrollo en cada iteración.

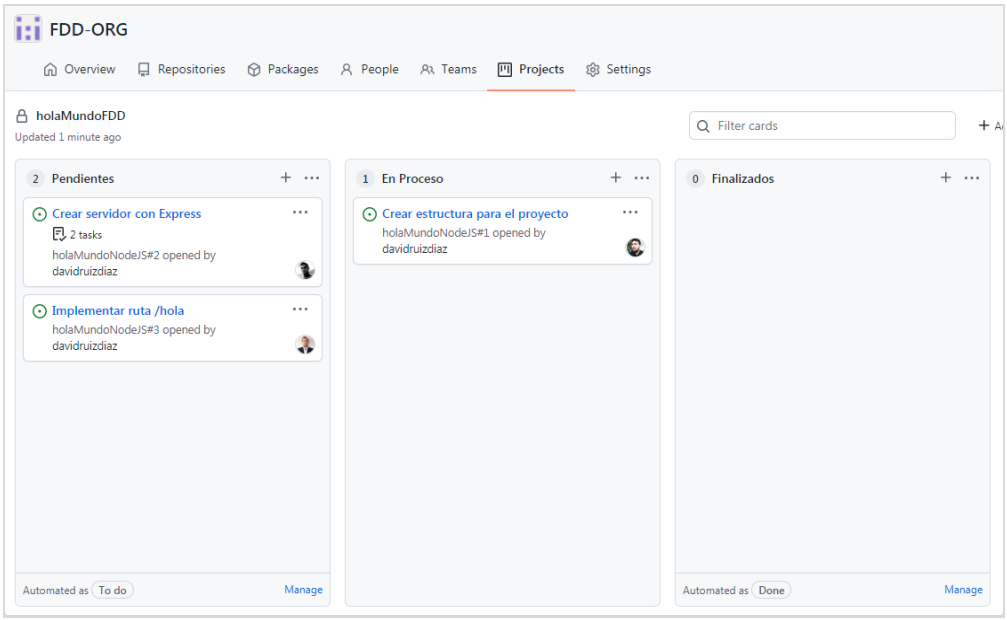


Figura 6. Configuración dl tablero kanban

CIDUNAE
Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)
UNIVERSIDAD AUTÓNOMA DE ENCARNACIÓN

El último proceso que debe realizar el encargado del proyecto es la creación del repositorio del proyecto. Todo el código generado por los *class owners* será almacenado en este repositorio. Por ende, el encargado de proyecto debe establecer los niveles de acceso pertinentes para cada miembro del equipo de desarrollo que necesite tener acceso al código fuente del software a desarrollar.

Una vez configurada la organización, los equipos de trabajo y el proyecto, el *chief programmer* crea, en el tablero kanban, una lista de todas las características o funcionalidades que deben ser implementadas e integradas al software principal. Esta lista de características es producto de un análisis realizado previamente y que incluyen funcionalidades generales y específicas que debe poseer el software.

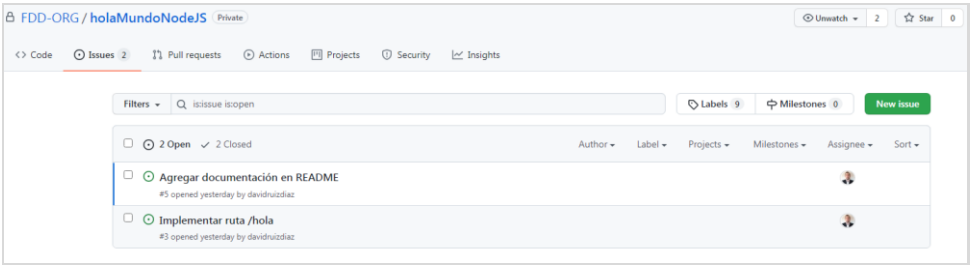
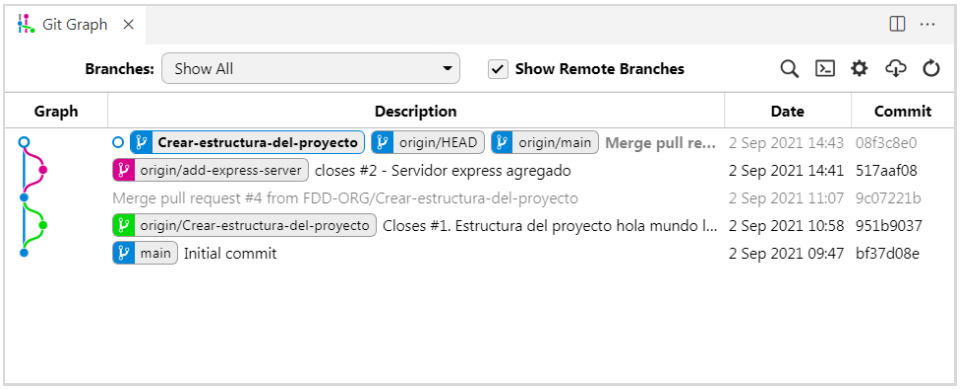


Figura 7. Issues generados a partir de las tareas del tablero Kanban

En el quinto proceso, el *chief programmer* debe organizar la lista de características obtenidas en la fase anterior según la prioridad de cada ítem en la lista. Una vez planificado el spring de desarrollo o la iteración de implementación, las características de mayor prioridad establecidas en la lista serán convertidas en *issues* que serán asignadas a los *class owners* correspondientes (ver Figura 7).



Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

Figura 8. Issues generados a partir de las tareas del tablero Kanban

De manera seguida, luego de que los *issues* fueron asignados a los desarrolladores, es necesario que se cree una rama separada para trabajar de manera aislada. Este proceso es el eje fundamental de la técnica de *feature branches*. En el Figura 8 se visualiza la creación de ramas para la implementación de manera aislada de las características, representadas por los *issues*, que fueron asignadas a los *class owners*. Una vez finalizado el proceso de implementación de las características, los desarrolladores solicitan la incorporación a la rama principal creando un *pull request* asociado a un *issue* específico (ver Figura 9).

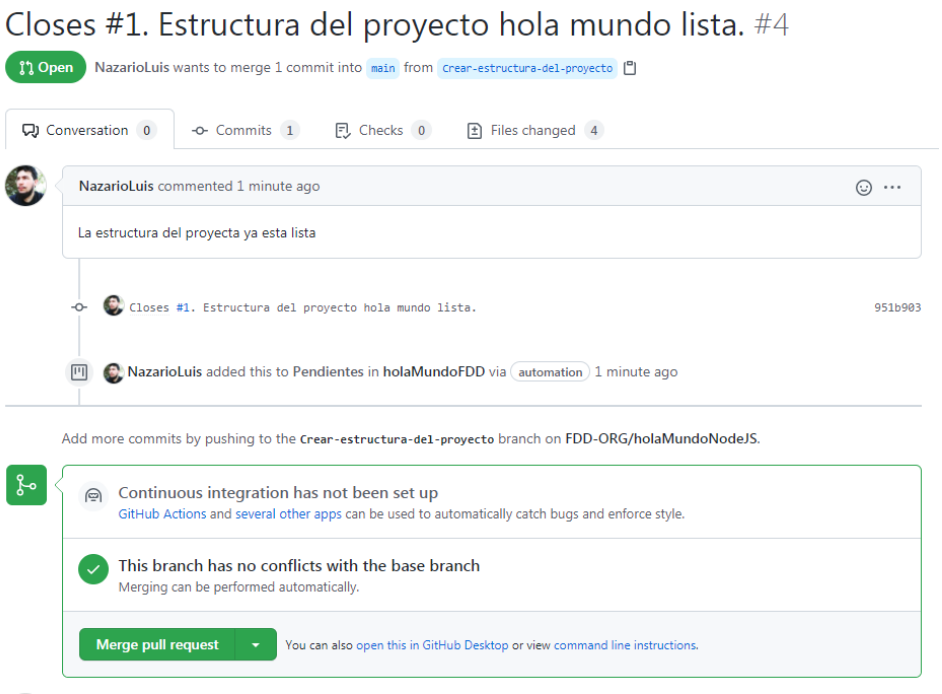


Figura 9. Solicitud de integración, Pull Request

El último paso es la revisión de las características por parte del *chief programmer* para la aceptación de las nuevas características y su integran a la rama principal del proyecto. Este proceso también puede visualizarse en la Figura 8 donde se aprecia como las ramas convergen en la rama principal mediante la ejecución de proceso *merge* sobre los *pull requests* generados. Cabe resaltar que, una vez cerrado un *issue* la tarea asociada pasa automáticamente a la columna de finalizados del tablero kanban (ver Figura 10).

Saeta Digital
Tecnología, Diseño e Innovación
ISSN: 2523-241X
Vol. 7 Núm. 1 (2021)

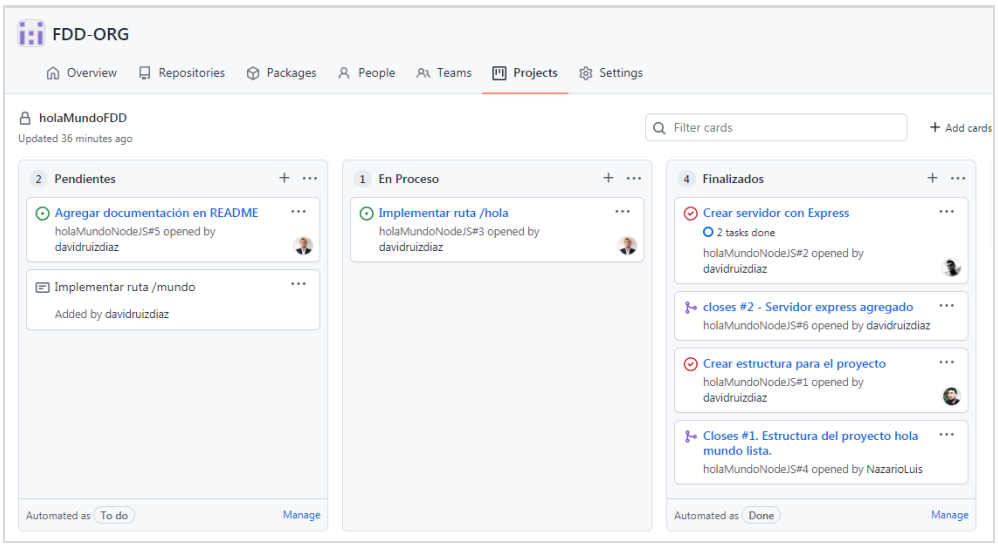


Figura 10. Visualización de tareas en diferentes estados en el tablero kanban

En el caso de estudio presentado se especifican funcionalidades o características con un bajo nivel de complejidad con fines meramente ilustrativos. Sin embargo, los procesos planteados en el documento deberían ser aplicables a otros escenarios con un nivel de complejidad mayor en sus requerimientos.

CONCLUSIONES FINALES

La técnica de *feature branches* fue sumamente útil en las fases de la FDD que fueron objeto de estudio (*Build feature list*, *Plan by feature* y *Build by feature*), debido a que el conjunto de herramientas seleccionadas sirve de soporte para todos los roles involucrados en esas fases de la metodología.

El flujo de trabajo propuesto, ilustrado en la Figura 2, describe de forma general el proceso que pueden seguir los equipos de desarrollo utilizando el modelo FDD y las herramientas de GitHub seleccionadas en este trabajo. También se deja abierta la posibilidad de adaptar las herramientas y el flujo de trabajo a la medida de cada equipo de desarrollo.

Finalmente, gracias a las pruebas realizadas en el caso de estudio, se puede mencionar que la aplicación de la metodología FDD, con la ayuda de la herramienta GitHub, proporciona a los pequeños equipos de desarrollo la posibilidad de gestionar los proyectos de manera simple siguiendo los procesos y roles establecidos en la metodología.

REFERENCIAS

[1] J. Hunt, Ed., "Feature-Driven Development," in *Agile Software Construction*, London: Springer London, 2006, pp. 161–182.

[2] S. Goyal, "Major seminar on feature driven development," *Jennifer Schiller Chair of Applied Software Engineering*, 2008, [Online]. Available: http://csis.pace.edu/~marchese/CS616/Agile/FDD/fdd.pdf

[3] F. Anwer, S. Aftab, U. Waheed, and S. S. Muhammad, "Agile software development models tdd, fdd, dsdm, and crystal methods: A survey," *International journal of multidisciplinary sciences and engineering*, vol. 8, no. 2, pp. 1–10, 2017.

[4] Z. Nawaz, Department of Computer Science, Virtual University of Pakistan, S. Aftab, and F. Anwer, "Simplified FDD Process Model," *International Journal of Modern Education and Computer Science*, vol. 9, no. 9. pp. 53–59, 2017. doi: 10.5815/ijmecs.2017.09.06.

[5] "Feature branch," *Optimizely*. https://www.optimizely.com/optimization-glossary/feature-branch/ (accessed Aug. 27, 2021).

[6] A. F. Chowdhury and M. N. Huda, "Comparison between Adaptive Software Development and Feature Driven Development," in *Proceedings of 2011 International Conference on Computer Science and Network Technology*, Dec. 2011, vol. 1, pp. 363–367.

[7] A. Firdaus, I. Ghani, and N. I. M. Yasin, "Developing secure websites using feature driven development (FDD): A case study," *J. Clean Energy Technol.*, pp. 322–326, 2013.

[8] M. Rychlý and P. Tichá, "A Tool for Supporting Feature-Driven Development," *Balancing Agility and Formalism in Software Engineering*. pp. 196–207, 2008. doi: 10.1007/978-3-540-85279-7_16.

[9] K. Ghane, "Quantitative planning and risk management of Agile Software Development," presented at the 2017 IEEE Technology & Engineering Management Conference (TEMSCON), Santa Clara, CA, USA, Jun. 2017. doi: 10.1109/temscon.2017.7998362.

[10] GitHub, Inc, "GitHub.com Documentación de Ayuda," *GitHub Docs*. https://docs.github.com/es (accessed Sep. 01, 2021).