Universidad de las Ciencias Informáticas

Facultad 4

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Aplicación Xedro-GESPRO para dispositivos móviles versión 1.0.

Autora: Lainet Padrón Rodríguez

Tutora: MSc. Iliana Pérez Pupo

La Habana, 2017

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Lainet Padrón Rodríguez	Msc. Iliana Pérez Pupo
Autora	Tutora

Pensamiento

La prueba más clara de la sabiduría, es la felicidad.

Montaigne

Agradecimientos

A Papi Rubén, por todo el amor que me entregó, por su comprensión, su apoyo incondicional y los valores que supo transmitirme. Gracias por haber sido mi abuelo, mi padre, mi amigo, mi maestro, mi consejero; por escucharme y confiar en mí siempre.

A mi abuela Hidalmis, por su apoyo brindado en todo momento, por su preocupación y desvelo constante para con mi bienestar y superación. Por su amor incondicional.

A mis padres Leisys y Alexis, por su apoyo y comprensión, por estar a mi lado en todo momento y ayudarme a superar etapas difíciles.

A mi hermana Lianet, por su cariño y comprensión en todo momento, todos mis esfuerzos y logros son gracias a ella por ser un ejemplo a seguir.

A Jansell por su apoyo, dedicación, preocupación constante, por guiarme y enseñarme a vivir, por su amor y comprensión y sus lecciones, por demostrarme que se puede encontrar todo lo que uno desea, en una persona. Gracias a él soy tan feliz.

A mi tío Alain, que ya no está a mi lado pero es un ejemplo para mí por esa fuerza con que se enfrentó a los obstáculos que la vida le puso.

A mis abuelos Pepe y Eva por darme todo su cariño.

Al resto de mi familia: Tata, Laura y Onelia.

A mi tutora Iliana que sin su ayuda no habría sido posible realizar mi sueño, ha estado dispuesto a ayudarme siempre y en los

momentos de mayor tensión estuvo a mi lado, ayudándome a seguir adelante.

A mis amigos Patry, Yasser, Randy, Damián, Pi, Sheila y Leyanet por estar presentes en los momentos más difíciles y felices de mi vida durante estos cinco años, por su apoyo constante.

A Johanet, Ñiquito, Magali.

Dedicatoria

A mi abuelo Rubén Luis Rodríguez Pedraza, la persona que más amo y amaré siempre, y a pesar de ya no estar a mi lado, sigue siendo mi motor impulsor y mi guía. A él le debo la persona que soy hoy.

A mi tía Leivys, que hoy ya no está conmigo, pero siempre conté con su cariño y sus consejos. Gracias por ser mi apoyo en momentos difíciles.

Resumen

Los dispositivos móviles en los últimos años han tenido un gran avance ya que con el paso del tiempo ha aumentado su nivel de utilidad y funcionalidad. Una función importante que tienen estos dispositivos es que nos permiten planificar nuestro trabajo ya sea desde un editor de texto, donde podamos introducir las tareas a realizar, como por una aplicación donde podamos gestionar nuestras tareas principales, es decir, no solo insertar, borrar o modificar tareas sino también revisar su estado. Actualmente en la Universidad de las Ciencias Informáticas (UCI) existe una aplicación para la gestión de proyectos Xedro-GESPRO; una de sus funciones principales es que les permite a los usuarios tener control de todas las tareas que tiene que realizar; sin embargo, esta aplicación no dispone de alguna opción que permita estar portable y accesible todo el tiempo independientemente del estado de la conectividad, pues el usuario depende de estar trabajando en una computadora conectada al servidor para conocer el estado de sus tareas y realizar otras funciones de gestión de proyectos. Por ello este departamento identificó como necesidad, contar con una aplicación Xedro-GESPRO que pueda ser usada desde los dispositivos móviles y así garantizar alta portabilidad y accesibilidad en funciones principales como lo es la visualización de las tareas de un proyecto y sus detalles, garantizando poder ver estas tareas cuando exista o no conectividad. Por esta razón se propone crear una aplicación que visualice el cronograma de peticiones (tareas) de Xedro-GESPRO, que tiene como objetivo facilitarles a los usuarios la consulta de sus tareas a través de un dispositivo móvil.

Palabras Claves:

Tecnología móvil, gestión de proyectos, GESPRO, peticiones, cronograma.

Abstract

Mobile devices in recent years have had a breakthrough as over time has increased its level of usefulness and functionality. An important function of these devices is that they allow us to plan our work either from a text editor, where we can introduce the tasks to be performed, or by an application where we can manage our main tasks, ie not only insert, delete or modify tasks but also check their status. Currently in the University of Computer Science (UCI) there is an application for the management of Xedro-GESPRO projects; One of its main functions is that it allows users to have control of all the tasks they have to perform; however, this application does not have some option that allows to be portable and accessible all the time regardless of the state of the connectivity, since the user depends on being working on a computer connected to the server to know the status of their tasks and perform other functions of project management. That is why this department identified as a necessity, to have a Xedro-GESPRO application that can be used from mobile devices and thus ensure high portability and accessibility in main functions such as the visualization of the tasks of a project and its details, ensuring that you can see these tasks when there is connectivity or not. For this reason, it is proposed to create an application to visualize the request schedule (tasks) of Xedro-GESPRO, which aims to make it easier for users to consult their tasks through a mobile device.

Keywords:

Mobile technology, project management, GESPRO, requests, timeline

Índice

Índice de Figuras

Índice de Tabla

Introducción

Nos encontramos en una época de cambios donde la sociedad trata cada vez más de independizarse de los medios computacionales y de que las Tecnologías de la Información y las Comunicaciones (TIC) estén cada vez más accesibles para los usuarios. Bajo este principio surge la tecnología móvil, que con el paso de los años ha evolucionado hasta llegar al surgimiento de los smartphone y los tablets.

La idea inicial del smartphone o teléfono inteligente era básicamente, unir las funciones de un PDA (Asistente Digital Personal) con las de un teléfono para mayor comodidad (Somos Comunicación, 2016) .El “Simon”, de IBM, fue el primer intento real de la industria tecnológica de crear un teléfono “con algo más” (Punto Geek, 2011). A medida que pasaron los años fueron evolucionado los smartphone hasta que en el 2016 surgieron el IPhone 7 y el Samsung Galaxy SVII, los cuales han seguido revolucionando en la historia y evolución de la tecnología móvil.

Otros de los dispositivos móviles que han hecho auge son las tablets ya que es un dispositivo nacido de la necesidad de unir dos mundos que durante muchos años estuvieron separados. Por un lado, tenemos a los PCs u ordenadores de toda la vida y por otro los dispositivos móviles usados para llevar tu trabajo encima como eran las PDA (Alsitecno.com, 2014). Una tablet es lo que entendemos por aparato electrónico móvil, no necesita estar siempre conectado a la red eléctrica, lo puedes llevar en las manos, y con el cual puedes interactuar gracias a su pantalla táctil. Sus usos son de lo más variado, desde GPS, reproductor de video o música hasta para navegar por Internet o consultar con el correo (Alsitecno.com, 2014).

El desarrollo de las tecnologías móviles nos permite el entretenimiento a niveles inimaginados, tomar fotos, videos, la comunicación a través de mensajes, llamadas, redes sociales, conectarnos en línea (*on-line*) entre programas o bases de datos y poder llevarnos a donde queramos nuestros programas favoritos; es decir, prácticamente nos permite una vida llevada a la virtualidad que le facilita la vida al usuario.

Pero no solo es beneficioso para distraerse, también facilita nuestro trabajo y permiten una óptima utilización del tiempo, cada vez más limitado en la era moderna y hace más accesible el conocimiento. Nos permiten, ya sea alertas o avisos de nuestras tareas programadas, de mapas, recetas, agendas o cualquier otro programa de interés para el usuario y cronogramas de nuestras tareas diarias. Una función importante es que nos posibilita planificar nuestro trabajo ya sea desde un editor de texto, donde podamos introducir las tareas a realizar, como por una aplicación donde podamos revisar el estado de nuestras tareas.

El uso de los dispositivos móviles es tan indispensable como la utilización de las computadoras en las empresas para la realización de sus actividades, es por ello que la Universidad de las Ciencias Informáticas (UCI) necesita potenciar el desarrollo de aplicaciones móviles. La UCI cuenta con varios centros productivos, entre los cuales se encuentra el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE). Como parte de su estructura se encuentra el departamento de Gestión de Proyectos. En este departamento fue creada la aplicación Xedro-GESPRO, que tiene como función principal la gestión de proyectos y que actualmente se utiliza en la universidad para gestionar todos los proyectos en cada uno de los centros productivos de la universidad.

Actualmente los dispositivos móviles nos permiten difundir y generalizar más el uso de aplicaciones y softwares, pues con esta tecnología se contrarresta la limitación que tienen las aplicaciones desktops y las aplicaciones web que se ofrecen desde servidores, y es precisamente la falta de portabilidad y accesibilidad; sin embargo:

Xedro-GESPRO es una herramienta desarrollada totalmente en software libre y su arquitectura está diseñada para ser desplegada en un servidor web. El cronograma de peticiones es el módulo más usado ya que con una simple consulta se puede acceder a las tareas planificadas en un proyecto para una persona determinada; además estas son la base para realizar el cálculo de varios indicadores. Ya que es una plataforma web, es necesario tener conexión con el

servidor y estar fijos en el lugar de acceso para lograr trabajar con ella. En este sentido, entre las principales dificultades detectadas se encuentran:

- Limitación con la accesibilidad, pues se depende de tener acceso a la red para acceder por web a los servicios de Gespro
- Limitación con la portabilidad porque no es posible disponer de los servicios de Xedro-GESPRO en otros entornos donde la conectividad es inaccesible.

A continuación se muestran algunos de los indicadores principales de Gespro que son calculados a través de las peticiones (García, 2012):

IE: Índice de Ejecución del Proyecto.

IRE: Índice de Rendimiento de la Ejecución.

IRP: Índice de Rendimiento de la Planificación.

IRC: Índice de Rendimiento de Costos.

ICD: Índice de Calidad del Dato.

IRL: Índice de Rendimiento de la Logística.

IRRH: Índice de Rendimiento de los Recursos Humanos.

IREF: Índice de Rendimiento de la Eficacia.

Debido a la situación descrita anteriormente, sale a relucir el siguiente **problema a resolver**:

La insuficiente portabilidad y accesibilidad en dispositivos móviles de la herramienta Xedro-GESPRO, está afectando el acceso a la información, fundamentalmente a las tareas planificadas para un proyecto.

Este problema se enmarca en el **objeto de estudio**: Aplicación de gestión para dispositivos móviles, dentro del que se ha identificado como **campo de acción**: Aplicación de Xedro-GESPRO para dispositivos móviles.

Se define como **objetivo** de este trabajo, visualizar tareas de Xedro-GESPRO en dispositivos móviles mediante el desarrollo de una aplicación web.

Dicho esto, tenemos como **posibles resultados**:

1- Actualización de datos desde un dispositivo móvil a la aplicación Xedro-GESPRO de un centro determinado.
2- Visualización del calendario de tareas en dispositivos móviles.

Para dar cumplimiento al objetivo, se ha organizado el trabajo a partir de las siguientes **tareas de investigación**:

1- Elaborar el marco teórico de la investigación.
2- Seleccionar las tecnologías para el desarrollo de la aplicación Xedro-GESPRO para dispositivos móviles.
3- Definir los requisitos funcionales y no funcionales que debe poseer el sistema para satisfacer las necesidades existentes.
4- Desarrollar la aplicación Xedro-GESPRO para dispositivos móviles que permita visualizar las tareas de Xedro-GESPRO web.
5- Validar y corregir la aplicación desarrollada.

Para alcanzar el objetivo propuesto se utilizan los siguientes métodos científicos:

Métodos teóricos:

Análisis histórico-lógico: Este método se empleó para la fundamentación de los aspectos teóricos contemplados en el desarrollo de la investigación acerca de la Gestión de Proyecto y desarrollo de aplicaciones para dispositivos móviles.

Métodos empíricos:

Encuesta: Aplicada para obtener información de los especialistas y usuarios de la aplicación, para realizar análisis del grado de aceptación, aplicabilidad y utilidad de la propuesta.

Estructura de la tesis:

El trabajo está estructurado en 3 capítulos, tal y como se describe a continuación:

Capítulo 1: Fundamento teórico: en este capítulo se presenta la definición del marco teórico de la investigación, se fundamenta el uso de los distintos temas referentes a la metodología, herramientas y tecnologías a utilizar para el desarrollo del sistema Xedro-GESPRO para dispositivos móviles.

Capítulo 2: Propuesta de solución: En este capítulo se presenta el modelo de dominio, requisitos funcionales y no funcionales, así como también se muestra los diagramas de clases, los diagramas de secuencia y el modelo de datos los cuales será de ayuda para la implementación de la aplicación.

Capítulo 3: Validación y pruebas: En este capítulo se presenta los componentes que tiene la aplicación mediante un diagrama, además se exponen los resultados de las pruebas realizadas a la solución.

Capítulo 1: Fundamento teórico

En este capítulo se analizará de forma teórica los principales aspectos que fundamentan la propuesta de solución, así como el objetivo que se persigue. Se hará un análisis de las tecnologías actuales para el análisis, diseño

y desarrollo de sistemas informáticos para dispositivos móviles.

1.1 Gestión de Proyectos

La gestión de proyectos es una disciplina de gestión que se está implantando de forma generalizada en el entorno empresarial y consiste en la aplicación de conocimientos, habilidades, técnicas y herramientas a las actividades necesarias para alcanzar los objetivos del proyecto (Pablo Lledo, 2007).

1.1.1 Áreas de conocimiento de la Gestión de Proyectos

Según el PMBOK_5ta Edición la gestión de proyectos tiene 10 áreas las cuales son:

- Gestión de la Integración
- Gestión del Alcance
- Gestión del Tiempo
- Gestión de los Costos
- Gestión de la Calidad
- Gestión de los Recursos Humanos
- Gestión de las Comunicaciones
- Gestión de los Riesgos
- Gestión de las Adquisiciones
- Gestión de los interesados (Proyect Manager Institute, 2013)

De las áreas de conocimiento expuestas anteriormente, el presente trabajo de investigación se enmarca en la gestión de alcance del proyecto, pues es donde se genera la propuesta de solución.

1.1.2 Gestión del Alcance

La Gestión del alcance del proyecto incluye los procesos necesarios para garantizar que el proyecto incluya todo el trabajo requerido y únicamente el trabajo para completar el proyecto con éxito. Gestionar el alcance del proyecto se enfoca primordialmente en definir y controlar qué se incluye y qué no se incluye en el proyecto (Proyect Manager Institute, 2013).

Existen un grupo de procesos dentro de la Gestión del alcance que generalmente se definen como parte del ciclo de vida del proyecto y se documentan en el plan de gestión del alcance del proyecto ver estos proceso consultar: (Proyect Manager Institute, 2013).

La suite Xedro-GESPRO, para gestionar los proyectos contiene todas estas áreas del conocimiento. En específico, la de Gestión de Alcance que se encuentra en el módulo Planificación, el cual contiene un grupo de funcionalidades como listar proyectos y listar peticiones o tareas, las cuales se pueden gestionar en el diagrama de Gantt, en el calendario y en el cronograma.

Todas estas funciones contenidas en el módulo Planificación son básicas para la gestión de un proyecto, por lo que es fundamental que la suite cuente con alguna herramienta que le permita garantizar la accesibilidad, portabilidad y disponibilidad de este módulo a sus usuarios, lo cual se lograría contando con una versión de la herramienta para dispositivos móviles.

1.2 Tipos de aplicaciones móviles

Existen tres tipos de aplicaciones móviles

1.2.1 Aplicación nativa

Las aplicaciones nativas son aquellas que se conciben para ejecutarse en una plataforma específica, es decir, se debe considerar el tipo de dispositivo, el sistema operativo a utilizar y su versión (Lisandro Delía, 2013).

El código fuente se compila para obtener código ejecutable, proceso similar que el utilizado para las tradicionales aplicaciones de escritorio (Lisandro Delía, 2013).
Cuando la aplicación está lista para ser distribuida debe ser transferida a las App stores (tiendas de aplicaciones) específicas de cada sistema operativo. Estas tienen un proceso de auditoría para evaluar si la aplicación se adecúa a los requerimientos de la plataforma a operar. Cumplido este paso, la aplicación se pone a disposición de los usuarios (Lisandro Delía, 2013).
La principal ventaja de este tipo de aplicaciones es la posibilidad de interactuar con todas las capacidades del dispositivo (cámara, GPS, acelerómetro, agenda, entre otras). Además, no es estrictamente necesario poseer acceso a internet. Su ejecución es rápida, puede ejecutarse en modo background y notificar al usuario cuando ocurra un evento que necesite su atención (Lisandro Delía, 2013).
Claramente estas ventajas se pagan con un mayor costo de desarrollo, pues se debe utilizar un lenguaje de programación diferente según la plataforma. Por ende, si se desea cubrir varias plataformas, se deberá generar una aplicación para cada una de ellas. Esto conlleva a mayores costos de actualización y distribución de nuevas versiones (Lisandro Delía, 2013).

1.2.2 Aplicación web o webapp

Las aplicaciones web para móviles son diseñadas para ser ejecutadas en el navegador del dispositivo móvil. Estas aplicaciones son desarrolladas utilizando HTML, CSS y JavaScript, es decir, la misma tecnología que la utilizada para crear sitios web (Lisandro Delía, 2013).
Una de las ventajas de este enfoque es que los dispositivos no necesitan la instalación de ningún componente en particular, ni la aprobación de algún fabricante para que las aplicaciones sean publicadas y utilizadas. Solo se requiere acceso a internet. Además, las actualizaciones de la aplicación son visualizadas directamente en el dispositivo, ya que los cambios son aplicados sobre el servidor y están disponibles de inmediato. En resumen, es rápido y fácil de poner en marcha (Lisandro Delía, 2013).

La principal ventaja de este tipo de aplicación es su independencia de la plataforma. No necesita adecuarse a ningún entorno operativo. Solo es necesario un navegador (Lisandro Delía, 2013).
Por contrapartida, esto disminuye la velocidad de ejecución y podrían llegar a ser menos atractivas que las aplicaciones nativas. Finalmente, este tipo de aplicaciones no pueden utilizar todos los elementos de hardware del dispositivo, como, por ejemplo, cámara, GPS, entre otros (Lisandro Delía, 2013).

1.2.3 Aplicación híbrida

Las aplicaciones híbridas combinan lo mejor de los dos tipos de aplicaciones anteriores. Se utilizan tecnologías multiplataforma como HTML, JavaScript y CSS, pero se puede acceder a buena parte de las capacidades específicas de los dispositivos (Lisandro Delía, 2013).
En resumen, son desarrolladas utilizando tecnología web y son ejecutadas dentro de un contenedor web sobre el dispositivo móvil (Lisandro Delía, 2013).
Entre las principales ventajas de esta metodología se pueden mencionar la posibilidad de distribución de la aplicación a través de las tiendas de aplicaciones, la reutilización de código para múltiples plataformas y la posibilidad de utilizar las características de hardware del dispositivo (Lisandro Delía, 2013).
Una de las desventajas es que, al utilizar la misma interfaz para todas las plataformas, la apariencia de la aplicación no será como la de una aplicación nativa (Lisandro Delía, 2013).
Finalmente, la ejecución será más lenta que la ejecución en una aplicación nativa (Lisandro Delía, 2013).

A continuación, en la ***Tabla 1*** se muestra una comparación entre los diferentes tipos de aplicaciones móviles teniendo en cuenta cuatro aspectos fundamentales que permiten identificar cual es la más adecuada para desarrollar la solución propuesta.

Tabla 1. Comparación de los tipos de aplicaciones móviles

Tipo de aplicación	Reutilización de código	Publicación de servicios en la web	Multiplataforma	Portabilidad
Nativa	El código no es reutilizable para cada una de las plataformas.	No se puede publicar servicios en la web ya que la aplicación no se ejecuta a través de un navegador	Solo se puede ejecutar en la plataforma para la cual fue programada.	La aplicación nativa es portable.
Webapp	El código es reutilizable para cada una de las plataformas.	Se pueden publicar servicios en la web ya que la aplicación se ejecuta sobre el navegador.	Es multiplataforma, ya que no importa la plataforma donde se está ejecutando la aplicación ya que se ejecuta a través del navegador.	La aplicación Webapp no es portable.
Híbrida	El código es reutilizable para cada una de las plataformas.	Si se pueden publicar servicios en la web porque este tipo de aplicación se instala como una aplicación nativa pero se programa con lenguajes propios de las aplicaciones web, además de que pueden ser ejecutadas en la web.	Es multiplataforma porque con un único código se puedo compilar la aplicación para todas las plataformas.	La aplicación hibrida es portable.

Luego de haber analizado la ***Tabla 1*** se ha decidido que la aplicación a desarrollar sea híbrida ya que funcionará en todos los sistemas operativos; sin embargo, la UCI no cuenta con las licencias para los sistemas operativos que son privados, por lo que solo se compilará la aplicación para Android, quedando el código disponible para cuando la universidad cuente con estas licencias pueda generar la aplicación para dichos sistemas operativos.

1.3 Tecnologías utilizadas en el desarrollo de la aplicación.

1.3.1 Metodología de desarrollo

Una metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información (SELECTING A DEVELOPMENT APPROACH , 2008). Existen dos tipos de metodologías, las Ágiles y las Pesadas o Tradicionales; de estas solo se enfoca en las Ágiles ya que para el proceso de desarrollo se utilizará una metodología Ágil.

A continuación, en la ***Figura 1*** se muestran las metodologías ágiles más usadas hasta el 2015.

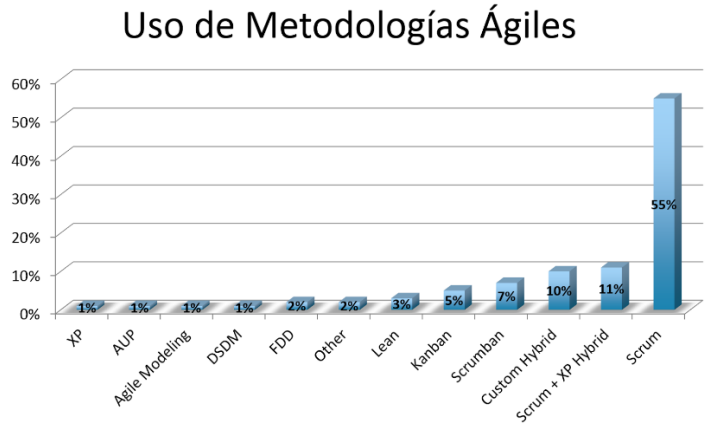


Figura 1. Metodologías ágiles más usadas hasta el 2015 (Rivas, 2015)

Esta figura nos indica que Scrum es la metodología ágil más usada dentro de los equipos de proyectos que trabajan con metodologías ágiles. La metodología seleccionada para el desarrollo de la solución propuesta se denomina AUP-UCI y se basa en una variación a la metodología ágil AUP (Proceso Unificado Ágil). Esta metodología garantiza la calidad y se ajusta al proceso productivo de la universidad.

AUP-UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, entre otros) exigiéndose así que el proceso sea configurable. Se decide

hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el Modelo CMMI (Capability Maturity Model Integration)-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (Sánchez, 2015).

Fases

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. (Sánchez, 2015). A continuación, en la ***Figura 2*** se muestra como quedan las fases en la metodología AUP-UCI.

Figura 2. Fases AUP-UCI

Disciplinas

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas:

Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (Sánchez, 2015). A continuación, en la ***Figura 3*** se muestra como quedan las disciplinas en la metodología AUP-UCI.

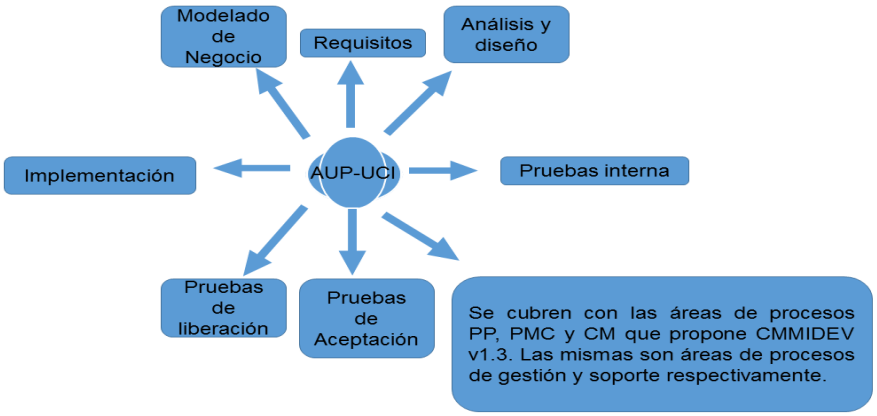


Figura 3. Disciplinas AUP-UCI

Escenarios para la disciplina Requisitos.

Escenario No 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización (Sánchez, 2015).

Escenario No 2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información (Sánchez, 2015).

Escenario No 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados (Sánchez, 2015).

Escenario No 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. Todas las disciplinas antes definidas (desde Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen Iteraciones y se obtengan resultados incrementales. En una iteración se repite el flujo de trabajo de las disciplinas, Requisitos, Análisis y diseño, Implementación y Pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr esto, cada requisito debe tener un completo desarrollo en una única iteración (Sánchez, 2015).

Para guiar el proceso de desarrollo de la propuesta de solución se utilizó la metodología AUP-UCI puesto a que es una metodología adaptada al proceso de desarrollo de software en la UCI y dentro de los escenarios que posee se seleccionó el escenario cuatro debido al que negocio está bien definido y el cliente siempre acompaña al equipo de desarrollo.

1.3.2 Herramientas utilizadas

Para la realización de la aplicación se necesitan las siguientes herramientas:

Visual Paradigm for UML 8.0

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos,

construcción, pruebas y despliegue. Permite modelar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación (Yoandri Quintana Rondón, 2011).
La herramienta agiliza la construcción de aplicaciones con calidad y a un menor coste. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como obtener ingeniería inversa de bases de datos (Yoandri Quintana Rondón, 2011).

Para el desarrollo del sistema propuesto es conveniente utilizar esta herramienta, pues la misma permite diseñar y modelar un conjunto de diferentes diagramas que se convertirán en el punto de partida para las posteriores tareas de investigación, también aporta una documentación general del funcionamiento del negocio.

SQLite

SQLite es sin duda uno de los motores de base de datos más interesante, dado que sus características hacen que se trate de un motor universal, puede encontrarse en prácticamente cualquier dispositivo y por tanto es óptimo para que los archivos generados puedan utilizarse en cualquiera de las múltiples plataformas, dispositivos y sistemas embebidos disponibles. SQLite es el motor de bases de datos más utilizado del mundo (Menéndez, 2017).

Las bases de datos generadas por SQLite son como un archivo más en tu dispositivo de almacenamiento, y que dicho archivo contiene tanto los datos almacenados como la estructura de base de datos relacionados definida en su creación. Todo en un mismo archivo, que simplemente podrías copiar hacia otro ordenador, sistema o dispositivo, por ejemplo, para seguir utilizándolo sin más, con independencia de que el sistema operativo utilizado sea de 32 o 64 bits (Menéndez, 2017).

SQLite ofrece un rendimiento más que notable, permitiendo realizar operaciones tanto en disco como en memoria para proporcionar respuestas a consultas complejas en cuestión de pocos milisegundos. A lo anterior se suma el hecho de que al tratarse de un motor de bases de datos local no es preciso contar con la

compleja infraestructura cliente-servidor propia de otros motores de bases de datos: los archivos de bases de datos generados por SQLite son locales, lo que significa que por lo general se almacenan sobre el mismo dispositivo sobre el que se está ejecutando la aplicación que hace uso de la base de datos propiamente dicha. Es decir, no requiere infraestructura adicional, no requiere de una configuración previa, y no requiere de mantenimiento. Simplemente, funciona. El motor de bases de datos SQLite está empotrado como una librería más en tu aplicación, formando parte del ejecutable final y aumentando mínimamente el tamaño del archivo resultante (Menéndez, 2017).

GNU EMACS

GNU Emacs es mucho más que un procesador de textos. Los programadores están impresionados por sus funciones integradas de depuración y gestión de proyectos. Emacs también es un procesador de textos multilingüe, puede manejar todas sus necesidades de correo electrónico, mostrar páginas web e incluso tiene un diario y un calendario para sus citas (Stallman, 2007). Algunas características de Emacs son:

- Modos de edición especiales para 25 lenguajes de programación incluyendo Java, Perl, C, C ++, Objetivo C, Fortran, Lisp, Scheme y Pascal.
- Soporte para escribir y mostrar en 21 idiomas no ingleses, incluyendo chino, checo, ruso, vietnamita y todos los idiomas de Europa Occidental.
- Compilar y depurar desde dentro de Emacs (Stallman, 2007).

Jasmine

Jasmine es un marco de desarrollo basado en el comportamiento para probar código JavaScript. No depende de ningún otro framework de JavaScript. No requiere un DOM (Modelo de Objeto de Documento). Y tiene una sintaxis clara y obvia para que pueda escribir fácilmente pruebas (Jasmine, 2017).

Karma

Karma es esencialmente una herramienta que genera un servidor web que ejecuta el código fuente contra el código de prueba para cada uno de los navegadores conectados. Los resultados de cada prueba en cada navegador se examinan y se muestran a través de la línea de comandos al desarrollador de tal manera que pueden ver qué exploradores y pruebas pasaron o fallaron (Karma, 2017).

1.3.3 Lenguajes para el modelado y desarrollo

UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios.

Este lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código, así como generadores de informes. La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (James Rumbaugh, 2000)

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre los objetos. El

comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos. (James Rumbaugh, 2000)

UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. Contiene construcciones para representar decisiones de implementación y para elementos de tiempo de ejecución en componentes. (James Rumbaugh, 2000)

Se decide emplear este lenguaje de modelado en el proceso de desarrollo de la propuesta de solución a causa de las características antes mencionadas y por su amplio uso en la UCI.

HTML5

El lenguaje HTML (hypertext markup language) se utiliza para crear documentos que muestren una estructura de hipertexto. Un documento de hipertexto es aquel que contiene información cruzada con otros documentos, lo cual nos permite pasar de un documento al referenciado desde la misma aplicación con la que lo estamos visualizando. HTML permite, además, crear documentos de tipo multimedia, es decir, que contengan información más allá de la simplemente textual, como por ejemplo videos, imágenes y sonido (Mateu, 2004).

SASS

Sass (*Syntactically Awesome StyleSheets*) es una extensión de CSS que añade características muy potentes y elegantes a este lenguaje de estilos. Sass permite el uso de variables, reglas CSS anidadas*,* mixins*,* importación de hojas de estilos y muchas otras características, al tiempo que mantiene la compatibilidad con CSS. (Hampton Catlin, 2015)

Sass permite organizar mejor las hojas de estilos grandes y permite ser mucho más productivo con las hojas de estilos pequeñas (Hampton Catlin, 2015). Sass incluye las siguientes características:

- 100% compatible con CSS3.
- Permite el uso de variables, anidamiento de estilos y *mixins*.
- Incluye numerosas funciones para manipular con facilidad colores y otros valores.
- Permite el uso de elementos básicos de programación como las directivas de control y las librerías.
- Genera archivos CSS bien formateados y permite configurar su formato (Hampton Catlin, 2015).

TypeScript

TypeScript es un lenguaje de programación moderno que permite crear aplicaciones web robustas en JavaScript. TypeScript no requiere de ningún tipo de plugin, puesto que lo que hace es generar código JavaScript que se ejecuta en cualquier navegador, plataforma o sistema operativo.
TypeScript es un "transpilador", es decir, un compilador que se encarga de traducir las instrucciones de un lenguaje a otro, también se llamará pre-compilador ya que este realmente intenta realizar las funciones de un compilador más las funciones de un traductor de instrucciones. La búsqueda de errores de runtime en JavaScript puede ser una tarea imposible por lo que TypeScript proporciona detección temprana de errores (en tiempo de compilación (Emmanuel Valverde Ramos, 2016).

1.3.4 Frameworks

Los frameworks que van ser utilizados son:

Ionic

Ionic Framework es un SDK de código abierto que permite a los desarrolladores crear aplicaciones móviles de alta calidad utilizando tecnologías web familiares (HTML, CSS y JavaScript). Ionic se centra principalmente en el aspecto y la sensación, o la interacción de interfaz de usuario, de una aplicación. Esto significa que no es un reemplazo para PhoneGap o su marco de JavaScript favorito (Docs, 2017).

Ionic actualmente requiere Angular con el fin de trabajar en todo su potencial. Si bien todavía puede utilizar la parte CSS del marco, se perderá poderosas interacciones de interfaz de usuario, gestos, animaciones y otras cosas. En el futuro, Ionic planea volverse más agnóstico para soportar una variedad más amplia de frameworks de JavaScript.

Apache Cordova

Apache Cordova es un framework de código abierto que permite el desarrollo de aplicaciones móviles usando tecnologías web HTML, CSS y JavaScript. Contiene un conjunto de *API* (*Interfaz de Programación de Aplicaciones*) JavaScript para acceder a las características de los dispositivos. Tiene soporte para muchos sistemas operativos (iOS, Android, Blackberry, Windows Phone, entre otros) (Rodríguez, 2013).

Consideraciones Parciales

- Las aplicaciones híbridas son ejecutadas en todos los tipos de dispositivos móviles sin la necesidad de crear una aplicación para cada dispositivo en específico, son independientes del sistema operativo instalado en el dispositivo móvil y su código es reutilizable.
- AUP-UCI es una variación de la metodología ágil AUP que permite garantizar la calidad de la solución propuesta y a través de sus fases y disciplinas guía el proceso de desarrollo; además es una metodología adaptada al proceso de desarrollo de software de la UCI.

- Visual Paradigm es una herramienta que permite el diseño de aplicaciones mediante diagramas de casos de usos, prototipos de interfaz, entre otros. Esta herramienta está disponible en múltiples plataformas y soporta aplicaciones Web.
- SQLite permite almacenar información de manera sencilla y ligera, se puede usar tanto en dispositivos móviles como en sistemas de escritorio. Este permite desde las consultas más sencillas hasta las más complejas.
- HTML5 permite el desarrollo de aplicaciones que pueden ser usadas en múltiples dispositivos. Este lenguaje nos permite acceder a sitios web de manera offline, sin estar conectados a internet.
- Ionic posee un alto rendimiento, nos permite desarrollar aplicaciones más robustas y nos permite poder trabajar con todos los dispositivos móviles actuales. Este framework utiliza para su desarrollo Angular JS que es un framework de JavaScript que trabaja del lado del cliente y nos permite hacer más dinámica la aplicación Web.
- Apache Cordova nos permite crear aplicaciones para distintas plataformas móviles utilizando como lenguajes de programación HTML5, CSS y JavaScript.

Capítulo 2: Propuesta de Solución

En este capítulo se presenta la modelación del sistema, donde se realiza el análisis previo para comenzar la confección de la aplicación y transformar requisitos y necesidades de los usuarios, de modo que satisfagan sus expectativas. Se puntualiza el modelo de dominio para representar la estructura estática de la solución, así como la relación entre las diferentes clases, apoyadas en el diagrama de clases. Se muestra además el diseño de la base de datos, que no es más que la representación visual de cada tabla de la base de datos con las relaciones entre ellas. Por último, se listan los requisitos funcionales y no funcionales a los cuales debe responder la solución, se presentan las interfaces de usuario y los patrones utilizados en el proceso de desarrollo.

2.1 Modelo de dominio

Un modelo de domino es una representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que describen clases de software, u objetos de software con responsabilidad (Larman, 2003).

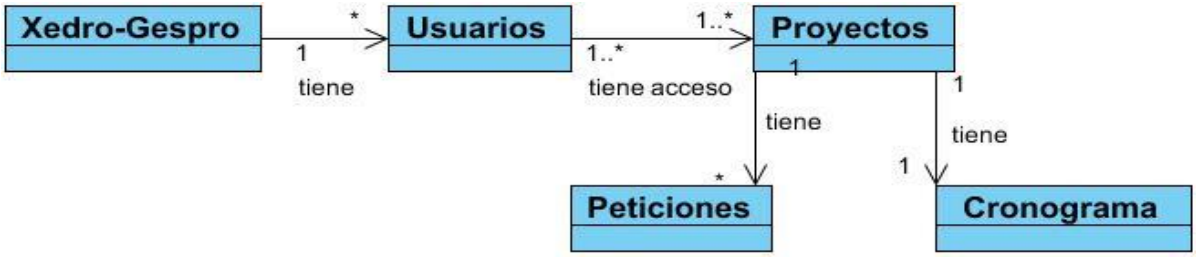


Figura 4. Modelo de Dominio.

En la ***Figura 4*** se muestran las clases que intervienen en el negocio; la clase Xedro-GESPRO es la herramienta que permite llevar el control de todos los proyectos de la UCI, la clase Usuarios es donde están registrados todos los usuarios que utilizan esta herramienta, la clase Proyectos es donde se encuentran todos los proyectos con los que cuenta la universidad, en la clase peticiones se encuentran todas las peticiones que están asociadas a cada uno de los proyectos y en la clase cronograma es donde se encuentra el cronograma de cada uno de los proyectos. Este diagrama no solo muestra las clases que intervienen en el

negocio, sino también la relación que existe entre ella; Xedro-GESPRO tiene muchos usuarios los cuales están vinculados a uno o más proyectos, estos proyectos tienen muchas peticiones y un único cronograma.

2.2 Requisitos

2.2.1 Requisitos Funcionales

Los requisitos funcionales definen el comportamiento interno de un software, son condiciones que el sistema ha de cumplir. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software. Para ello se definen los siguientes requisitos funcionales:

RF1. Autenticar Usuario: Esta funcionalidad les permite a los usuarios una vez instalada la aplicación, y luego de haber accedido por primera vez, autenticarse para poder guardar sus datos; permite además que al estar conectados a una red Wifi puedan autenticarse para poder actualizar sus datos.

RF2. Visualizar Proyectos: Esta funcionalidad les permite a los usuarios según los permisos que tenga, ver todos los proyectos; además permite seleccionar el proyecto que desee y ver sus peticiones y el cronograma.

RF3. Visualizar Peticiones: Esta funcionalidad les permite a los usuarios ver las peticiones luego de seleccionar un proyecto.

RF4. Visualizar Detalles Peticiones: Esta funcionalidad les permite a los usuarios visualizar los detalles que tiene cada petición.

RF5. Visualizar Cronograma: Esta funcionalidad les permite a los usuarios, luego de haber seleccionado un proyector ver el cronograma.

2.2.2 Requisitos No Funcionales

Los requisitos no funcionales son propiedades y características que hacen la aplicación más atractiva, usable y confiable. Se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar. Para la solución propuesta se definen los siguientes requisitos no funcionales:

RnF1. Requisito de Usabilidad

RnF1.1. El nombre de los botones debe de tener relación con la función que realizan.

RnF1.2. La herramienta tiene que ser portable.

RnF1.3. La herramienta tiene que tener una fácil accesibilidad.

RnF2. Requisito de Interfaz y Apariencia externa.

RnF2.1. Debe de ser semejante al sitio de Xedro-GESPRO y poseer el logo de Xedro-GESPRO.

RnF3. Requisito de Seguridad

RnF3.1. El sistema debe permitir a los usuarios autenticarse.

RnF4. Requisito de software

Requisitos mínimos para el dispositivo móvil:

RnF4.1. Soporte para conexiones WIFI.

RnF4.2. Sistema operativo Android v4.4.2 o superior.

2.2.3 Historia de Usuarios

Una historia de usuario es una representación de un requerimiento de software escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requerimientos (Ecured, 2017).

Las historias de usuario son una forma rápida de administrar los requerimientos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Estas permiten responder rápidamente a los requerimientos cambiantes (Ecured, 2017).

Tabla 2. HU_AutenticarUsuario

Número: 1	**Nombre del requisito:** AutenticarUsuario
Programador: Lainet Padrón Rodríguez	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 5 días

Riesgo en Desarrollo: N/A	**Tiempo Real:** 3 días
Descripción: **1- Objetivo:** Permitir al usuario autenticarse en la aplicación. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para autenticarse: - El usuario debe de estar registrado en el sistema. **3- Flujo de la acción a realizar:** Cuando el usuario se autentica podrá ver la lista de los proyectos a los cuales está vinculado.	
Observaciones: N/A	
Prototipo de interfaz: 	

Tabla 3. HU_VisualizarProyectos

Número: 2	**Nombre del requisito:** VisualizarProyectos
Programador: Lainet Padrón Rodríguez	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 15 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 10 días
Descripción: **1- Objetivo:** Permitir visualizar los proyectos a los cuales esté vinculado el usuario autenticado. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para visualizar proyectos hay que: - Ser miembro de algún proyecto. - Debe existir en el sistema al menos un proyecto.	

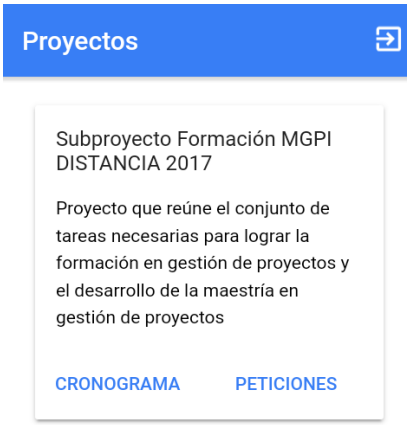
3- Flujo de la acción a realizar: Cuando el usuario se autentica aparecen todos los proyectos a los cuales está vinculado.
Observaciones: los proyectos contienen todas las peticiones asignadas a cada uno de sus miembros.
Prototipo de interfaz: 

Tabla 4. HU_VisualizarPeticiones

Número: 3	**Nombre del requisito:** VisualizarPeticiones
Programador: Lainet Padrón Rodríguez	**Iteración Asignada:** 3era
Prioridad: Alta	**Tiempo Estimado:** 15 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 10 días
Descripción: **1- Objetivo:** Permitir listar las peticiones de cada proyecto en la aplicación. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para visualizar las peticiones hay que: - Estar autenticado en el sistema y ser miembro de al menos un proyecto. - Debe existir en el sistema al menos un proyecto. - El proyecto debe contener al menos una petición. **3- Flujo de la acción a realizar:** Cuando el usuario selecciona la opción VisualizarPeticiones, se mostrarán las peticiones que tenga el proyecto al cual está vinculado. Además, el usuario tiene la posibilidad de ver detalles de cada uno de ellas.	
Observaciones: las peticiones describen y controlan la ejecución de las actividades del	

proyecto.

Prototipo de interfaz:

Tabla 5. HU_VisualizarDetallesPeticiones

Número: 4	**Nombre del requisito:** VisualizarDetallesPeticiones
Programador: Lainet Padrón Rodríguez	**Iteración Asignada:** 4ta
Prioridad: Alta	**Tiempo Estimado:** 15 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 10 días

Descripción:

1- Objetivo:

Permitir visualizar los detalles de las peticiones de cada proyecto en la aplicación.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para visualizar los detalles de las peticiones hay que:

- Estar autenticado en el sistema y ser miembro de al menos un proyecto.
- Debe existir en el sistema al menos un proyecto.
- El proyecto debe contener al menos una petición.
- La petición debe tener información que mostrar.

3- Flujo de la acción a realizar:

Cuando el usuario selecciona la opción Ver, verá en detalle cada uno de los elementos que componen una petición.

Observaciones: las peticiones describen y controlan la ejecución de las actividades del proyecto y en esta vista se puede observar cada uno de los componentes de cada petición.

Prototipo de interfaz:

Tabla 6. HU_VisualizarCronograma

Número: 5	**Nombre del requisito:** VisualizarCronograma
Programador: Lainet Padrón Rodríguez	**Iteración Asignada:** 5ta
Prioridad: Alta	**Tiempo Estimado:** 15 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 10 días
Descripción: **1- Objetivo:** Permitir visualizar el cronograma de tareas de los proyectos. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para visualizar el cronograma hay que: - Estar autenticado en el sistema y ser miembro de al menos un proyecto. - Debe existir en el sistema al menos un proyecto. - El proyecto debe contener al menos una petición. **3- Flujo de la acción a realizar:** Cuando el usuario selecciona la opción de Visualizar el cronograma, verá el cronograma del proyecto.	
Observaciones: el cronograma de tareas muestra más detallado la dependencia que existen entre las tareas del proyecto.	
Prototipo de interfaz:	

2.3 Diseño

La esencia del diseño del software es la toma de decisiones sobre la organización lógica del software. Algunas veces, se representa esta organización lógica como un modelo en un lenguaje definido de modelado tal como UML y otras veces simplemente se utiliza notaciones informales y esbozos para representar el diseño (Sommervile, 2006).

El proceso de diseño tiene asociado la decisión del tipo arquitectura y los patrones de diseño que empleará el sistema, así como la confección de distintos diagramas que favorezcan el trabajo en la fase de implementación

2.3.1 Arquitectura de la aplicación

La arquitectura Modelo-Vista-Controlador (MVC) surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. La arquitectura MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de frameworks basados en la arquitectura MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores (Yenisleidy Fernández Romero, 2012).

Modelo: Se encuentra la base de datos de la aplicación.

Vista: Se encuentra todas las interfaces de la aplicación.

Controlador: Se encuentra el código para obtener los datos de la base de datos.

Figura 5. Arquitectura de software

2.3.2 Patrones de diseño

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos (Larman, 2003).

Los patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones (Larman, 2003).

Los patrones GRASP definidos son:

Bajo acoplamiento: posibilita que una clase no dependa mucho de otras clases. Este patrón se empleará en las clases "*proyectos*", "*peticiones*", "*detallesPeticiones*", "*login*" y "*cronograma*", pues al crear la clase "*databaseService*" se disminuye la dependencia al no tener que conocer ni recurrir demasiado a la clase que brinda Ionic.

Alta Cohesión: este patrón caracteriza a las clases que posean responsabilidades estrechamente relacionadas, es decir, que no realicen un trabajo enorme. Con el objetivo de que las clases "*proyectos*", "*peticiones*", "*detallesPeticiones*", "*login*" y "*cronograma*" no realizaran un trabajo enorme y poder reutilizar código se creará la clase "*databaseService*", encargada de las funcionalidades de conexión a la base de datos.

Experto: mediante su uso, se asignan responsabilidades a la clase que cuenta con la información necesaria. Se evidenciará en la clase "*databaseService*" pues esta poseerá un objeto de la clase "*SQLite*" para poder acceder a la información y funcionalidades de la base de datos.

Lo patrones GOF (Pandilla de cuatro) definidos son:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Este patrón se evidencia en la clase "*databaseService*".

2.3.3 Diagrama de clases del diseño

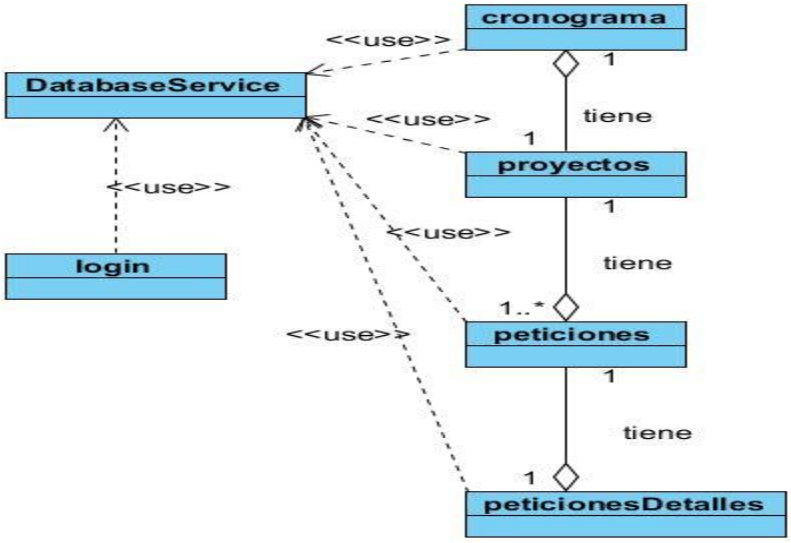


Figura 6. Diagrama de clases del diseño

En la ***Figura 6*** se muestra la relación que existe entre todas las clases que están presentes en la solución. Las clases login, proyectos, peticiones, peticionesDetalles y cronograma dependen de la clase DatabaseService. Además,

se representa las relaciones que tienen las clases anteriormente mencionadas; la clase proyecto tiene un único cronograma y de una a muchas peticiones se relaciona con una única peticionDetalles.

2.3.4 Diagramas de secuencias del diseño

La aplicación tiene dos escenarios, uno cuando el usuario tiene red Wifi y otro cuando no tiene. A continuación, se presentan los diagramas de secuencia para cada uno de los escenarios:

Diagramas cuando se está conectado al servidor de GESPRO

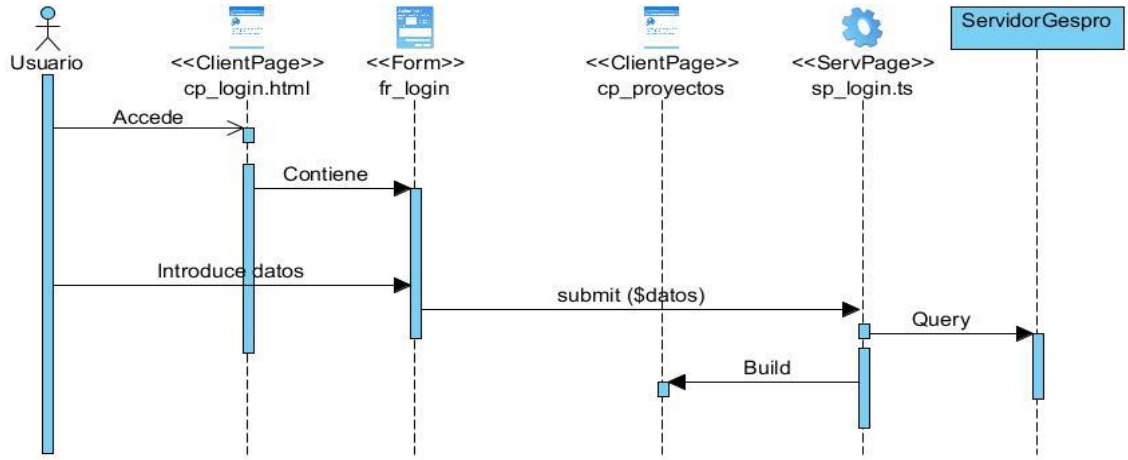


Figura 7. Autenticar usuario

En la ***Figura 7*** se muestra la secuencia que tiene la aplicación cuando un usuario se autentica; este accede a la página de autenticarse la cual contiene un formulario, el usuario introduce los datos y estos son enviados a la clase controladora la cual hace una consulta al servicio de Gespro que verifica si los datos introducidos son válidos y si es así se muestra la página de los proyectos.

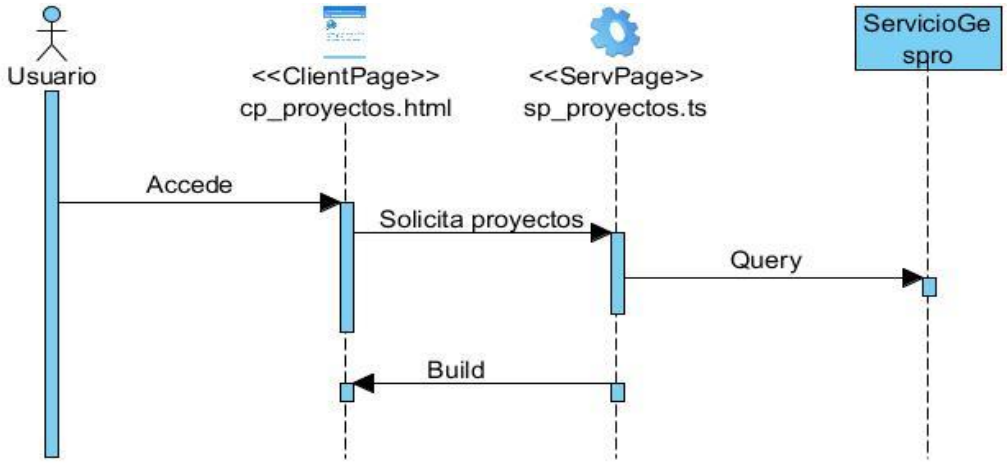


Figura 8. Visualizar proyectos

En la ***Figura 8*** se muestra la secuencia que tiene la aplicación cuando un usuario solicita ver sus proyectos; este accede a la página de los proyectos y los solicita a la clase controladora sus proyectos la cual hace una consulta al servicio de Gespro para buscar los proyectos de ese usuario.

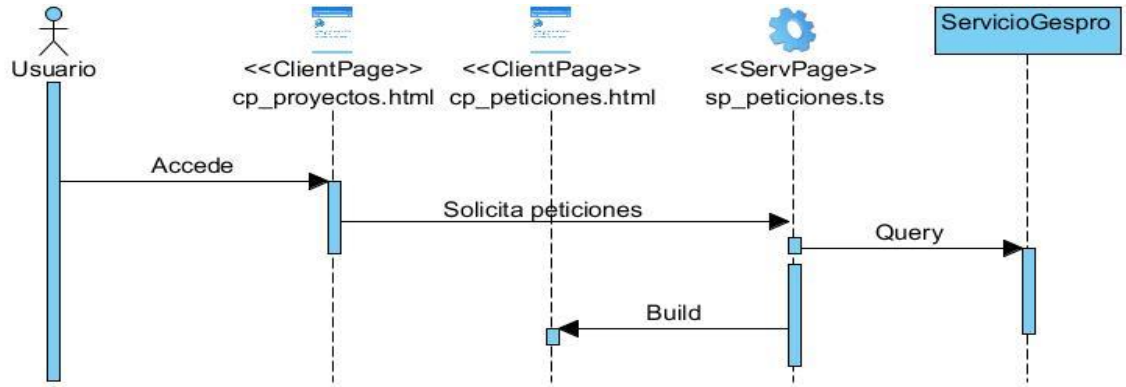


Figura 9. Visualizar peticiones

En la ***Figura 9*** se muestra la secuencia que tiene la aplicación cuando un usuario solicita ver las peticiones de un proyecto; este accede a la página de los proyectos y solicita a la clase controladora de las peticiones las peticiones de ese proyecto, la cual hace una consulta al servicio de Gespro para buscar las peticiones de ese proyecto y se muestra la página con las peticiones.

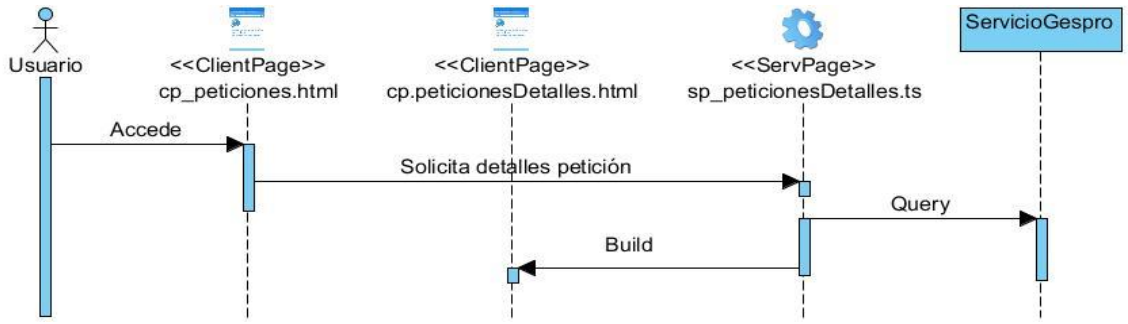


Figura 10. Visualizar peticionesDetalles

En la ***Figura 10*** se muestra la secuencia que tiene la aplicación cuando un usuario solicita ver los detalles de una petición; este accede a la página de las peticiones y solicita a la clase controladora de los detalles de las peticiones lo detalles de la petición seleccionada, la cual hace una consulta al servicio de Gespro para buscar los detalles de esa petición y se muestra la página con todos los detalles de la petición.

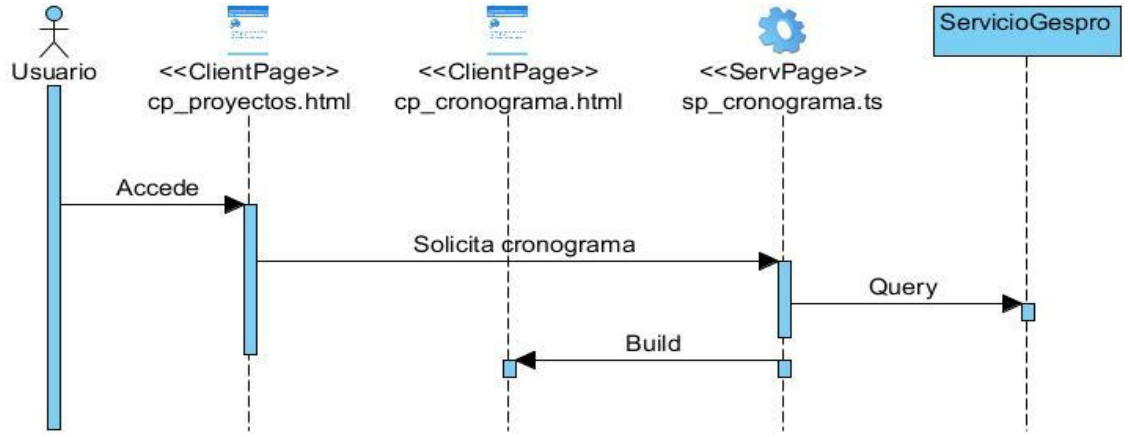


Figura 11. Visualizar cronograma

En la ***Figura 11*** se muestra la secuencia que tiene la aplicación cuando un usuario solicita ver el cronograma de un proyecto; este accede a la página de los proyectos y solicita a la clase controladora del cronograma de los proyectos el cronograma del proyecto seleccionado, la cual hace una consulta al servicio de Gespro para buscar el cronograma y se muestra la página con el cronograma.

Diagramas cuando se está desconectado del servidor de GESPRO

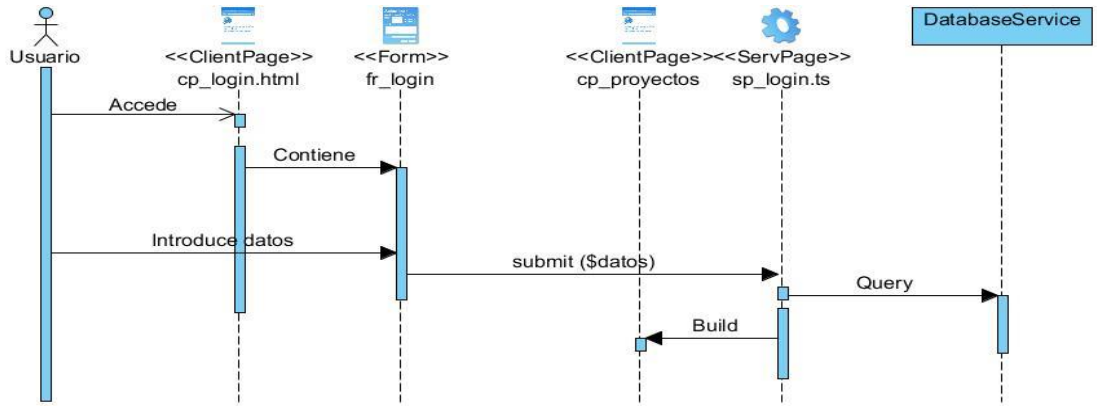


Figura 12. Autenticar usuario

En la ***Figura 12*** se muestra la secuencia que tiene la aplicación cuando un usuario se autentica; este accede a la página de autenticarse la cual contiene un formulario, el usuario introduce los datos y estos son enviados a la clase controladora la cual hace una consulta al servicio de la base de datos que verifica si los datos introducidos son válidos y si es así se muestra la página de los proyectos.

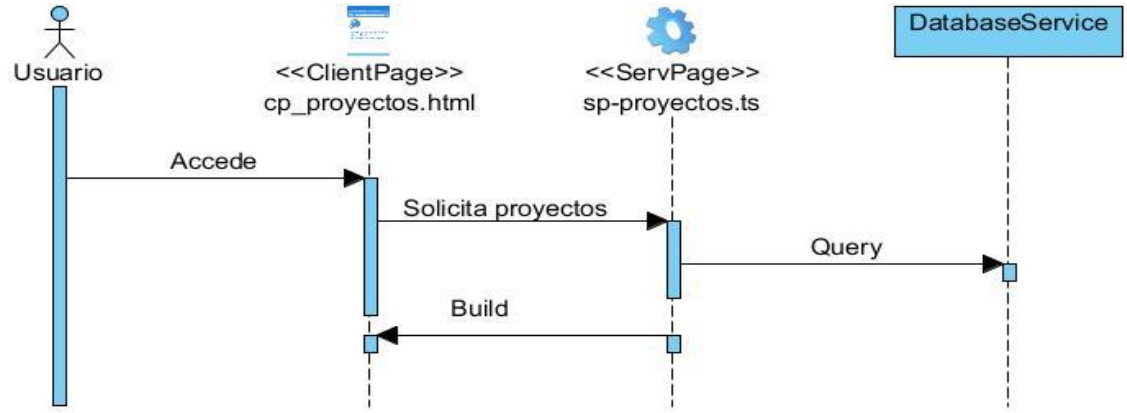


Figura 13. Visualizar proyectos

En la ***Figura 13*** se muestra la secuencia que tiene la aplicación cuando un usuario solicita ver sus proyectos; este accede a la página de los proyectos y los solicita a la clase controladora la cual hace una consulta al servicio de la base de datos de la aplicación móvil para buscar los proyectos de ese usuario.

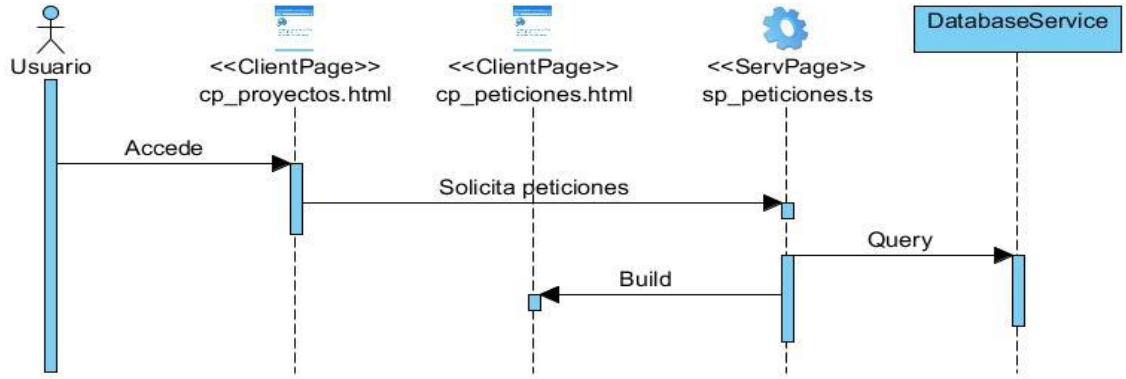


Figura 14. Visualizar peticiones

En la ***Figura 14*** se muestra la secuencia que tiene la aplicación cuando un usuario solicita ver las peticiones de un proyecto; este accede a la página de los proyectos y solicita a la clase controladora de las peticiones las peticiones de ese proyecto, la cual hace una consulta al servicio la base de datos de la aplicación móvil para buscar las peticiones del proyecto.

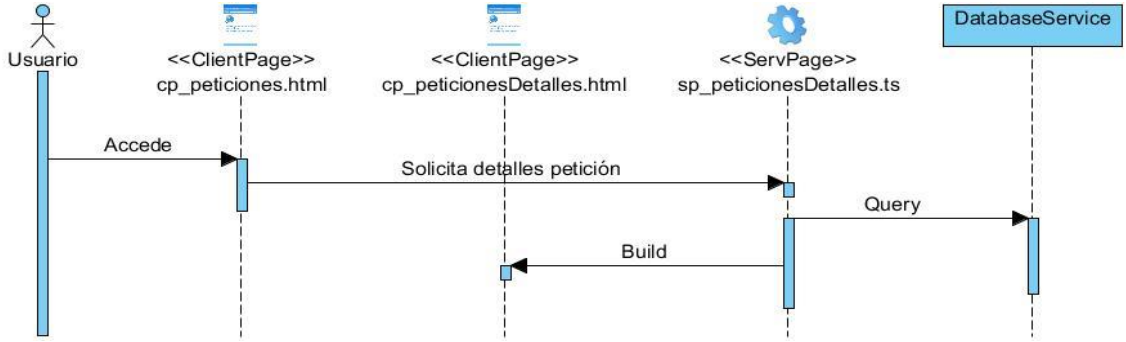


Figura 15. Visualizar peticionesDetalles

En la ***Figura 15*** se muestra la secuencia que tiene la aplicación cuando un usuario solicita ver los detalles de una petición; este accede a la página de las peticiones y solicita a la clase controladora de los detalles de las peticiones lo detalles de la petición seleccionada, la cual hace una consulta al servicio la base de datos de la aplicación móvil para buscar los detalles de esa petición y se muestra la página con todos los detalles de la petición.

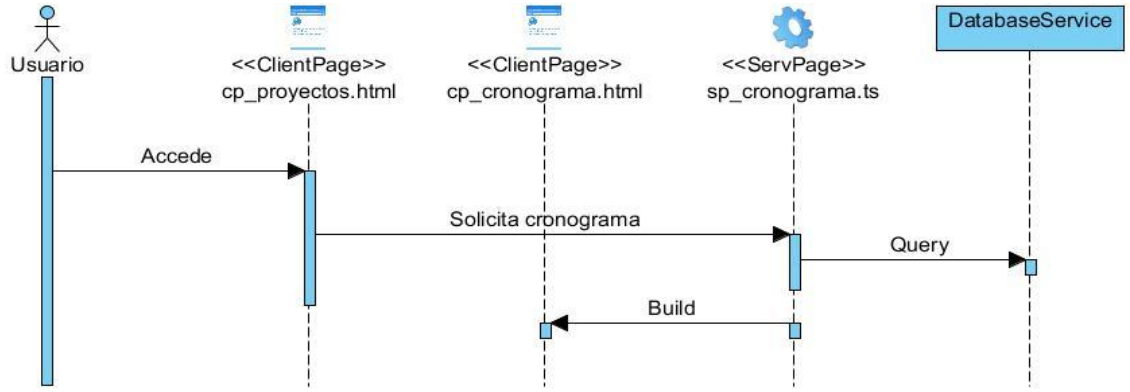


Figura 16. Visualizar cronograma

En la ***Figura 16*** se muestra la secuencia que tiene la aplicación cuando un usuario solicita ver el cronograma de un proyecto; este accede a la página de los proyectos y solicita a la clase controladora del cronograma de los proyectos el cronograma del proyecto seleccionado, la cual hace una consulta al servicio la base de datos de la aplicación móvil para buscar el cronograma y se muestra la página con el cronograma.

2.3.5 Modelo de datos

La técnica de modelado de datos más ampliamente usada es el modelado Entidad-Relación, que muestra las entidades de datos, sus atributos asociado y las relaciones entre estas entidades (Sommerville, 2005).

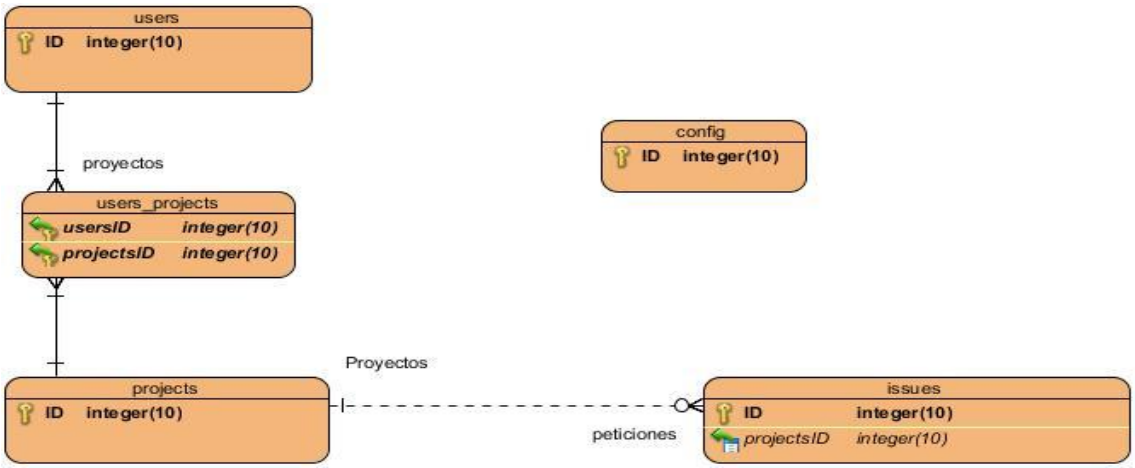


Figura 17. Modelo de datos

A continuación, se muestra la descripción de las tablas del modelo de datos representado:

Tabla 7. Descripción de las tablas del Modelo de datos

Nombre	Descripción
Usuario	Es la tabla responsable de almacenar todos los usuarios que están registrados en la herramienta de Xedro-GESPRO.
Proyectos	Es la encargada de almacenar todos los proyectos que están registrados en la herramienta de Xedro-GESPRO.
Peticiones	Tabla encargada de almacenar todas las peticiones asociadas a cada uno de los proyectos que están registrados en la aplicación de Xedro-GESPRO.
Config	Es la tabla encargada de almacenar la URL del usuario autenticado.

Consideraciones Parciales

- El modelo de dominio muestra a través de un conjunto de entidades y sus relaciones todo el funcionamiento del negocio.
- El sistema desarrollado cumple con todos los requisitos que fueron especificados, tanto funcionales como no funcionales.
- Los patrones de diseño nos ayudan a cumplir con las reglas de diseño, ya sea el control de cohesión y acoplamiento o reutilización de código son algunos de los beneficios que podemos conseguir al utilizar patrones.
- Con el propósito de representar la organización de la aplicación se desarrollaron los diagramas de clases; mostrando la relación entre clases, atributos e interfaces.
- El diseño de la base de datos aborda cómo se comporta el almacenamiento y captura de los datos.

Capítulo 3: Validación y Pruebas

En este capítulo se abordan todos los componentes que integran la etapa de validación y prueba. Se exponen las pruebas de validación mediante los diseños de casos de pruebas, las pruebas unitarias, así como los resultados obtenidos en la encuesta de satisfacción.

3.1 Diagrama de componentes

Un diagrama de componentes muestra dependencias entre los componentes, que no son más que una unidad física de implementación con interfaces bien definidas, pensada para ser utilizada como parte reemplazable de un sistema. Puede mostrar un sistema configurado, con la selección de componentes usados para construirlo o un conjunto de componentes disponibles (una biblioteca de componentes) con sus dependencias (James Rumbaugh, 2000).

A continuación, en la ***Figura 18*** se muestra el diagrama de componentes de la solución:

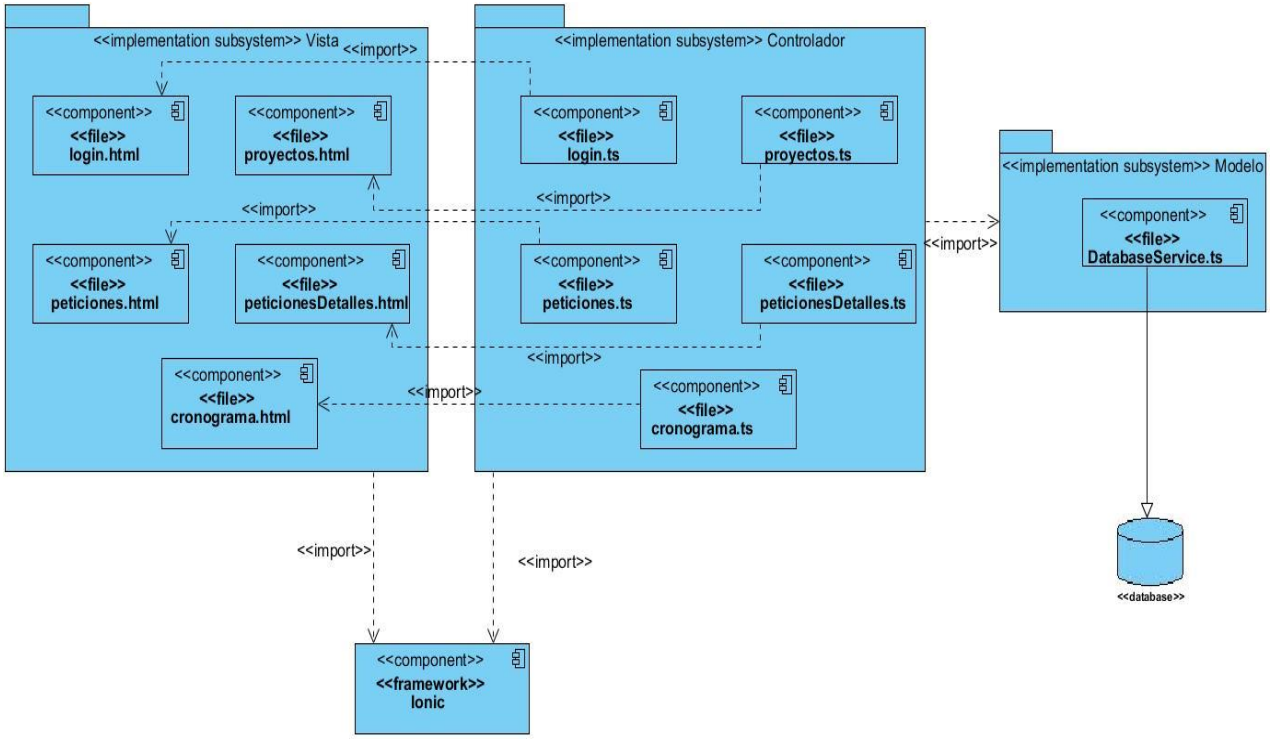


Figura 18. Diagrama de componentes

En la ***Figura 18*** se muestra el diagrama de componentes; hay un paquete donde se encuentran todos los componentes que conforman la vista, otro paquete donde están las clases controladoras que es donde está la lógica de la aplicación y hay otro paquete que es el modelo y donde se encuentra DatabaseService que es donde se crean todas las tablas de la base de datos SQLite. Además, hay un componente que es el framework Ionic y la base de datos SQLite. Los paquetes de la vista y del controlador usan de Ionic, el paquete de los controladores es el que tiene acceso al modelo y el modelo es el que trabaja directamente con la base de datos SQLite; además cada componente del paquete de los controladores depende de cada componente de la vista.

En la ***Tabla 8*** se muestra el caso de prueba correspondiente a la funcionalidad de

Tabla 8. Caso de prueba Autenticar usuario

Id del escenario	Escenario	Variable 1 (Usuario)	Variable 2 (Contraseña)	Respuesta del Sistema	Resultado de la prueba
EC 1	Autenticar usuario	V	I	Usuario o contraseña incorrecta.	Al hacer las pruebas introduciendo los datos de las cuatro formas diferentes el sistema muestra el mismo resultado del sistema.
		I	V	Usuario o contraseña incorrecta.	
		I	I	Usuario o contraseña incorrecta.	
		V	V	Muestra la página de los proyectos.	

autenticar usuario. Al introducir los datos de usuario y contraseña estos pueden tomar valores válidos (*V*) o inválidos (*I*) y dependiendo de estos valores es la respuesta que tiene el sistema.

3.2 Pruebas de validación

3.2.1 Caso de Prueba

3.2.1.1 Descripción General: La historia de usuario se inicia cuando el usuario introduce su usuario y contraseña y selecciona la opción Autenticar.

3.2.1.2 Condiciones de Ejecución: El usuario debe llenar todos los campos del formulario.

3.2.1.3 Secciones a probar en la historia de usuario.

En la ***Tabla 8*** se muestra el caso de prueba correspondiente a la funcionalidad de autenticar usuario. Al introducir los datos de usuario y contraseña estos pueden tomar valores válidos (*V*) o inválidos (*I*) y dependiendo de estos valores es la respuesta que ofrece el sistema.

A continuación, en la ***Figura 19 y 20*** se muestra la similitud existente en las interfaces de la aplicación Xedro-GESPRO web y la aplicación móvil, cuando el usuario introduce los datos correctamente.

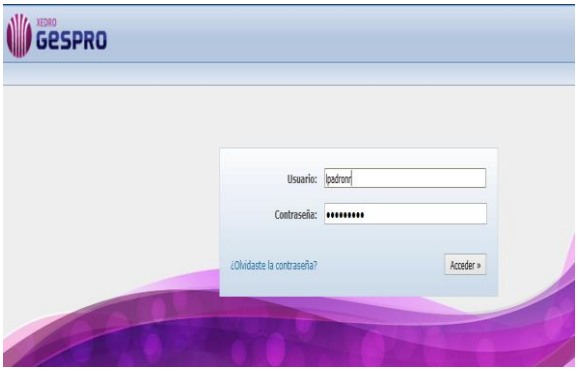


Figura 19. Autenticar con datos correctos. Xedro-GESPRO web

Figura 20. Autenticar con datos correctos. Aplicación Xedro-GESPRO móvil

A continuación, en la ***Figura 21 y 22*** se muestra la similitud existente en las interfaces de la aplicación Xedro-GESPRO web y la aplicación móvil, cuando el usuario introduce datos incorrectos. Se puede observar que al introducir usuario y contraseña inválidos las dos aplicaciones muestran el mismo cartel de error.

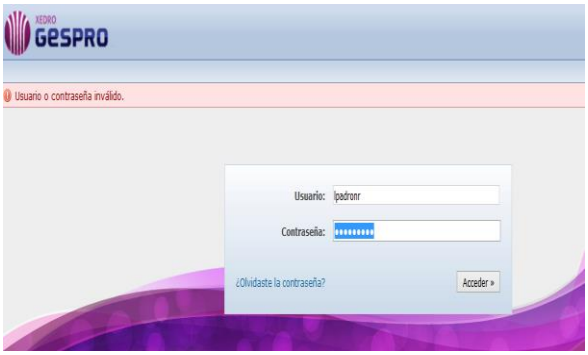


Figura 21. Autenticar con datos incorrectos. Xedro-GESPRO web

Figura 22. Autenticar con datos incorrectos. Aplicación Xedro GESPRO móvil

3.2.2.1 Descripción General: La historia de usuario se inicia luego de que el usuario se ha autenticado donde se muestran los proyectos a los cuales está vinculado.

3.2.2.2 Condiciones de Ejecución: El usuario debe de estar autenticado.

3.2.2.3 Secciones a probar en la historia de usuario.

A continuación, en las ***Figuras 23 y 24*** se muestra la similitud existente en las interfaces de la aplicación Xedro-GESPRO web y la aplicación móvil en la interfaz de visualizar proyectos. Se puede observar que ambas aplicaciones muestran los proyectos con una descripción del mismo.

Figura 23. Visualizar proyectos. Xedro-GESPRO web

Figura 24. Visualizar proyectos. Aplicación Xedro-GESPRO móvil

3.2.3.1 Descripción General: La historia de usuario se inicia luego de que el usuario visualiza sus proyectos donde puede seleccionar las peticiones que tiene ese proyecto.

3.2.3.2 Condiciones de Ejecución: El usuario debe de estar autenticado y estar vinculado al menos a un proyecto.

3.2.3.3 Secciones a probar en la historia de usuario.

A continuación, en las ***Figuras 25 y 26*** se muestra la similitud existente en las interfaces de la aplicación Xedro-GESPRO web y la aplicación móvil en la interfaz de visualizar peticiones. Se puede observar que ambas aplicaciones muestran las peticiones, lo que la aplicación web lo muestra en forma de cuadro y la aplicación móvil en forma de tarjetas.

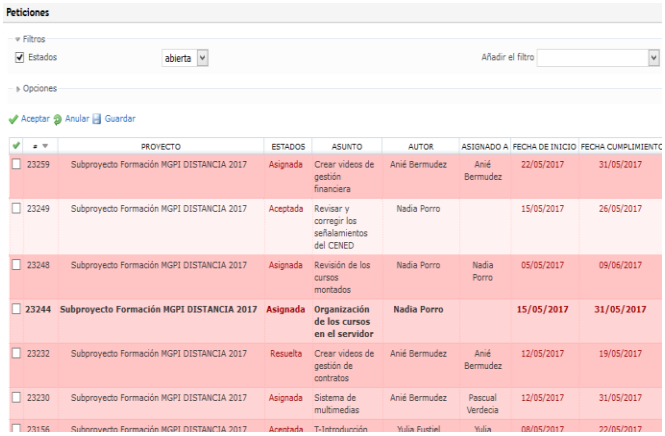


Figura 25. Visualizar peticiones. Xedro-GESPRO web

Figura 26. Visualizar peticiones. Aplicación Xedro-GESPRO móvil

3.2.4.1 Descripción General: La historia de usuario se inicia luego de que el usuario visualiza las peticiones de un proyecto, donde puede seleccionar en la opción "*Ver"* de cada una de las peticiones para visualizar los detalles de la misma.

3.2.4.2 Condiciones de Ejecución: El usuario debe de estar autenticado y estar vinculado al menos a un proyecto.

3.2.4.3 Secciones a probar en la historia de usuario.

A continuación, en la ***Figura 27 y 28*** se muestra la similitud existente en las interfaces de la aplicación Xedro-GESPRO web y la aplicación móvil en la interfaz de visualizar detalles de las peticiones. Se puede observar que ambas aplicaciones muestran los detalles de las peticiones, lo que la aplicación web lo muestra en la parte derecha del cronograma, donde cambiamos cada una de las 4 pestañas que tiene y vemos los detalles y la aplicación móvil cuando presionas en el botón "*Ver"* se muestran todos los detalles que tiene esa petición.

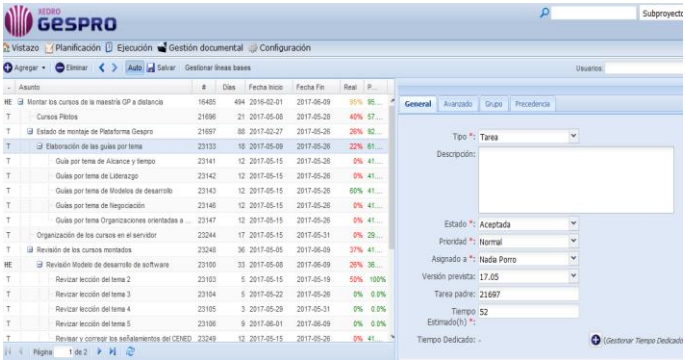


Figura 27. Visualizar detalles de peticiones. Xedro-GESPRO web

Figura 28. Visualizar detalles de peticiones. Aplicación Xedro-GESPRO móvil

3.2.5.1 Descripción General: La historia de usuario se inicia luego de que el usuario visualiza sus proyectos donde puede seleccionar el cronograma que tiene ese proyecto.

3.2.5.2 Condiciones de Ejecución: El usuario debe de estar autenticado y estar vinculado al menos a un proyecto.

3.2.5.3 Secciones a probar en la historia de usuario.

A continuación, en las ***Figuras 29 y 30*** se muestra la similitud existente en las interfaces de la aplicación Xedro-GESPRO web y la aplicación móvil en la interfaz de visualizar cronograma.

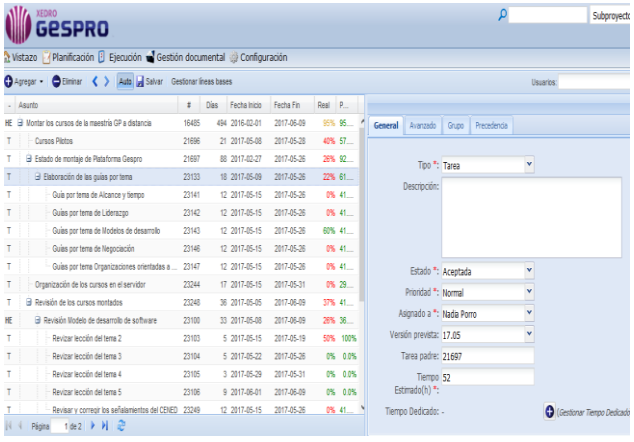


Figura 29. Visualizar cronograma. Xedro-GESPRO web

Figura 30. Visualizar cronograma. Aplicación Xedro-GESPRO móvil

La solución fue desarrollada teniendo en cuenta el diseño, las características y funcionalidades de la aplicación Xedro-GESPRO web.

3.3 Pruebas de unitarias

Se realizaron tres iteraciones de pruebas al constructor de la clase myApp, al método *login()* de la clase login y al método *serverUrl()* de la clase DatabaseServive donde fueron detectados 3 errores:

- El servicio de la base de datos no se inicializó correctamente.
- La URL del servidor no tenía valor por defecto.
- La petición al servidor estaba mal formada por lo que la respuesta era fallida.

Las siguientes figuras muestran los errores detectados en cada una de las iteraciones:

Figura 31. Errores en la 1ra iteración

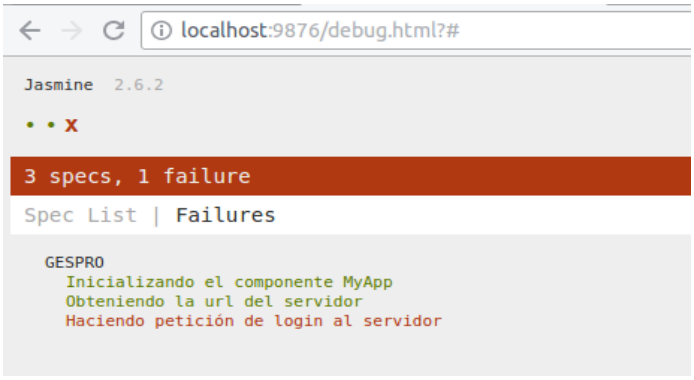


Figura 32. Errores en la 2da iteración

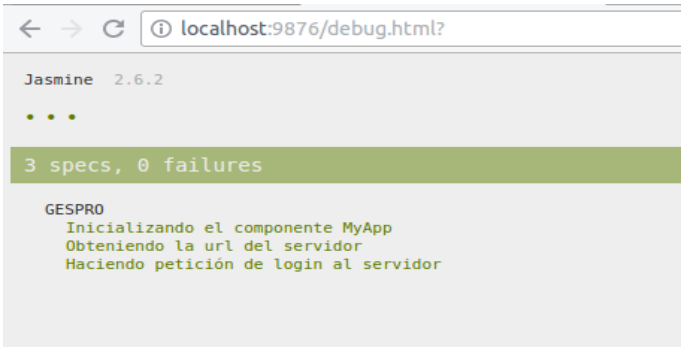


Figura 33. Errores en la 3ra iteración

La siguiente gráfica muestra el resultado por iteraciones de las pruebas unitarias realizadas:

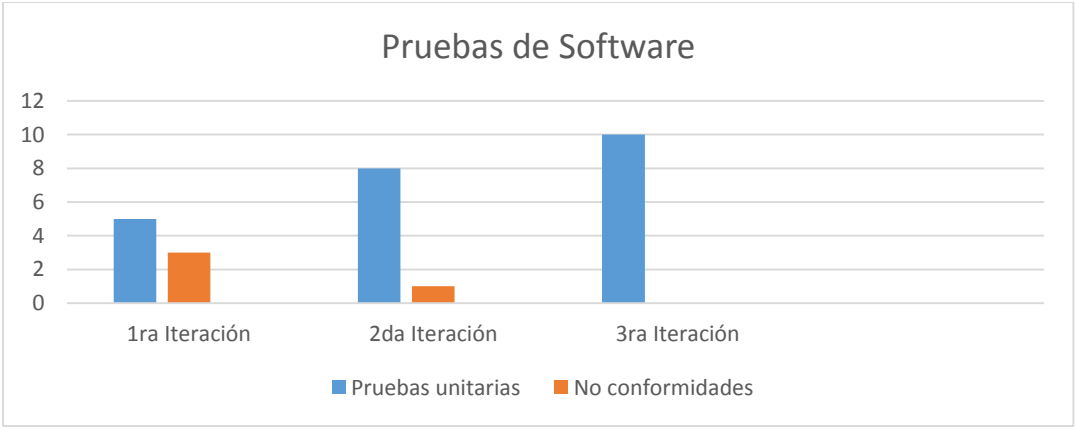


Figura 34. Resultados de las pruebas unitarias

3.4 Encuesta a especialistas

Luego de realizada la encuesta a los especialistas para verificar el nivel de satisfacción de la aplicación Xedro-Gespro móvil; teniendo en cuenta los parámetros de portabilidad, accesibilidad y diseño de la aplicación se realizó un análisis obteniendo un 100% de satisfacción de la aplicación.

A continuación, la siguiente gráfica muestra los resultados obtenidos:

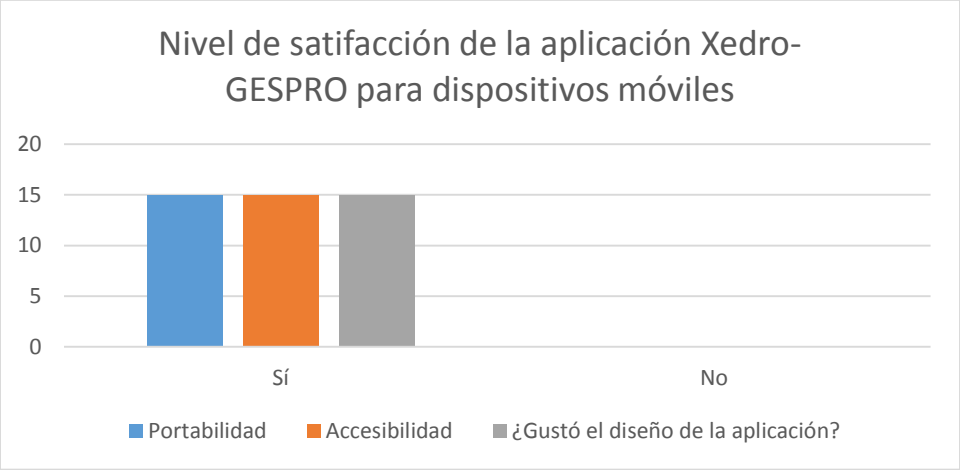


Figura 35. Resultados de la encuesta de satisfacción.

El resultado de la encuesta también arrojó sugerencias, las cuales son:

- Seguir desarrollando las funcionalidades de la aplicación para extender su uso.
- Hacer la sincronización bidireccional.
- Aplicar al resto de los módulos de Xedro-GESPRO.
- En las peticiones en la parte donde dice "*Buscar"* especificar si es por nombre o por otro elemento.
- Que se integre con otras aplicaciones que podrían brindarse a los usuarios/clientes de Gespro.
- Mejorar la interfaz.
- Extender al tablero de control.

Consideraciones Parciales

- Las pruebas de validación realizadas mediante los casos de prueba permiten ver cómo cada una de las interfaces de la herramienta Xedro-GESPRO para dispositivos móviles cumple con la estructura que posee esta herramienta para la versión web; logrando así que la versión móvil garantice la correcta funcionalidad de la aplicación.
- Las pruebas unitarias arrojaron en la 1ra iteración 3 no conformidades de las cuales fueron solucionadas 2 encontrándose en la 2da iteración 1 no conformidad la cual fue corregida dando paso a que en la 3ra iteración no se detectaran no conformidades.
- La encuesta de satisfacción permitió ver el nivel de satisfacción de los especialistas de Gespro con respecto a la aplicación Xedro-GESPRO móvil.

Conclusiones Generales

Una vez finalizado el trabajo se puede arribar a las siguientes conclusiones:

- Con la aplicación Xedro-GESPRO móvil, se logra que los usuarios accedan al módulo de peticiones del Xedro-GESPRO web desde su dispositivo móvil independientemente de la conectividad.
- Las pruebas de validación demostraron que la versión móvil de Xedro-GESPRO garantiza confidencialidad, pues para el acceso a ella el usuario debe autenticarse con los mismos datos de autenticación que utiliza para el acceso al Xedro-GESPRO web.
- Las pruebas de validación demostraron que la versión móvil de Xedro-GESPRO web garantiza integridad de los datos, pues la información mostrada en Xedro-GESPRO móvil es fiel a la registrada en Xedro-GESPRO web.
- Las encuestas aplicadas a los especialistas corroboran que la aplicación propuesta en este trabajo de diploma cumple con los objetivos propuestos, es decir, mejorar la portabilidad y la accesibilidad de la suite Xedro-GESPRO web.

Recomendaciones

A partir de los resultados de este trabajo de diploma surgieron las siguientes recomendaciones:

1. Hacer la sincronización bidireccional, es decir, lograr que la actualización de los datos no sea únicamente desde el servidor de la aplicación web para la versión móvil, sino también en el sentido contrario; de esta forma el usuario podría modificar sus tareas desde su dispositivo móvil y éstas se actualizarían en el Xedro-GESPRO web.
2. Aplicar al resto de los módulos de Xedro-GESPRO. De forma escalonada se podrían ir incorporando el resto de los módulos para la tecnología móvil y así extender la usabilidad del Xedro-GESPRO web en dispositivos móviles.

Bibliografía

Alsitecno.com. (22 de septiembre de 2014). Obtenido de Alsitecno.com: alsitecno.com/2014/09/22/historia-de-las-tablets/

Docs. (2017). Obtenido de Docs: https://ionicframework.com/docs/intro/concepts/

Ecured. (2017). Obtenido de Ecured: https://www.ecured.cu/Extreme_Game_Development

Ecured. (Marzo de 2017). Obtenido de Ecured: https://www.ecured.cu/Patrones_Gof

Emmanuel Valverde Ramos, P. H.-M. (2016). *TypeScript.*

García, I. J. (2012). *Modelo para el control de la ejecución de proyectos basado en índicadores y lógica borrosa.* La Habana.

Hampton Catlin, N. W. (2015). *Sass, el manual oficial.*

James Rumbaugh, I. J. (2000). *El Lenguaje Unificado de Modelado. Manual de referencia.* Madrid.

Jasmine. (2017). Obtenido de Jasmine: https://jasmine.github.io/

Karma. (2017). Obtenido de Karma: http://karma-runner.github.io/1.0/intro/how-it-works.html

Larman, C. (2003). *UML y Patrones.Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2da Edición.* Madrid.

Lisandro Delía, N. G. (2013). Un Análisis Experimental de Tipo de Aplicaciones para Dispositivos Móviles. En XVIII Congreso Argentino de Ciencias de la Computación.

Mateu, C. (2004). *Desarrollo de aplicaciones web.* Barcelona.

Menéndez, J. R. (2017). *SQLite como nunca antes te lo habían contado.*

Pablo Lledo, G. R. (2007). *Gestión de Proyectos. Comdo dirigir proyectos exitos, coordinar los recursos humanos y administrar los riesgos.* Buenos Aires.

Proyect Manager Institute, I. (2013). *Guías de los fundamentos para la dirección de proyectos(Guía del PMBOK) Quinta edición.* Pensilvania. Obtenido de https://www.gob.mx/cms/uploads/attachment/file/79535/PMBOK_5ta_Edicion_Espanol__1_.pdf

Punto Geek. (14 de enero de 2011). Obtenido de Punto Geek: http://www.puntogeek.com/2011/01/14/breve-historia-de-los-smartphones/

Rivas, S. (28 de Marzo de 2015). *Agile Zone*. Obtenido de Agile Zone: http://agilezone.blogspot.com/

Rodríguez, M. (2013). *Definición de una arquitectura para aplicaciones móviles.* España.

Sánchez, T. R. (2015). *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana.

(2008). *SELECTING A DEVELOPMENT APPROACH .* Obtenido de https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf

Sommervile, I. (2006). *Ingeniería del software .8va Edición.*

Sommerville, I. (2005). *Ingeniería del software* (7ma edición ed.). Madrid.

Somos Comunicación. (24 de Junio de 2016). Obtenido de Somos Comunicación: http://www.somoscomunicacion.net/blog/historia-evolucion-de-los-smartphones/

Stallman, R. M. (2007). *GNU Emacs.*

Yenisleidy Fernández Romero, Y. D. (2012). Patrón Modelo-Vista-Controlador. *Revista digital de las Tecnologías de la Información y las Comunicaciones, 11*(1).

Yoandri Quintana Rondón, L. C. (2011). Diseño de la Base de Datos para Sistemas de Digitalización y Gestión de Medias. *8*.

Anexos

Anexo 1. Manual de instalación y configuración de las herramientas utilizadas.

Node JS

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor.

sudo apt install nodejs

NPM

Del témino inglés: Node Package Manager, es un gestor de paquetes para el lenguaje de programación JavaScript. Es el gestor de paquetes predeterminado para el entorno de ejecución de JavaScript Node.js. Los pasos para su instalación son:

sudo apt install npm

Para configurar NPM para el servidor de dependencia de la UCI, en el archivo ~/.npmrc del perfil del usuario añadir:

registry = http://nexus.prod.uci.cu/repository/npm-all

strict-ssl = false

Ionic y Cordova

Ionic permite a los desarrolladores crear aplicaciones móviles con un diseño más semejante al de una aplicación nativa usando HTML5 y AngularJS y Cordova es un marco de desarrollo móvil de código abierto que permite utilizar las tecnologías estándar web como HTML5, CSS3 y JavaScript para desarrollo multiplataforma, evitando el lenguaje de desarrollo nativo de cada plataforma móvil. Los pasos para su instalación son:

$ npm install -g ionic cordova

-Crear nuevo proyecto

ionic start myApp blank –v2

-Añadir plataforma

ionic platform add Android

SDK Android

Del témino inglés: *Software Development Kit,* es un kit de desarrollo de software el cual es el lenguaje de programación de Android.

Descomprimir el SDK y abrir el archivo ~/.bashrc

export ANDROID_HOME=<dirección donde se encuentra>

PATH=${PATH}:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools

Emacs

Es un editor de textos extensible, personalizable y mucho más. Para su instalación escriba la siguiente línea de código en la terminal:

sudo apt install Emacs

-Para correr el proyecto en un móvil.

ionic cordova run android --device

Jasmine y Karma

1- npm install -g karma-cli

2- npm install --save-dev @types/jasmine @types/node html-loader jasmine karma karma-webpack ts-loader karma-sourcemap-loader karma-jasmine karma-jasmine-html-reporter angular2-template-loader karma-chrome-launcher null-loader typescript@latest

3-https://nexus.prod.uci.cu/repository/github-proxy/roblouie/ionic-unit-test-config/archive/master.zip

Extraer en la carpeta test-config

4- Añadir en la etiqueta "scripts" del archivo package.json lo siguiente: "test": "karma start ./test-config/karma.conf.js"

*5- Añadir en el arreglo "exclude" del archivo tsconfig.json lo siguiente: "src/**/*.spec.ts"*

6- Escribir las pruebas en archivos con extensión .spec.ts

Para correr las pruebas

npm test

Anexo 2. Manual de instalación y configuración de SQLite.

Para adicionar el plugin de SQLite se escribe el siguiente comando:

ionic plugin add cordova-sqlite-storage

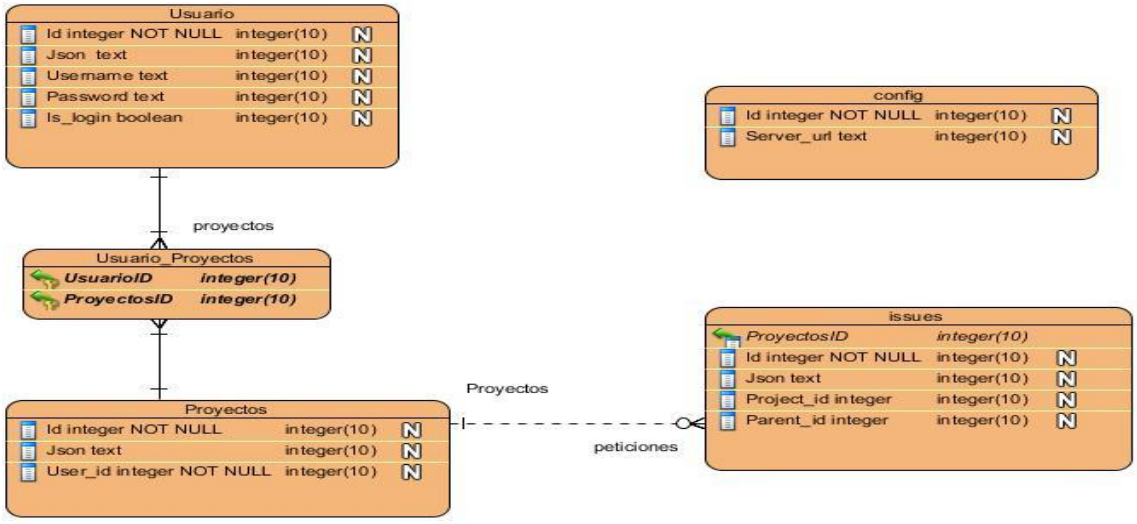


Figura 36. Modelo de datos

A continuación, se muestra la descripción de las tablas del modelo de datos representado:

Tabla 9. Descripción de las tablas del modelo de datos

Nombre	Descripción
Usuario	Es la tabla responsable de almacenar todos los usuarios que están registrados en la herramienta de Xedro-GESPRO; tiene como atributos un identificador para el usuario y también la notación de los objetos de JavaScript (JSON) que guarda toda la información que tiene el usuario en la aplicación.
Proyectos	Es la encargada de almacenar todos los proyectos que están registrados en la herramienta de Xedro-GESPRO; tiene como atributos un identificador para cada uno de los proyectos y también la notación de los objetos de JavaScript (JSON) que guarda toda la información que tienen los proyectos en la aplicación.
Peticiones	Tabla encargada de almacenar todas las peticiones asociadas a cada uno de los proyectos que están registrados en la aplicación de Xedro-GESPRO; tiene como atributos un identificador para cada una de las peticiones, la notación de los objetos de JavaScript (JSON) que guarda toda la información que tienen las peticiones en la aplicación, el identificador del proyecto al que pertenece dicha petición y si una petición es hija de otra también se guarda el identificador de la petición padre.
Config	Es la tabla encargada de almacenar la URL del usuario autenticado; tiene como atributos un identificador para la URL y la variable server_url en la cual se guarda la URL.

Anexo 3. Pasos para la sincronización de la base de datos SQLite con la base de datos de GESPRO.

Para explicar este paso se tomó como ejemplo la sincronización con el método de Visualizar Proyectos.

Paso 1. Adquirir información (proyectos) de GESPRO a través de la URL.

- dbService: Servicio donde se almacena la base de datos.

- serverURL: variable donde se guarda la URL introducida.
- **projects.**json: extrae el objeto *proyects* del json que se obtuvo de la URL.
- this.dbService.serverUrl: pide la URL al servicio de la base de datos.
- this.dbService.loginUser.api_key: devuelve la llave del usuario.
- map(…): convierte la respuesta en formato json.
- data: variable donde se guarda la respuesta.
- saveProjectsToDB(…): guarda los proyectos en la base de datos.

```
getProjectsFromServer() {
        this.http.get("https://" + this.dbService.serverUrl +
"/projects.json?key="                                          +
this.dbService.loginUser.api_key).map(res                     =>
res.json()).subscribe(data => {

            this._ngZone.run(() => {
                this.saveProjectsToDB(data.projects);
            });
        }, err => {
            this.readProjectsFromDB();
        });
    }
```

Paso 2. Guardar los datos en la base de datos SQLite.

- this.dbService.db.executeSql("DELETE FROM projects",{}): eliminar proyectos de la base de datos los actualizados.
- for (…) { if (entry.status != 5) {…}}: recorriendo proyectos y comprobando que su estado sea distinto de cerrado.
- this.dbService.db.executeSql("INSERT…"'): guarda los proyectos en la base de datos.
- JSON.stringify(entry): convierte el objeto json a texto.
- this.Proyectos = undefined: limpia los proyectos de la vista.

```
saveProjectsToDB(projects) {
    this.dbService.db.executeSql("DELETE FROM projects", {}
        ).then((d) => {
            console.log(d);
```

```
        }, (error) => {
            console.log(error);
        });

        for (let entry of projects) {
            if (entry.status != 5) {

                this.dbService.db.executeSql("INSERT INTO
projects (id,json) VALUES(" +
                    "" + entry.id + ",'" +
JSON.stringify(entry) + "')", {}
                ).then((data) => {
                    console.log(data);
                }, (error) => {
                    console.log(error);
                });
            }
        }

        this.Proyectos = undefined;
        this.readProjectsFromDB();
```

Paso 3. Leer los datos de la base de datos SQLite.

- if (this.Proyectos == undefined) {…}: verifica que la vista de proyectos esté limpia.
- this.dbService.db.executeSql("SELECT…"): selecciona de la base de datos los proyectos.
- let tasks = []: crea el arreglo temporal *task.*
- for (…): ciclo para recorrer la lista *data* que contiene los proyectos.
- tasks.push(JSON.parse(…).json): inserta en formato json en el arreglo *task*, los proyectos recorridos en *data.*
- this.Proyectos = tasks: guarda en la variable *Proyectos* el contenido del arreglo *task.*
- <div *ngFor='let value of Proyectos'>: línea del html en la que se muestran, en la interfaz del usuario, los proyectos guardados en la variable *Proyectos.*

```
readProjectsFromDB() {
        if (this.Proyectos == undefined) {
            this.dbService.db.executeSql("SELECT * FROM
projects", {}
            ).then((data) => {
                let tasks = [];
```

```
                for (let index = 0; index < data.rows.length;
index++) {

tasks.push(JSON.parse(data.rows.item(index).json));
                }
                this.Proyectos = tasks;
                console.log(data);
            }, (error) => {
                console.log(error);
            });
        }
        else {
            console.log("No Undefined");
        }
    }
```

Anexo 4. Manual de integración de la aplicación.

A continuación, en la ***Figura 37*** se evidencian los elementos que fueron necesarios integrar para obtener la aplicación Xedro-GESPRO versión móvil.

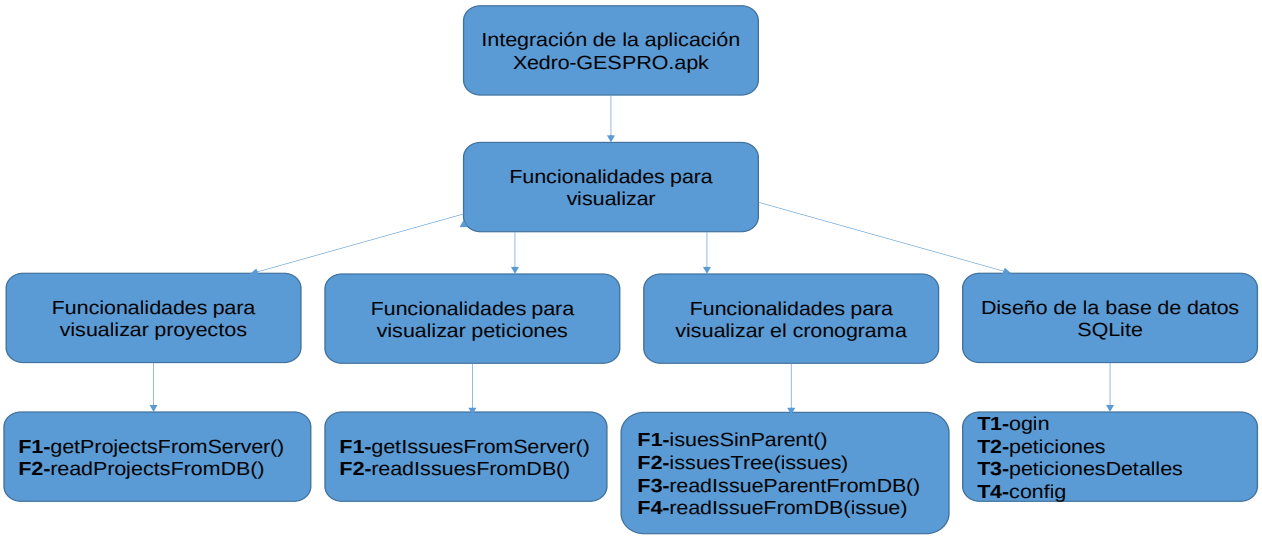


Figura 37. Integración de la aplicación

A continuación, se explican cada uno de los métodos:

getProjectsFromServer()

Este método hace una petición al servidor mediante la URL para obtener la lista de los proyectos y los guarda en la base de datos SQLite para luego de ahí ser mostradas.

readProjectsFromDB()

Este método hace una consulta pidiendo todos los proyectos a la base de datos SQLite y los muestra en la vista.

getIssuesFromServer()

Este método hace una petición al servidor mediante la URL para obtener las peticiones correspondientes a un proyecto, al hacer esta petición devuelve las peticiones padres y los guarda en una lista; luego se recorre esa lista preguntando para cada una de las peticiones padres cuales son las peticiones hijas y luego se guarda en la base de datos SQLite la información de las peticiones padres con sus peticiones hijas para luego ser mostradas.

readIssuesFromDB()

Este método hace una consulta pidiendo todas las peticiones de un proyecto a la base de datos SQLite y los muestra en la vista.

isuesSinParent()

Este método hace una petición al servidor mediante la URL para obtener las peticiones que son padres y manda a ejecutar el método issuesTree(issues) al cual se le pasa por parámetro las peticiones que son padre; en caso de no tener red manda a leer de la base de datos SQLite a través del método readIssueParentFromDB().

issuesTree(issues)

Este método recibe la lista de peticiones padres para recorrerla y por cada padre hace una consulta al servidor para que incluya los hijos y luego en la librería Tree

que es la que organiza las tareas en forma de árbol se le adiciona la petición padre con sus peticiones hijas.

readIssueParentFromDB()

Este método hace una consulta pidiendo todas las peticiones padres de un proyecto a la base de datos SQLite y crea un arreglo llamado task para guardar las peticiones padres y manda a ejecutar el método issuesTree(task) al cual se le pasa por parámetro el arreglo con las peticiones padre.

readIssueFromDB(issue)

Este método recibe la petición padre y hace una consulta pidiendo las peticiones hijas que el id del padre coincida con el id de la petición padre pasada por parámetro, luego crea un arreglo donde guarda la petición padre con sus peticiones hijas y los adiciona a la librería Tree para que las muestre en forma de árbol.

Anexo 5. Manual de usuario.

Requisitos:

- El usuario debe tener un dispositivo móvil con la versión de Android 5 o superior.
- El usuario debe tener copiado en su dispositivo la aplicación Xedro-GESPRO.apk.

Luego de cumplirse todos los requisitos el usuario debe de realizar los siguientes pasos:

1) Instalar la aplicación presionando en el botón de instalar.
2) Buscar en el menú la aplicación y seleccionarla para que se abra.
3) Luego de abierta la APK por primera vez saldrá la página de autenticación que muestra un cartel con la URL del centro CDAE.

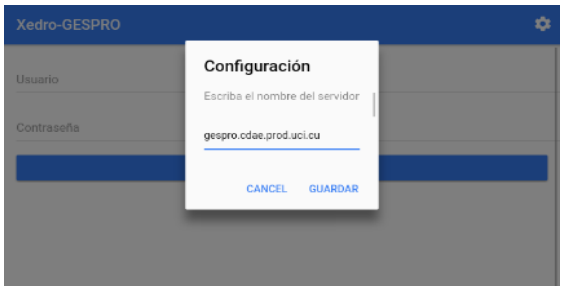


Figura 38. Inicio. Aplicación Xedro-GESPRO

4) Si el usuario es de ese centro presiona en el botón de guardar contraseña y le saldrá la página de autenticarse, en caso contrario escribe la URL de su centro y presiona en el botón guardar e igualmente se muestra la página de autenticarse.

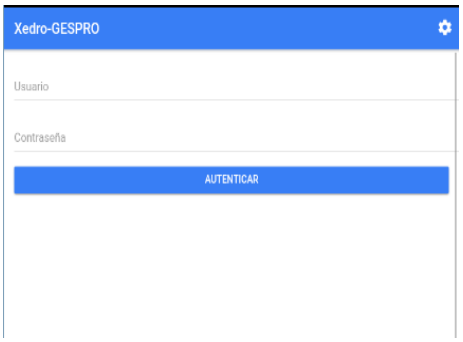


Figura 39. Autenticar. Aplicación Xedro-GESPRO

5) En caso de que se use la aplicación y no sea la primera vez, el usuario puede verificar si la dirección que está es la de su centro y cambiarla; en la página de autenticarse en la parte superior derecha aparece un botón para cambiar la dirección, luego de presionar en el botón realiza el mismo procedimiento descrito en el **Paso 4.**

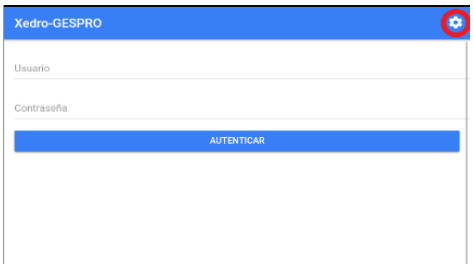


Figura 40. Configuración

6) Luego el usuario introduce su usuario y contraseña y presiona en el botón de **Autenticar**.

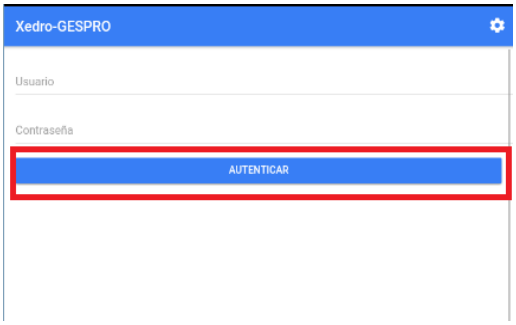


Figura 41. Acción de autenticar

7) Después de autenticarse aparecerán los proyectos a los que pertenece ese usuario y aparecerán dos botones, uno para ver las **Peticiones** que tiene el proyecto y otro para ver el **Cronograma**.

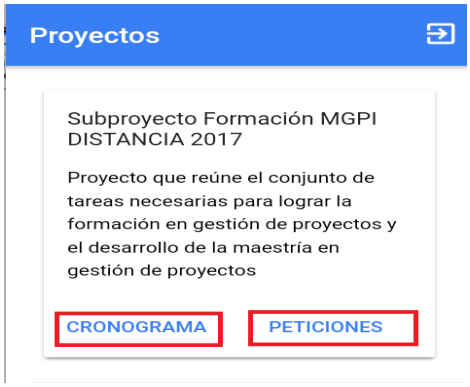


Figura 42. Visualizar proyectos

8) Si el usuario presiona el botón de **Cronograma** le saldrá el cronograma del proyecto y puede ver los detalles de las peticiones presionando encima de la petición.

Figura 43. Visualizar cronograma

9) Si el usuario presiona el botón de **Peticiones** se muestran todas las peticiones del proyecto las cuales se pueden filtrar por el nombre o por la fecha de inicio.

Figura 44. Visualizar filtros de peticiones

10) El usuario cuando está en la página de peticiones puede ver los detalles de la misma presionando en el botón "*Ver*".

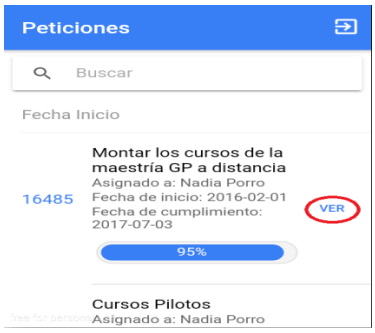


Figura 45. Acción ver detalles de las peticiones

11) Si el usuario desea salir de su sesión en las páginas de los proyectos, peticiones, detalles de las peticiones y cronograma aparece un botón en la parte superior derecha para salir.

Figura 46. Salir

Anexo 6. Encuesta de satisfacción a los especialistas de Xedro-GESPRO.

¿Cree usted que la aplicación resuelve el problema de portabilidad y accesibilidad del calendario de tareas? Marque con una X según convenga:

Portabilidad Sí___ No___

Accesibilidad Sí___ No___

¿Le gusta el diseño que posee la aplicación?

Sí___ No___

Sugerencias:

_____

_____

_____