UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA

FACULTAD DE INGENIERÍA

PRÁCTICA PROFESIONAL, HEROUNIT.

PREVIO A LA OBTENCIÓN DEL TÍTULO

INGENIERO EN SISTEMAS COMPUTACIONALES

PRESENTADO POR:

21511105 LUIS ALFREDO ÁLVAREZ RAMÍREZ

ASESOR: ING. CÉSAR ORELLANA

CAMPUS SAN PEDRO SULA

FEBRERO, 2020

DEDICATORIA Y AGRADECIMIENTOS

Primero quiero agradecer a mi familia por darme su apoyo en especial a mis padres, mi hermana y mis abuelos. A mis padres porque se esforzaron mucho en colocarme en esta carrera y darme todo su apoyo al igual que mi hermana. A mis abuelos por todos los valores que me inculcaron y por darme un ejemplo de superación. Si no hubieran estado a mi lado posiblemente no hubiera llegado a este punto.

Quiero agradecer a mis amigos por toda la ayuda que me brindaron en las clases que compartimos. También agradezco el conocimiento que me transfirieron mis profesores y por los consejos que me dieron. Para ser un profesor es una de las vocaciones más honorables que existen en el mundo.

También quiero agradecer a la empresa HeroUnit por darme la oportunidad de trabajar con ellos y también por todo lo aprendido durante mi practica profesional.

RESUMEN EJECUTIVO

Este documento tiene como propósito principal explicar todo el trabajo realizado en el proyecto al que fui asignado en la compañía HeroUnit de Acklen Avenue. La duración de la práctica profesional fue de 22 de semanas que comenzó el 7 de octubre de 2019 y finalizo el 11 de marzo de 2020.

El proyecto donde me desempeñe es llamado Pisto. Ese proyecto es conformado por varios módulos, pero los dos más importantes son PistoPOS y Move. PistoPOS es un punto de venta que satisface las necesidades de la pequeña empresa por ejemplos restaurantes, cafeterías o abarroterías. Move es un manejador de inventarios donde se crean productos más complejos, hay manejo de gastos e inventario. Move extiende el alcance de PistoPOS y satisface las necesidades de la mediana empresa como distribuidoras o una familia de restaurantes.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

I. INTRODUCCIÓN

Este informe describe todo el trabajo realizado durante la práctica profesional en la empresa desarrolladora de software, HeroUnit. El informe describe las actividades realizadas durante las 22 semana que duro la practica dicho periodo de tiempo que comenzó el 7 de octubre del 2019 y finalizo el 11 de marzo del 2020.

En el capítulo 2 se encuentra la información de la empresa, se describirá el rubro en la que se desempeña la empresa, los servicios que ofrece y se definen los objetivos del proyecto.

En el capítulo 3 nos encontramos el marco teórico donde se explicará toda la teoría relacionada con el proyecto, que metodología se siguió, se describirá los lenguajes de programación y frameworks que se usaron en el proyecto además de las herramientas que se emplearon.

En el capítulo 4 se encuentra el resumen de las actividades que se trabajaron en mi periodo de práctica donde se mostrara evidencia del trabajo realizado. Al inicio se explicará el funcionamiento de proyecto que estuve involucrado. Y al final se encuentra la bitácora donde se describirá el trabajo en semanas.

Por último, tenemos el capitulo 5 y 6 donde se encuentran las conclusiones y recomendaciones respectivamente. Estas provienen de los resultados finales de mi practica profesional en HeroUnit.

II. GENERALIDADES DE LA EMPRESA

2.1. DESCRIPCIÓN DE LA EMPRESA

HeroUnit es una compañía de la empresa Acklen Avenue, LLC. Acklen Avenue y HeroUnit son empresas que se dedican al desarrollo de software. Sus oficinas centrales están ubicadas en Nashville, Tennessee, Estados Unidos. Estas dos compañías prestan sus servicios a otras empresas que no tienen un departamento de desarrollo o necesitan ayuda para desarrollar un proyecto. La mayoría de los empleados están en Honduras en las ciudades de San Pedro Sula y Tegucigalpa.

Acklen Avenue aposto por el talento hondureño obteniendo buenos resultados. Encontrando personas que se esfuerzan por entregar lo mejor de ellos, no solo los desarrolladores de software sino también personal de Recursos Humanos, diseñadores y reclutadores de talento.

HeroUnit ofrece a los clientes los siguientes servicios:

- Soporte proactivo 24/7
- Desarrollo Web
- Desarrollo de aplicaciones móviles
- Desarrollo de Backend
- DevOps
- Gestión de proyectos

2.2. DESCRIPCIÓN DEL DEPARTAMENTO

En Acklen Avenue y HeroUnit existen varios roles: desarrolladores de software, diseñadores, DevOps, aseguradores de calidad y lideres de proyecto. Cada miembro es importante en el desarrollo, el equipo es muy colaborativo si un miembro tiene un problema los demás miembros del equipo lo apoyan para que el trabajo este en movimiento. Los lideres de proyecto son el puente entre el equipo de desarrollo y el cliente, este es capaz de comunicar los deseos del cliente al equipo de desarrollo y resolver problemas que se presenten al equipo. Los diseñadores se encargan de crear la interfaz de usuario y entregarla a los desarrolladores. Los desarrolladores de software se encargan de hacer realidad

las tareas del proyecto. Los aseguradores de calidad tienen que revisar que el trabajo entregado por los desarrolladores cumpla con los requisitos sin errores y si se presenta uno se debe comunicarse al todo el equipo. DevOps tienen la tarea de crear todas las automatizaciones como crear ambientes de pruebas y hacer los lanzamientos de las nuevas versiones para el cliente.

Acklen Avenue usa las practicas Ágiles para entregar lo mejor de ellos al cliente, donde se debe tener un cliente feliz con el proyecto para tener una mejor comunicación, pero para llegar a ese punto se debe tener un equipo comprometido con el proyecto y tener diferentes habilidades que se complementen entre si.

2.3. OBJETIVOS DEL PUESTO

2.3.1. OBJETIVO GENERAL

Agregar nuevas funcionalidades a los módulos de Pisto, usando las herramientas que nos fueron proveídas para entregar la mejor solución al cliente también aplicando las mejores prácticas de desarrollo que he aprendido. Seguir las practicas Ágiles y entender cuales son los principales objetivos de estas.

2.3.2. OBJETIVO ESPECÍFICOS

- Desarrollar las nuevas funcionalidades para proyecto Pisto usando como guía las especificaciones de este.
- Aplicar la nueva interfaz al proyecto Pisto para que el usuario sienta comodidad en usarlo.

2.4. DESCRIPCIÓN DEL PROYECTO REALIZADO

PistoPOS en un software de punto de venta, está pensado para la pequeña y mediana empresa donde los usuarios podrán gestionar productos, cajeros, gastos, descuentos, etcétera. En el inicio el producto estará para todo publico después el usuario tendrá que adquirir una suscripción dependiendo de sus necesidades por ejemplo un usuario tiene dos dispositivos entonces el mejor plan sería el que incluye dos dispositivos. PistoPOS tiene un complemento llamado Move. Move está pensado para la mediana empresa donde se podrá gestionar el inventario, gastos, compras de suministros y proveedores.

III. MARCO TEÓRICO

3.1. METODOLOGÍAS

3.1.1. PRÁCTICAS ÁGILES

Las prácticas Ágiles son un enfoque iterativo para la gestión de proyectos y el desarrollo de software que ayuda a los equipos a entregar valor a sus clientes más rápido y sin la menor cantidad de problemas. En vez de entregar todo el producto una sola entrega, las practicas Ágiles entregan pequeños incrementos de producto en periodos de tiempo establecido por el equipo de desarrollo. En las prácticas Ágiles los requisitos, los planes y entregas siempre se evalúan para mejorar aún más el proceso de desarrollo. Cabe destacar que las practicas Ágiles no sufren con los cambios que pide un cliente gracias a las iteraciones cortas en el desarrollo.

3.1.1.1. Waterfall contra Las Prácticas Ágiles

Desarrollo de software en cascada o Waterfall has sido una de la mas tradicionales. Los desarrolladores y los clientes están de acuerdo en lo que se entregará al principio del ciclo de vida del desarrollo. Esto puede hacer que la planificación y el diseño sean más sencillos (Haworth, 2019). Cuando todo el proyecto ha sido esclarecido con el cliente, el proyecto se desarrolla en fases, el equipo debe terminar a totalidad cada una de ellas para evitar problemas en el futuro, con eso parece que un proyecto va en buen camino, pero desafortunadamente en el desarrollo del software pueden surgir cambios en cualquier fase o etapa del proyecto. Una vez que un proyecto pasa una fase, es una perdida de tiempo volver a el porque ya afecta al flujo de trabajo. Por ejemplo, tenemos este ejemplo de las fases del desarrollo del proyecto.

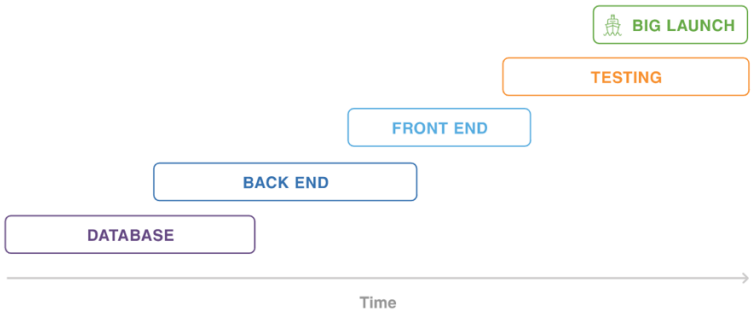


Ilustración 1 Desarrollo de un proyecto (Cascada)

Fuente: (Radigan, s/f)

Con una gestión de proyecto ágil, primero tenemos que definir que es lo que se entregara al final de la iteración para esto debe haber una reunión con el cliente para hacer las tareas de mayor prioridad. Al final de cada iteración se le muestra el avance al cliente y recibimos una retroalimentación por parte de cliente y recibiendo nuevos cambios que se pueden ejecutar en la iteración siguiente. Por ejemplo si en la iteración de estanca en el desarrollo del frontend, el equipo de desarrollo puede trabajar en otra sección del proyecto por ejemplo el backend mientras se investiga cual es el problema con el frontend. La ilustración siguiente explica la planeación de cada iteración.

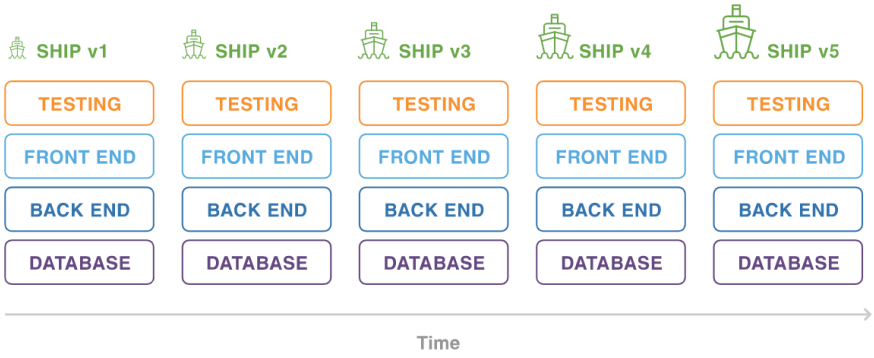


Ilustración 2 Desarrollo de un proyecto (Prácticas Ágiles)

Fuente: (Radigan, s/f)

3.1.1.2. Manifiesto de Ágil

Ágil tiene un manifiesto que contiene 12 principios que el equipo de desarrollo tiene que respetar.

1. Satisfacción de cliente: Este el objetivo de todo. Siempre se busca que el cliente este satisfecho en todo tiempo haciendo que el desarrollo sea más fácil.
2. Abierto a los nuevos cambios: Los cambios no vienen a retrasar el desarrollo sino a mejorar el producto y darle valor al cliente.
3. Entregas periódicas: es una de las ventajas de esta practica. El equipo de trabajo no este abrumado con tareas colosales.
4. Medir el progreso: Ayuda a saber como se esta desenvolviendo el equipo de desarrollo y en como mejorar las debilidades.
5. Trabajo cercano: Los lideres de proyectos siempre debe saber como esta su equipo para resolver los problemas que se le están presentado a ellos.
6. Comunicación cara a cara: todos la comunicación debe ser muy clara para evitar dudas que traigan problemas en el desarrollo del proyecto.
7. Motivación: El proyecto solo tendrán éxito si el equipo esta motivado y con confianza para entregar el mejor producto al cliente.
8. Excelencia técnica y buen diseño: El producto tiene que ser el entregado en el mejor estado posible, debe ser un producto calidad y desarrollado con las mejores prácticas de programación.
9. Simplicidad: Las tareas deben ser los más simples posibles para que no haya problemas en el desarrollo. Si una tarea es muy grande debe ser dividida en pequeñas partes para que el equipo puede abordadas.
10. Autogestión: aunque hay un líder de equipo. El equipo debe ser capaz de organizarse por si mismos.
11. Adaptación: el equipo debe ser capaz de adaptarse a los nuevos cambios que pueden surgir durante el desarrollo del proyecto.
12.

3.1.2. SCRUM

SCRUM es una metodología que nace ajena al desarrollo del software, de hecho sus principios fundamentales fueron desarrollados en procesos de reingeniería por Goldratt, Takeuchi y Nonaka en la década de 1980 (Fernández, s/f).

SCRUM se define como un proceso empírico, iterativo e incremental; empírico porque se basa en la práctica y en la observación de los procesos, iterativo porque las tareas se trabajan en periodos de tiempo definidos e incremental porque en cada iteración al final se le muestra una demostración al cliente. En SCRUM se acepta la naturaleza cambiante del desarrollo de software.

El marco SCRUM consta de equipos y sus roles, eventos, artefactos y reglas asociados. Cada componente dentro del marco cumple un propósito específico y es esencial para el éxito y uso de SCRUM (Sutherland & Schwaber, 2017).

SCRUM puede ser implementado en los siguientes trabajos:

- Investigaciones de mercados y nuevas tecnologías
- Desarrollo de productos y actualizaciones
- Liberar productos y mejoras durante el día
- Mantener productos
- Desarrollar y mantener servicios en la nube

3.1.2.1. Roles

En el framework SCRUM tenemos 4 tipos de roles, cada unos de ellos realizan un trabajo de importante para el desarrollo de un producto.

Product Owner. Este rol representa a la empresa. A veces es un empleado de la empresa o es un representante contratado por esta. Es el responsable de entregar el máximo valor a la empresa. En la manera en como se hace varia mucho de la empresa y de los individuos involucrados.

El Product Owner es la única persona de manejar el Product Backlog o la lista del producto. Las responsabilidades que debe hacer son:

- Expresar claramente las historias de usuario.
- Priorizar los elementos de la lista del producto.
- Asegurar que la lista se los mas clara para el equipo de desarrollo.
- Asegurarse que el equipo de desarrollo entienda lo suficiente para iniciar el trabajo.

El product owner no puede ser un grupo de personas porque puede surgir conflictos de intereses o pueden tomar mucho tiempo para hacer una decisión.

Para que el product owner tenga éxito, la empresa tiene que respetar las decisiones de este.

SCRUM Master. Es la persona encargada de ejecutar el marco de trabajo SCRUM. El principal objetivo del SCRUM Master es hacer que el equipo conozca las prácticas Ágiles y la metodología SCRUM.

El SCRUM Master es el líder del equipo de desarrollo y éste esta a sus servicios. El SCRUM Master ayuda a las personas externas al Equipo SCRUM a entender qué interacciones con el Equipo SCRUM pueden ser útiles y cuáles no. El SCRUM Master ayuda a todos a modificar estas interacciones para maximizar el valor creado por el Equipo SCRUM (Sutherland & Schwaber, 2017).

Equipo de Desarrollo. Este rol es encargado de crear el producto. Está conformado por personas que tienen las habilidades necesarias para crear el incremento que se entregará al final del sprint. Y solo los miembros participan en la creación del producto. Puede ser conformados por personas de diferentes rubros por ejemplo programadores, diseñadores, QA o DevOps.

3.1.2.2. Eventos

En SCRUM hay una series de actividades que tienen como objetivo de crear regularidad y evitar reuniones innecesarias que están afuera del marco SCRUM. Todos estas actividades tienen que resolverse en un periodo de tiempo o time-boxes donde deben cumplirse en ese tiempo establecido. Pero si esa actividad se cumple con un menor tiempo no hay problemas ese tiempo sobrante se puede emplear para otra actividad. Estos eventos se diseñaron específicamente para habilitar los pilares vitales de transparencia e inspección (Sutherland & Schwaber, 2017).

Sprint

El Sprint es la base de SCRUM. Es un periodo de tiempo de 1 mes, 3 o 2 semanas, durante el cual se crea un incremento de producto, utilizable y potencialmente liberable. Los Sprints no deben ser largos porque el equipo de desarrollo estará sobrecargado por la cantidad de tareas que se deberán hacer y en cierto grado se

pierde contacto con el cliente. El cliente quiere ver resultados en el menor tiempo posible. Todos los eventos de SCRUM tienen que ocurrir dentro del sprint.

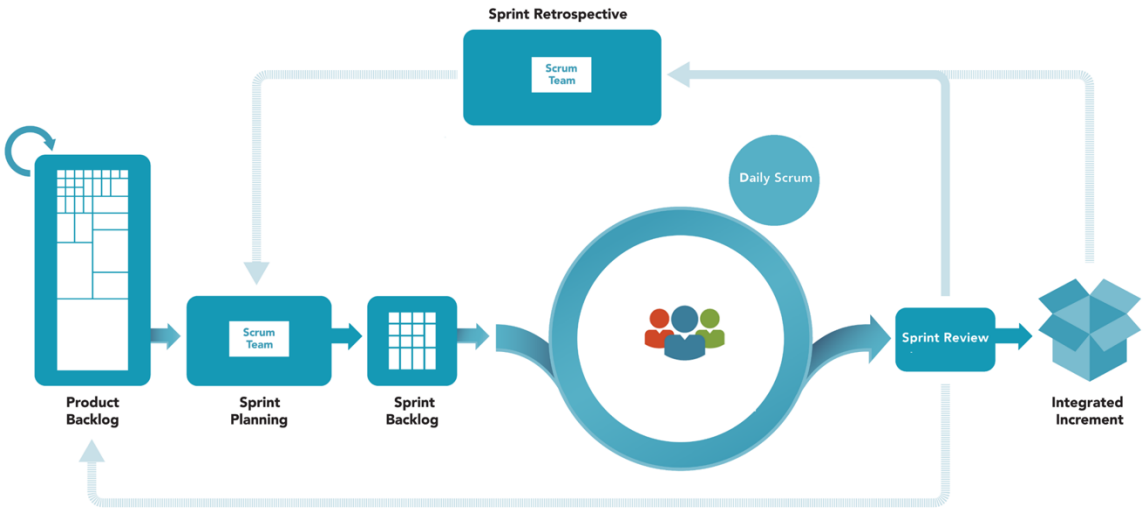


Ilustración 3 Vista general de un sprint

Fuente: (Schwaber, s/f)

Sprint Planning

El primer evento de un Sprint es el Planning y solo sucede una vez. La Planificación de Sprint tiene un máximo de duración de ocho horas para un Sprint de un mes (Sutherland & Schwaber, 2017). El time-box del planning depende de la duración de sprint. Por ejemplo, para un sprint de dos semanas el time-box del planning seria de 4 horas. El objetivo del Sprint Planning es planear que es lo que se trabajará en el sprint y como se logrará cumplirlo.

En este evento todos los roles de SCRUM están involucrados, el product owner tiene que comunicarle al equipo cuales son las prioridades para ese sprint, y trasladar las tareas del product backlog hacia el sprint backlog. Pero para trasladarlas todos tienen que estar de acuerdo que las tareas están listas para empezar.

Ilustración 4 Entrada y Salida del Sprint Planning

Fuente: Propia

Sprint Goal

El Sprint Goal es el resultado esperado del sprint implementado todo lo que contiene el sprint backlog. Todas las tareas del sprint backlog debieron ser revisadas durante el planning porque será la guía del equipo de desarrollo para que el incremento incluya todo lo que espera el product owner.

Sprint Execution

El Sprint Execution ocurre durante un sprint y casi tiene la misma duración de este y comienza después del Planning y termina antes del Review. Durante este evento el equipo de desarrollo crea el incremento que será el resultado del Execution. Durante este evento solo se debe trabajar en las tareas que se encuentran en el product backlog.

El proceso de ejecución de Sprint incluye la planificación de tareas, el desempeño, la administración de las tareas, la asistencia a los stand-ups diarios y la Comunicación con los equipos SCRUM (Knowledgehu Tutorials, 2018).

Todos los roles están involucrados en este evento. El equipo de desarrollo trabaja en las tareas del sprint eligiendo la mejor posible. El SCRUM Master debe de dirigir el equipo a la dirección correcta, debe de estar pendiente de la metodología y debe quitar los obstáculos para que no haya problemas durante el desarrollo. El Product Owner debe responder todas las dudas que surgen durante el desarrollo del incremento.

Daily Sprint

El Daily Sprint o Daily SCRUM o simplemente llamando Daily es el evento donde el SCRUM Master y el equipo de desarrollo se reúnen. Esta pequeña reunión no toma más de 15 minutos y ocurre cada día laboral de la fase de desarrollo del

proyecto. El principal objetivo de este evento es comunicar el estado del trabajo del sprint en donde cada miembro del equipo de desarrollo comunica cual fue su progreso del día anterior, que tareas harás hoy y que problemas pueden surgir.

El SCRUM Master no tiene que estar en todas las reuniones, reunión esta más enfocada al desarrollo y el equipo de desarrollo debe ser responsable que se realice. Es importante comportarse en la reunión, solo uno debe hablar y no puede ser interrumpido el objetivo de esto es para preservar la calidad de la comunicación.

Sprint Review

Durante el Sprint Review se hace una demostración del resultado final del sprint que es el incremento. Se busca que haya un feedback por parte de los presentes en la reunión. En este evento están involucrados el product owner, el SCRUM Master y el equipo de desarrollo. También estar presentes los usuarios finales, clientes y los interesados en el proyecto. El Sprint Review no puede ser la única vez que se puede recibir feedback (Francia, s/f).

La entrada de este evento es el incremento porque se debe hacer una demostración, no se debe presentar imágenes o diapositivas. Los interesados deben ver una demostración funcional mostrando todas las características agregadas que provenían del sprint backlog. El resultado de este evento son nueva entradas para el product backlog que pueden ser cambios que solicitaron los usuarios o interesados.

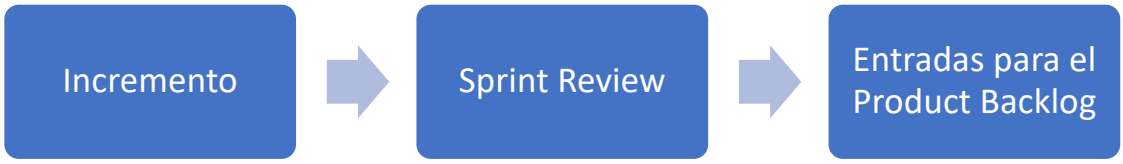


Ilustración 5 Entrada y Salida de Sprint Review

Fuente: Propia

Sprint Retrospective

En el Sprint Retrospective es una oportunidad de mejora para el equipo SCRUM de calificarse a si mismos y elaborar un plan de mejora para esas debilidades que serán abordadas en el siguiente sprint. Este evento comienza toma lugar después

del Review y antes del Planning. Se involucran el SCRUM Master y el equipo de desarrollo. El SCRUM Master tiene la responsabilidad que la reunión cumpla con el tiempo requerido. El equipo de desarrollo tiene que comprometerse con el plan de mejora y el SCRUM Master alienta el equipo a que mejore en las prácticas Ágiles y en el marco SCRUM.

3.1.2.3. Artefactos

Los Artefactos de SCRUM fueron creados para garantizar la transparencia de la información clave en la toma de decisiones. Una transparencia que ayude al equipo SCRUM a maximizar el valor del incremento del sprint y minimizar los riesgos. Si los artefactos no están los suficientemente sólidos, claros o transparentes el equipo no tomara las mejores decisiones (Ramos Vega, 2017).

Product Backlog

El Product Backlog se encuentran todos los requisitos o tareas que sé hacer en el producto este puede contener: historias de usuario, requerimientos, dependencias o bugs (Roche, s/f). El product owner es el responsable de mantener el product backlog ordenado, este debe priorizar las tareas que agreguen un mayor valor al producto. El Product Backlog es un artefacto vivo, puede cambiar en cualquier momento agregando nuevas tareas o cambiar otras, por lo que no se le considera completo.

Sprint Backlog

En el Sprint Backlog encontramos los elementos del product backlog que se trabajara en el sprint. Antes mover estos elementos, los elementos deben cumplir con la definición de listo que el Equipo Scrum acordó. Este artefacto ayudara al equipo de desarrollo a alcanzar el objetivo del sprint. Ayudara a visualizar el trabajo del todo el sprint.

Incremento

El incremento es resultado del Sprint, el incremento debe tener todas las tareas, historias de usuario, soluciones de bugs y cualquier cosa que se haya desarrollado durante el sprint y que será demostrado al usuario final. El incremento debe ser

utilizable. Cada vez que un incremento es entregado cada vez esta más cerca del producto final.

Definición de Ready

Cuando una tarea se considera lista significa que ya puede hacer abordada en el próximo sprint. Para que un elemento se considere listo, los requerimientos debe estar lo suficientemente claros, debe de estar claro como funcionara ese elemento, que recursos utilizará por ejemplo que base de datos utilizara o que UI framework se implementara.

Definición de Done

La definición de Done indica que una tarea ya está proporcionando valor al incremento. Pero todo el equipo SCRUM debe saber que significa. La definición puede ser flexible. La definición puede cambiar de un equipo a otro. Por ejemplo, para el equipo considera que la tarea esta Done cuando pasa todas pruebas, para el equipo B cuando la tarea se incorpora al ambiente de producción del incremento.

3.1.3. KANBAN

Es un método de gestión de trabajo creado por Toyota Production System a finales de los años 40 (Kanbanize, s/f). Kanban viene del japonés que significa tarjeta con signos o seña visual. Es un método de flujo continuo a diferencia de SCRUM que el trabajo se divide en intervalos de tiempo(sprints), en Kanban el flujo no se detiene hasta que no se encuentren tareas en el muro.

3.1.3.1. Muro Kanban

El método Kanban se basa en hacer visible lo que de otro modo es trabajo del conocimiento intangible, para asegurar que el servicio funciona con la cantidad de trabajo correcto (David J Anderson & Andy Carmichael, 2016). Esto se porque en el muro se visualiza el trabajo, como se está desarrollando o donde está el cuello de tarea.

El muro esta conformado por cartas que representan una tarea del proyecto que se está trabajando y por columnas que representan el estado de una tarea. Por

ejemplo, la crear una tarea llamada crear página de Login y se coloca en la columna que representa el desarrollo en ese momento esa tarea sé está desarrollando luego que termina esta etapa la tarea pasa a QA entonces la carta se deba pasa a la columna de QA.

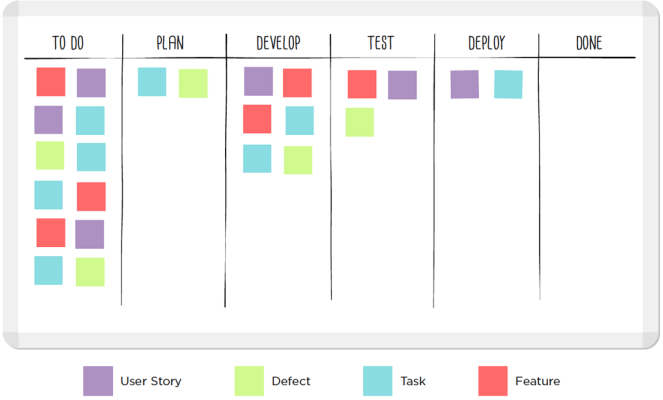


Ilustración 6 Ejemplo de un Muro Kanban

Fuente: (“The Kanban Board”, 2015)

El muro Kanban es de muy útil para identificar los cuellos de botella y tomar acciones para mejorar esa etapa y todo el proceso de desarrollo. Estas cartas tienen una descripción y también tienen periodos de tiempo para ser completadas. Primero tenemos el tiempo que tomara la tarea en pasar a la siguiente etapa y este tiene el nombre de Tiempo de Ejecución del Sistema. Luego tenemos el tiempo de entrega donde es el tiempo que el usuario y el equipo espera que dicha tarea este completa (David J Anderson & Andy Carmichael, 2016).

3.1.4. KATA

Kata es un entrenamiento de karate donde el usuario practica una serie de movimientos en solitario o en parejas. En el desarrollo de software es un ejercicio para los desarrolladores para mejorar sus habilidades a través de la práctica y la repetición (Novoseltseva, 2018).

El objetivo de kata en el desarrollo de software y en el karate es mantener las buenas practicas y mejorar en ellas. Al paso del tiempo que un estudiante entrena una tarea usando katas dicha tarea se vuelve natural para el estudiante hasta el punto de que el estudiante lo hace de manera automática. Aunque los katas de

código ayudan a realizar una tarea de manera automática el desarrollador también tiene que entender en como funciona dicha tarea.

El resultado final de un kata es que el desarrollador pueda poner en práctica los aprendido en diferentes proyectos con diferentes lenguajes sin problemas y con la total seguridad de lo que está haciendo.

3.2. LENGUAJES Y FRAMEWORKS

3.2.1. REACTJS

ReactJS es un framework para aplicaciones web programado en JavaScript. Fue creado por Jordan Walke, un ingeniero de Facebook. Fue ideada para la creación de interfaces de usuario en el desarrollo de single-page application o SPA. Así que React es el Vista en el patrón Modelo-Vista-Controlador o MVC.

Las aplicaciones hechas con React tiene un diseño modular, donde cada modulo es llamado componente, con esto viene una ventaja porque se vuelve sencillo de separar. Todo lo visual todo con lo que interactúa el usuario son componentes. Esto facilita que la aplicación sea más escalable y fácil de mantener ya que los errores sucederán en la propia funcionalidad del componente o en la comunicación con los demás(tiThink, 2018). Para que los componentes sean funcionales estos un state o estado, props o parámetros y un ciclo de vida. La principal ventaja de React es poder generar el DOM ("Modelo de Objetos del Documento"), estructura de los elementos que se generan en el navegador web al cargar una página de forma dinámica (tiThink, 2018).

3.2.1.1. Componentes

Los componentes le permiten dividir la IU en piezas independientes y reutilizables, y pensar en cada pieza de forma aislada(Facebook, 2019). Los componentes son las piezas de una aplicación de React. Como se mencionó anteriormente la ventaja de usar componente ayuda a la aplicación a ser escalable. Un componente puede ser un simple botón hasta ser la aplicación entera. Los componentes pueden ser encapsulados en una simple función o ser una clase de JavaScript que pueden ser opcionales recibir propiedades(props).

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}
```

Ilustración 7 Ejemplo de una Class Component

Fuente: (Facebook, s/f)

```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
  return <h1>Hello, Stranger.</h1>;
}
```

Ilustración 8 Ejemplo de una Functional Component

Fuente: (Facebook, s/f)

Tenemos varios tipos de componentes que se clasifican por función, estructura. Primero tenemos los componentes de comportamiento. Estos se diferencian por que usan por ejemplo si usan un estado o no y por su función. Hay 4 tipos de componente de comportamiento:

1. **Stateful Component:** se caracteriza por tener un estado que mantienen en constante actualización y ser una clase en JavaScript que heredan de React.Component.
2. **Stateless Component:** estos componentes son más simples porque no tienen un estado. Este tipo se encuentran encapsulados en funciones de flecha.
3. **Pure Component:** es parecido al Stateful Component las diferencias son que trabajan con un estado y estos heredan de React.PureComponent. Es un componente visual.
4. **Componentes de Orden Superior o HOC:** Son componentes que encapsulan otros componentes que son similares en unas funciones, pero defieren en otra. Por ejemplo, tenemos los componentes Empleado A y Empleado B donde ambos tienen el mismo estado que contiene el nombre,

ID, teléfono, pero ambos tienen diferentes cálculos de salario donde uno gana por hora y otro por tareas realizadas entonces hacemos diferentes cálculos para cada tipo y mantenemos el mismo código para los otros campos. Esto es similar al polimorfismo.

Luego tenemos los componentes estructurales, estos no se diferencian por sus elementos sino por el uso que se les da. El objetivo de esta categoría es para organizar la aplicación para que un futuro los desarrolladores que continúen se le haga sencillo entender lo que se realizó. Esta categorización no es un estándar de React si no de la comunidad y permite definir una arquitectura en nuestras aplicaciones (Vega, 2017) . Hay dos tipos:

Componentes Visuales: este tipo se enfoca en el renderizado de la interfaz de usuario, está compuesto por elementos estéticos.

Componentes Contenedor: este tipo se encarga de la parte funcional de la aplicación, son contenedores porque contienen componentes visuales u otros componentes contenedores. Estos solo deben estar enfocados en la lógica de la aplicación y de hacer las llamadas a una API o a una base datos y entregar los resultados a los componentes que contienen.

Se puede ver un ejemplo de estos en la Ilustración 9 donde el componente App es el componente contenedor que se encargan de enviar la información al componente Welcome que tiene la tarea de mostrar la información que recibió.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Ilustración 9 Componente Contenedor y Componente Visual

Fuente: (Facebook, s/f)

3.2.1.2. Estado

El estado o state es una estructura de datos tienen un componente de manera opcional. El estado se encarga de almacenar los datos que manejan en un componente de manera local y privada. Privada porque un estado solo puede pertenecer a un componente, un componente externo no puede modificar el estado de otro directamente, aunque es posible hacer por medio de funciones enviadas por medio de las props. El uso de un estado modifica el comportamiento de un componente, extiende las funciones de este. Tenemos pocos lugares para usar el estado pero existen convenios que guiar para el manejo de este(Mukherjee, s/f).

El estado debe inicializarse al principio del ciclo de vida un componente, esto sucede en el constructor del Stateful Component. Esto nos ayuda a identificar que tipos tiene el estado.

El estado no debes actualizado explícitamente. React utiliza un objeto observable como el estado para detectar que cambios se realizan en el estado y ayuda al componente a comportarse sin sorpresas (Mukherjee, s/f).

```
this.state.attribute = "new-value";
```

Ilustración 10 Mala actualización del estado

```
this.setState({attribute: "new-value"});
```

Ilustración 11 Correcta actualización del estado

Fuentes: (Mukherjee, s/f)

Las actualizaciones del estado son independientes, esto dice que no es necesario enviar todo el objeto State a la función setState, para actualizar solo basta enviar los atributos que se quieren cambiar.

3.2.1.3. Props

Las props o propiedades de un componente se utilizan para que haya una comunicación entre un componente padre a sus componentes hijos. Siempre en esta dirección no es posible que un componente hijo le pase sus props al componente padre, haciendo que este tipo de comunicación entre componentes de manera unidireccional. Los props son inmutables, no es posible cambiarlo y React no ofrece soluciones para esto.

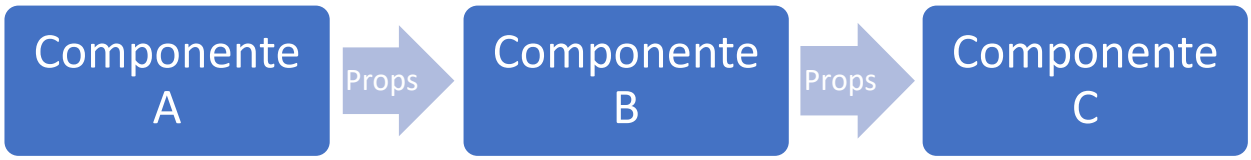


Ilustración 12 Flujo de las props a través de varios componentes

Fuente: Propia

3.2.1.4. Ciclo de Vida

Todos los componentes de React tienen un ciclo de vida. El nacimiento de un componente en React es cuando carga en la página web, la etapa de adultez o crecimiento es cuando el usuario está usando el componente o cuando se está mostrando y la muerte o cierre del componente.

Nacimiento o Montaje

El ciclo de vida de un componente de React comienza con la ejecución del render (Polanco & Pedersen, s/f). En esta fase empieza la pre configuración del componente donde también el estado y los props ya están definidos. El componente y todos sus componentes hijo ya ha sido montados en el DOM. Esta fase solo se ejecuta una vez por cada componente.

Crecimiento o Actualización

En esta fase los componentes pueden perfectamente actualizar sus estados y pueden recibir nuevas props de parte de su componente padre. En esta fase ocurre las interacciones entre el componente y el usuario. Esta fase se repite muchas veces y toma el mayor tiempo del ciclo de vida.

Muerte o Desmontaje

Esta es la fase final del ciclo de vida, es el momento que se remueve un componente del DOM, cambiando de vista, ocultando el componente, actualizando página. Simplemente irse a una página web esto en general pasa una vez en el ciclo de vida.

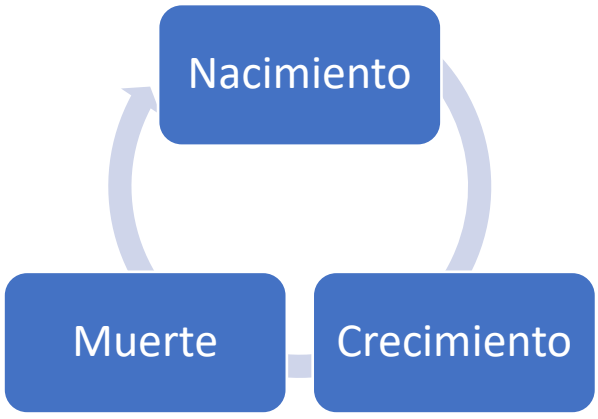


Ilustración 13 Ciclo de vida de un componente

Fuente: Propia

3.2.2. REASONML

ReasonML, es una extensión de sintaxis para el lenguaje de programación OCaml creado por Jordan Walke. OCaml es un lenguaje de programación funcional también es imperativo y orientado a objetos. Es fuertemente tipado, ha sido

probado durante los años y según la ilustración 14 OCaml es rápido comparado con otros lenguajes.

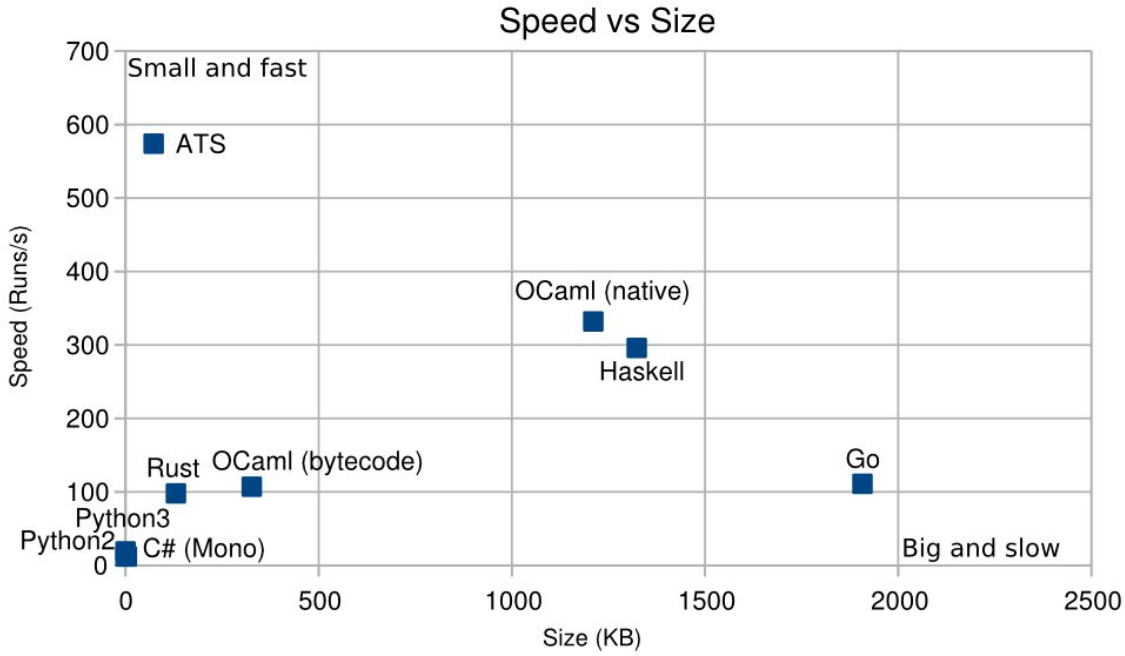


Ilustración 14 OCaml contra otros lenguajes

Fuente: (Leonard, 2014)

ReasonML ha sido diseñado con el objetivo que se parezca a JavaScript para que los desarrolladores sientan que están en un ambiente familiar. Por ejemplo, la sintaxis de los mapas de ReasonML es parecida a los mapas de JavaScript. "ReasonML aprovecha las ventajas de OCaml y JavaScript y puede compilar en ambos lenguajes"(Riva, 2018). ReasonML puede compilar a JavaScript gracias a BuckleScript.

Ilustración 15 Entrada y Salida del BuckleScript

Fuente: Propia

3.2.2.1. BuckleScript

BuckleScript no es un nuevo lenguaje, es un interprete que toma código escrito en OCaml y lo convierte en un código limpio y eficiente en JavaScript. BuckleScript ayuda a superar las debilidades que presenta JavaScript. Gracias al sistema de tipado de ReasonML siempre se sabe que tipos se están operando así evitando errores durante la ejecución. No es necesario escribir el tipo cuando se crea una variable o una función, BuckleScript infiere el tipo, en el caso de una función adopta el tipo de la última expresión. En la ilustración 16 se aprecia que la función ya infiere el tipo ReasonReact.reactElement.

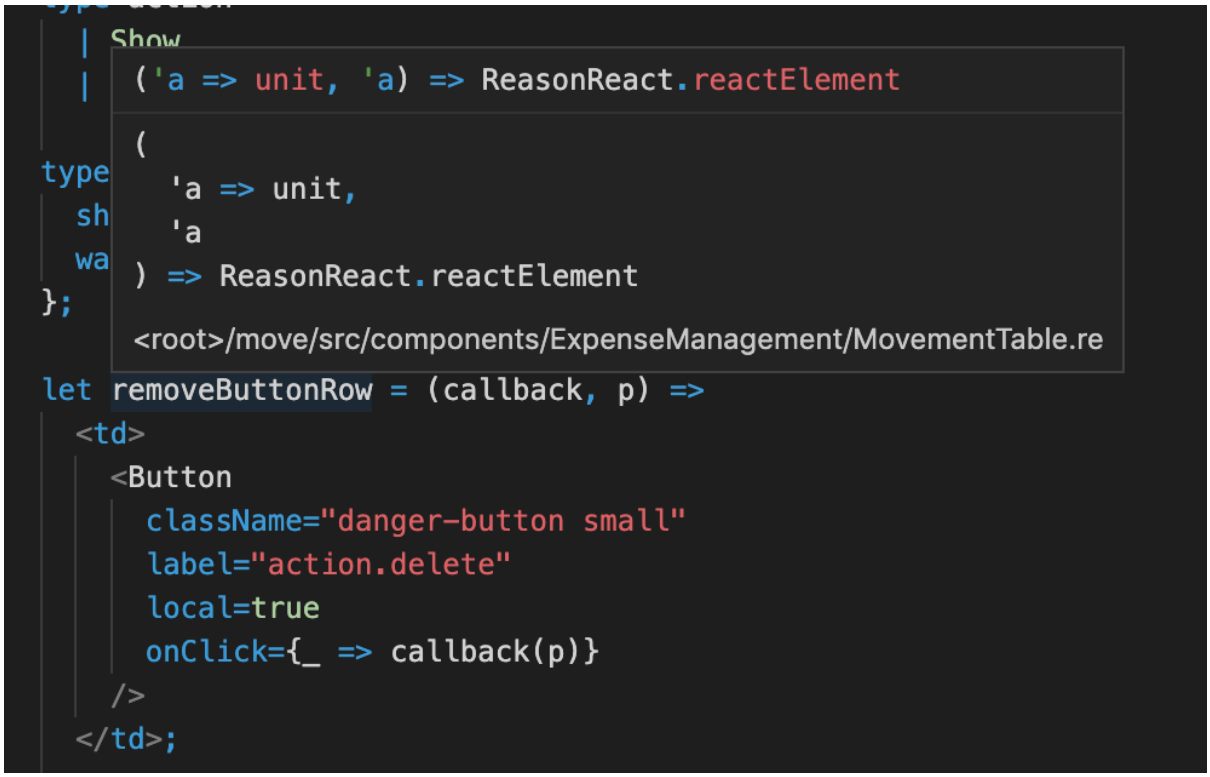


Ilustración 16 Inferencia de tipos de ReasonML

Fuente: Propia

3.2.3. REASONREACT

"ReasonReact es una biblioteca de Reason que se une a ReactJS y proporciona una forma más simple y segura de construir componentes de ReactJS. Al igual que ReactJS es solo JavaScript, ReasonReact es solo Reason"(Rafatpanah & D'mello, 2019).

ReasonReact presenta varias ventajas, Reason provee su sistema de tipado a React así evitando errores en ejecución. Es muy fácil reutilizar código existente de ReactJS en ReasonReact solo se necesita de unos cambios gracias a la similitud de Reason con JavaScript. Ver ilustración 17 y 18 para guía.

```
<Paper square>
  <Tabs
    value={value}
    indicatorColor="primary"
    textColor="primary"
    onChange={handleChange}
    aria-label="disabled tabs example"
  >
    <Tab label="Active" />
    <Tab label="Disabled" disabled />
    <Tab label="Active" />
  </Tabs>
</Paper>
```

Ilustración 17 Material UI en ReasonReact

Fuente: Propia

```
MaterialUi.(
  <Paper square=true>
    <Tabs
      value indicatorColor=`Primary textColor=`Primary onChange=handleChange>
      <Tab label={"Active"->React.string} />
      <Tab label={"Disabled"->React.string} disabled=true />
      <Tab label={"Active"->React.string} />
    </Tabs>
  </Paper>
);
```

Ilustración 18 Material UI en ReactJS

Fuente: Propia

React no estaba pensado para usar JavaScript, el objetivo era que se usara con Reason. Porque Reason provee las características que no provee JavaScript a React por ejemplo el sistema de tipado, Reason es inmutable y es provee las ventajas de un lenguaje funcional.

Solo por convenio ReactJS debe ser inmutable, aunque se sabe JavaScript no posee esta característica, es posible modificar el estado y props de un componente y esto hace ya impredecible. Con Reason esto no es posible, para hacer cambios de estado se debe usar funciones diseñada para ello y además estas funciones no se pueden llamar así por así solo pueden llamarse en ciertas partes.

Aunque React modifica el DOM, se considera programación funcional. No es estrictamente funcional, pero es fácil razonar porque el DOM es su único resultado. Los conceptos de FP pueden ayudarnos a comprender React (Normand, 2019).

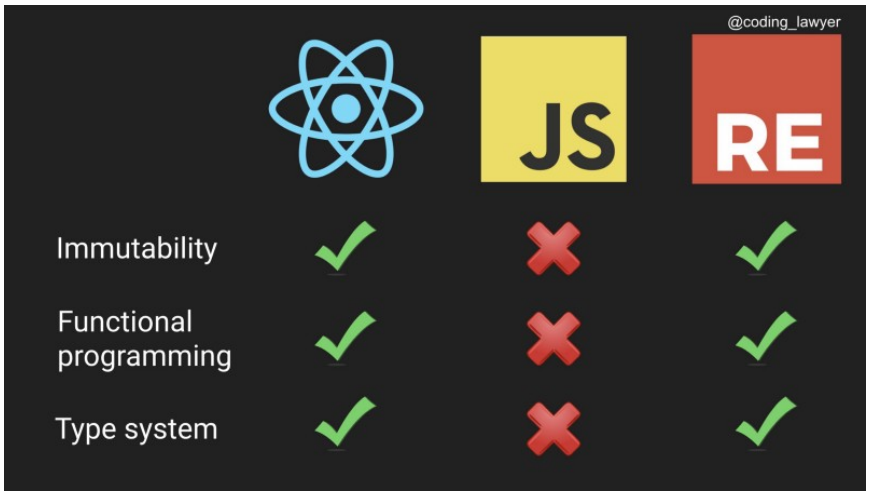


Ilustración 19 Comparativa entre React, JS y ReasonML

Fuente: (David Kopal, 2018)

ReasonReact tiene un buen catalogo de librerías y recibe soporte por parte de la comunidad. La mayoría de las librerías vienen librerías ya existen escritas JavaScript para alcanzar esto se deben crea bindings. Un binding es la adaptación de una librería, biblioteca o paquete que va a hacer usada en un lenguaje distinto al que fue creado. Gracias a la similitud de Reason y React es muy simple de crearlos.

3.2.4. POUCHDB

PouchDB es una base de datos no SQL escrita en JavaScript y una implementación de JavaScript que nos permite hacer operaciones sin la necesidad de tener conexión permanente con la base de datos porque todos los datos que este manejando se están guardando en el browser, en el momento que recuperemos

la conexión la base de datos comenzara a sincronizarse así guardando todos los cambios que hemos hecho. Esta es una gran ventaja porque esta sección se vuelve tolerante a errores, el usuario de la base de datos puede seguir trabajando con ella con normalidad.

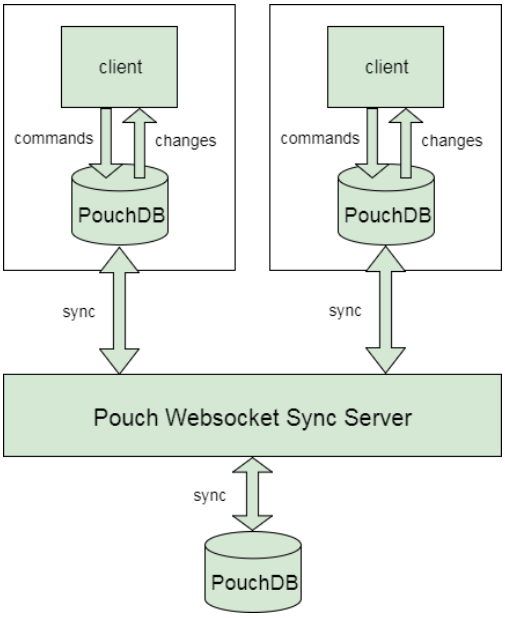


Ilustración 20 Funcionamiento de Sincronización de PouchDB

Fuente: (Javatpoint, s/f)

PouchDB puede trabajar en varios browsers, por ejemplo: Chrome, Safari, Firefox, Internet Explorer incluso en dispositivos móviles. PouchDB es muy rápido esto viene porque todas las operaciones se ejecutan en el browser.

3.2.5. CSS

CSS (Cascading Style Sheets) es un lenguaje de diseño gráfico creado para controlar la presentación de los documentos electrónicos escritos en HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas (Perez, s/f).

Los diseños para una pagina web a veces son limitados por los browsers o navegadores web, porque algunas características o algunas propiedades funcionan perfectamente en unos, pero unos no dan el resultado que esperaremos o simplemente no funcionan. Está es la razón se debe pensar a que browser será creado.

Aunque los navegadores parece que funcionan de la misma forma, internamente son muy diferentes. La parte del navegador que se encarga de interpretar el código HTML y CSS para mostrar las páginas se denomina motor.

Tabla 1 Motores gráficos de los navegadores Web

NAVEGADOR	MOTOR
Internet Explorer	Trident
Firefox	Gecko
Safari	Webkit
Opera	Presto
Google Chrome	Webkit

Fuente: (Pedro Ventura, 2011)

3.2.6. MATERIAL UI

Material UI es un CSS framework para la normativa de diseño Material Design creada por Google. La librería fue escrita en JavaScript con el objetivo que sea implementada en ReactJS. Material UI tiene casi todos los componentes de Material Design a excepción de los Date Pickers, para que se puedan usar se tiene que instalar la librería Material UI Pickers.

3.2.6.1. Material Design

Material Design fue creada por Matías Duarte y fue lanzada por Google en el año 2015. Material fue integrado como UI los sistemas operativos Android a partir de la versión Lollipop y fue implementado en las aplicaciones de Google como Calendar, Drive, YouTube y Gmail. También se expandió a los servicios web de Google. En la actualidad Material Design se encuentra en muchos dispositivos como televisores, tabletas, relojes inteligentes y también en dispositivos de Apple. Para un buen uso de este diseño se tiene que elegir los componentes perfectos, una buena combinación de colores y fuente que se usara. Lo más importante es

no cargar la pantalla con componentes ya que siempre hacer diseños lo más simple posible.

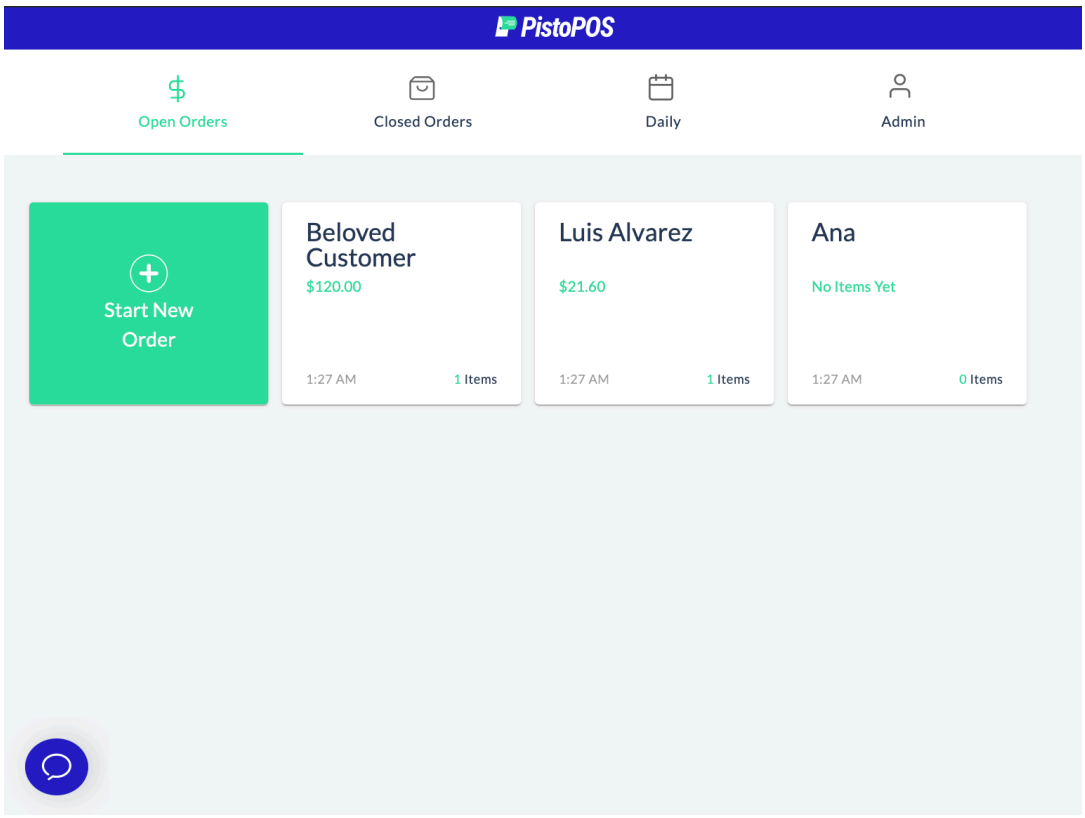


Ilustración 21 Ejemplo de una página usando Material UI

Fuente: Propia

3.3. HERRAMIENTAS

3.3.1. GIT

Git es un sistema de control de versiones creado por Linus Torvards en el año del 2005. Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones especificas más adelante (Chacon & Straub, 2014). El uso de Git no es exclusivo para mantener versiones de código también se puede usar en el ámbito de diseño gráfico si queremos guardar versiones de diseños por ejemplo el diseño de una pagina web, una simple imagen o video incluso podemos usar Git para guardar versiones de un documento de Word.

3.3.1.1. Funcionamiento

Cuando un cambio se produce en un o varios archivos, Git crea copias de todos los archivos del repositorio, pero solo de los archivos que se detectaron cambios y crea referencias de los archivos que no cambiaron. Por ejemplo, en la ilustración 22 se tiene la versión 1 con tres nuevos archivos luego se efectuó un cambio en los archivos A y C entonces se crea una copia para esos archivos y se crea una referencia del archivo B.

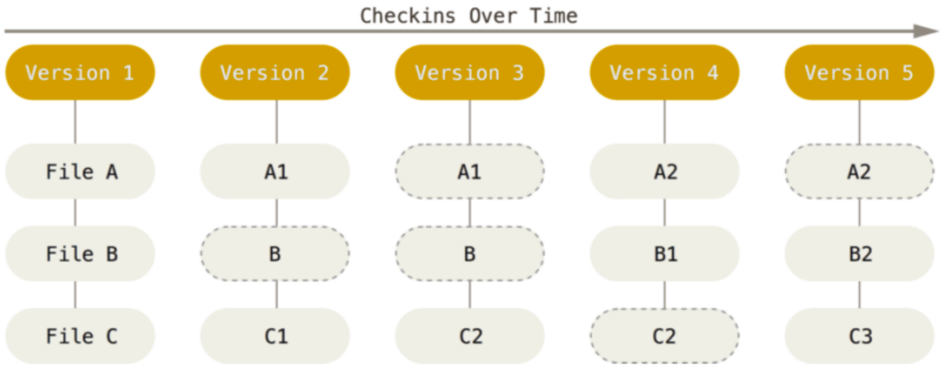


Ilustración 22 Funcionamiento de Git

Fuente: (Chacon & Straub, 2014)

Cabe destacar que gran mayoría las operaciones de Git ocurren localmente no es necesario tener conexión a internet haciendo que Git sea rápido. Necesitaremos tener conexión cuando queremos guardar o cargar los cambios en un servidor local.

3.3.2. GITLAB

Es un sistema manejador de repositorios que emplean Git. Gitlab está escrito en Ruby. Fue publicado en GitHub en octubre del 2011 a partir de esa fecha ha estado tomando aceptación por parte de la comunidad hasta convertirse en la útil herramienta de lo que es hoy.

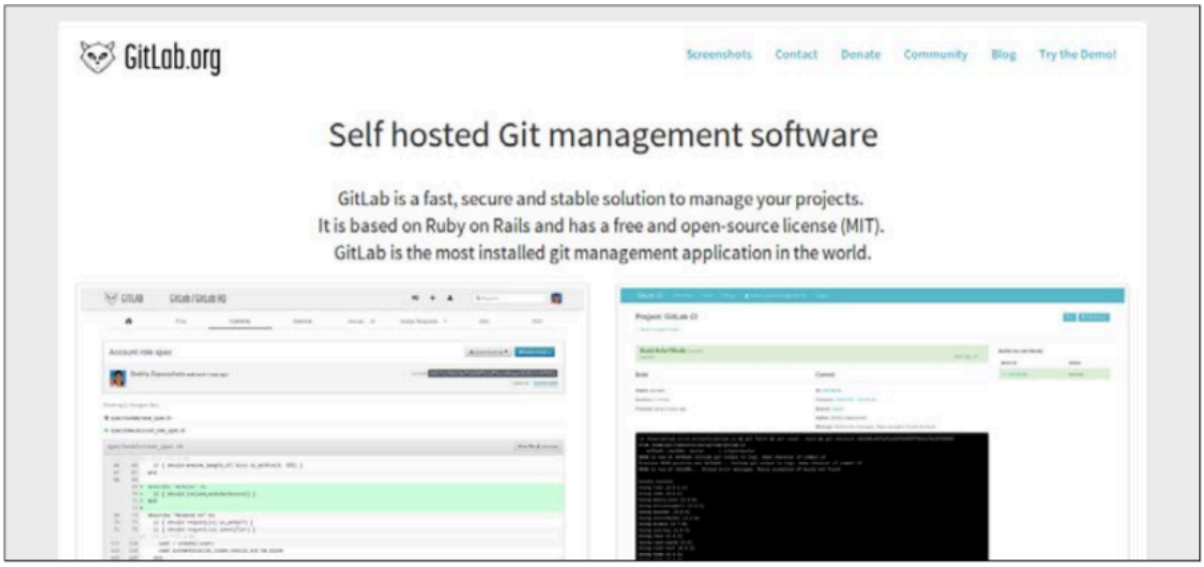


Ilustración 23 Vieja interfaz de Gitlab

Fuente: (Hethey, 2013)

Git posee una interfaz web muy amigable para el usuario y nos brinda la posibilidad de manejar permisos de los repositorios. Nos ofrece hacer un seguimiento de los cambios que es perfecto para encontrar el punto donde un proyecto comenzó a reportar errores.

La interfaz es muy similar a GitHub, gracias a eso los desarrolladores sientes que están en un ambiente familiar, nos ofrece la posibilidad de crear branches, merge request, ver los cambios que han sucedido en un archivo incluso es posible editarlos en la página web.

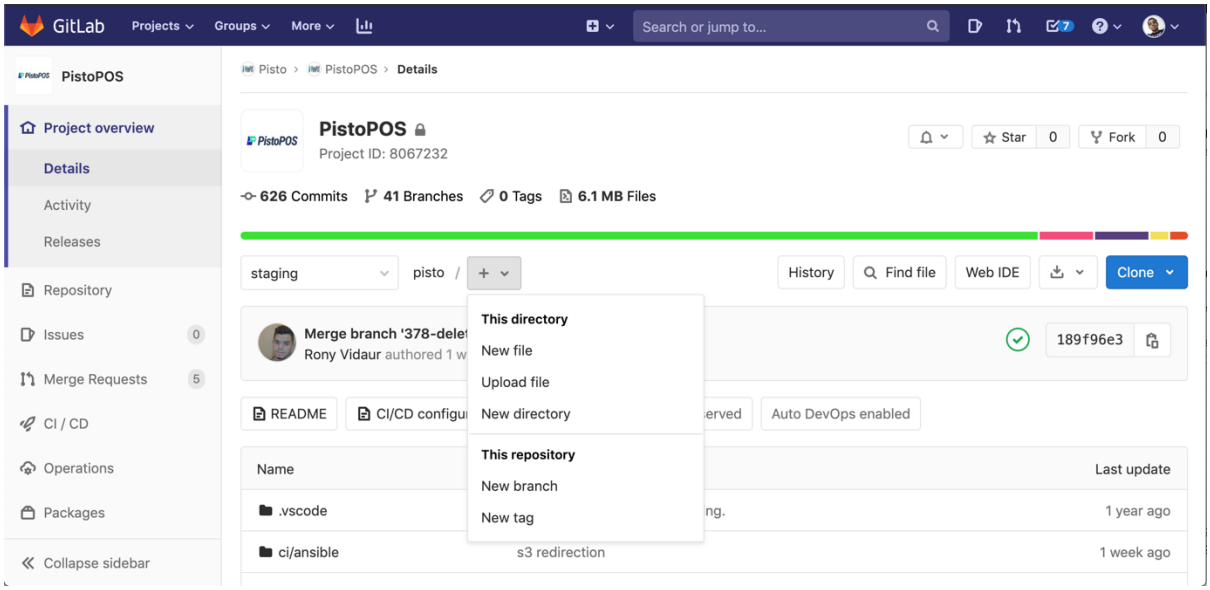


Ilustración 24 Interfaz actual de Gitlab

Fuente: Propia

Existen varias posibilidades para limitar u otorgar accesos a funciones por ejemplo quien puede subir cambios al repositorio, quienes pueden acceder a una branch en especifico. Gitlab ofrece roles bien definidos desde un invitado hasta el maestro del repositorio (Hethey, 2013).

3.3.3. SLACK

Slack es un servicio de mensajería en tiempo real que permite la colaboración entre miembro de un equipo o una empresa. Este servicio incluye todos los medios de comunicación como mensajería, llamadas, videoconferencias y también se pueden compartir archivos. Slack es la alternativa al uso del correo electrónico.

En Slack los grupos se conocen como canales donde puede ser públicos o privados, cada equipo de trabajo dentro de una organización puede tener su propio canal, por ejemplo: un canal para los desarrolladores, un canal para los diseñadores o un canal general.

Es posible tener varias aplicaciones conectadas a nuestro perfil de Slack para agregar nuevas funcionalidades a la aplicación por ejemplo Google Drive para compartir archivos o Calendar que envía notificaciones de eventos futuros.

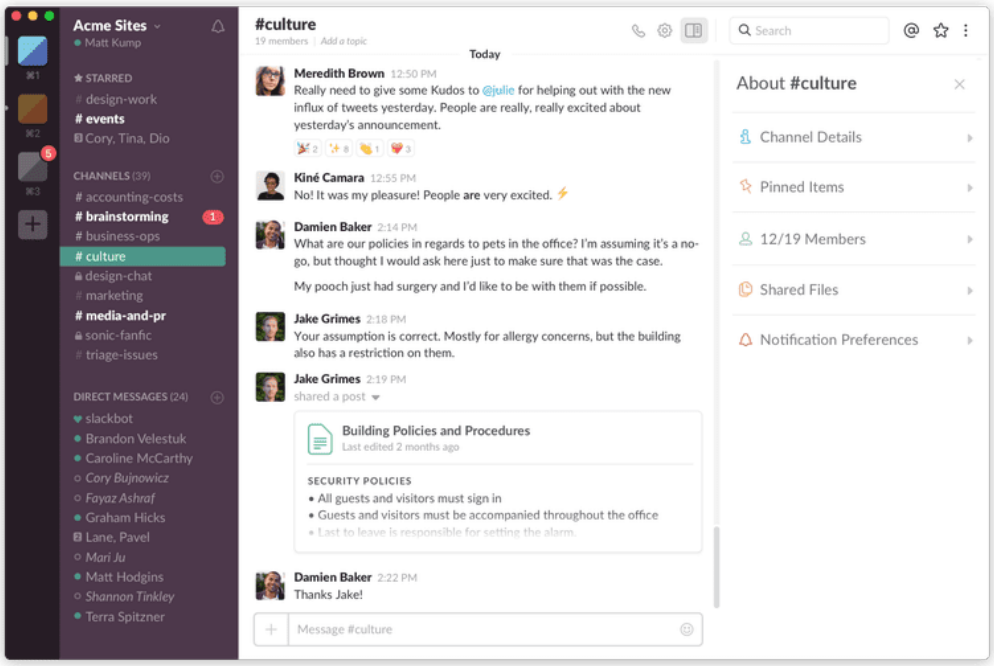


Ilustración 25 Interfaz de Usuario

Cada usuario puede tener permisos y restricciones. Los propietarios del espacio de trabajo tienen todo el control, este mismo puede asignar usuarios administradores para que administren el canal, agregar o remover miembros, después están los miembros estos pueden enviar y recibir mensajes y hacer llamadas, por último, tenemos los invitados quienes solo pueden ver canales específicos.

3.3.4. GOOGLE MEET

Google Meet es una aplicación de videoconferencias hecho por la compañía Google para ambientes profesionales. Este puede se encuentran varias plataformas como IOS, Android y plataformas web.

Esta herramienta pertenece a la suite de herramientas GSuite que ofrece Google para los entornos profesionales. Aunque también esta Hangouts y Meet se parece mucho a este la diferencia es que para entrar una reunión de Meet debe de tener el ID de la reunión.

3.3.5. TRELLO

Es una aplicación para la organización de tareas. Es ideal para la coordinación de equipos y perfecta para la metodología Kanban (Parra, 2016).

Trello nos permite crear varios tableros para proyectos diferentes por ejemplo un tablero para desarrollos un proyecto, uno para el diseño de una página o para organizar nuestras tareas cotidianas.

Tiene un funcionamiento similar a un muro Kanban, donde un tablero contiene listas que representan las etapas de un proyecto y el usuario puede mover las cartas de una etapa a otra. Las cartas pueden contener información acerca de una tarea, se pueden adjuntar fotos o videos. Los miembros que están subscritos a un tablero pueden hacer comentarios acerca de una carta.

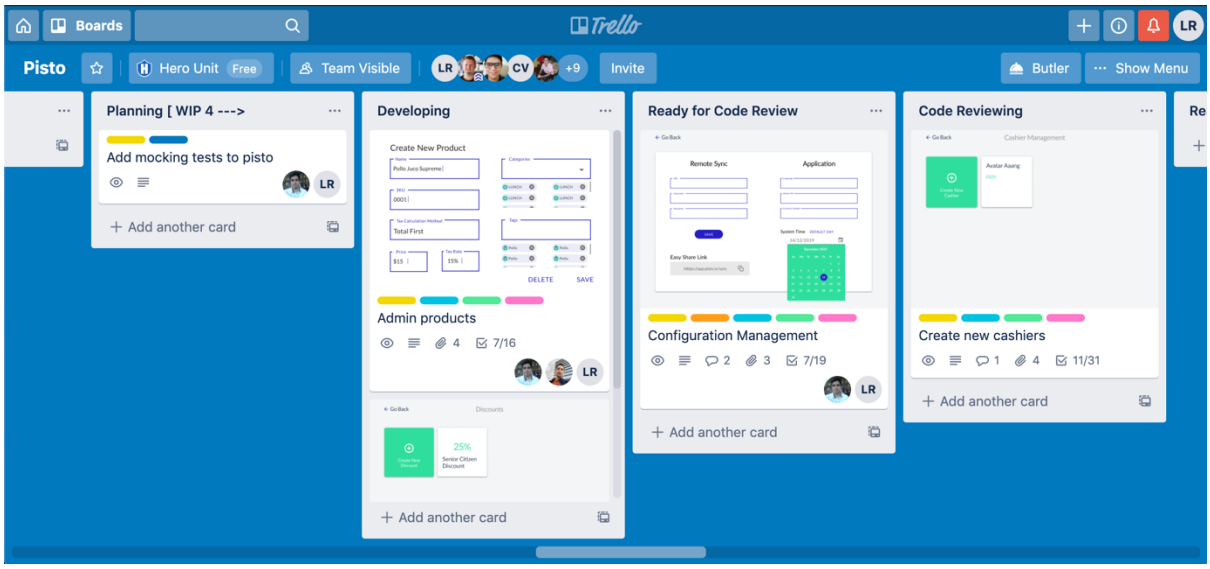


Ilustración 26 Interfaz de Usuario de Trello

Fuente: Propia

3.3.6. LIVESHARE

LiveShare es un complemento de VSCode que es útil para hacer pair programming sin estar presente físicamente en un mismo espacio. LiveShare permite al equipo de trabajar en el mismo código en tiempo real, no es necesario configurar un entorno o usar otras herramientas.

Para comenzar el pair programming se debe crear una sesión de colaboración y enviar la invitación a los que participaran en la sesión. Cuando los miembros del equipo se unen inmediatamente tiene acceso al código y estos pueden modificarlo si el anfitrión lo permitió. Además, el anfitrión puede compartir varios servidores de desarrollo y varias instancias de la consola.

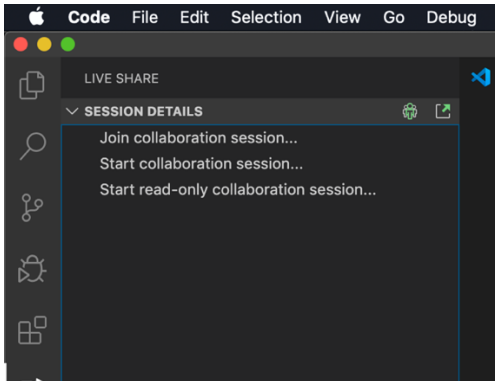


Ilustración 27 Creación de sesiones

Fuente: Propia

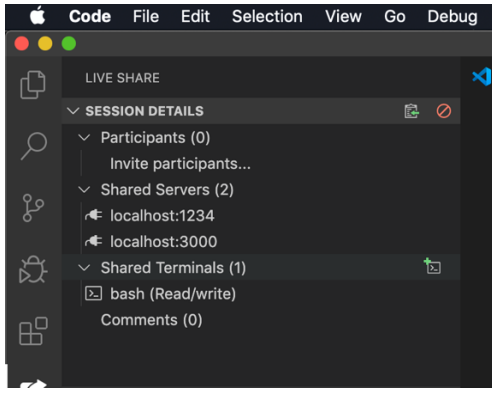


Ilustración 28 Interfaz de LiveShare

Fuente: Propia

IV. DESARROLLO

Durante mi periodo de práctica profesional realizada en la empresa o división de Acklen Avenue llamada HeroUnit. Me desempeñe en el rol de Software Developer (Desarrollador de Software). En el comienzo tome unos cursos en las primeras semanas para conocer los valores de HeroUnit y Acklen Avenue, en como ser un profesional de forma remota. Durante mi periodo de práctica profesional de 22 semanas comenzó el 9 de octubre del 2019 y concluyo el 11 de marzo de 2020.

Estuve involucrado en el desarrollo del Proyecto Interno PistoPOS que consiste en varios módulos.

4.1. DESCRIPCIÓN DEL TRABAJO DESARROLLADO

4.1.1. PISTO

Pisto fue el proyecto en que trabaje de mi práctica profesional. Cuando entre al equipo el proyecto tenia alrededor de dos años de inactividad y al momento que entre el proyecto había sido reanudado el 1 de octubre. El equipo estaba conformado por un Product Owner, un SCRUM Master, tres desarrolladores, dos diseñadores, un QA y dos encargados de DevOps. En el mes de diciembre hubo cambios, un desarrollador fue transferido y ingreso un nuevo SCRUM Master al poco tiempo después un diseñador fue transferido. En el mes de enero un nuevo miembro se unió al equipo y obtuvimos otro miembro de DevOps.

Ilustración 29 Equipo de Desarrollo antes de diciembre

Fuente: Propia

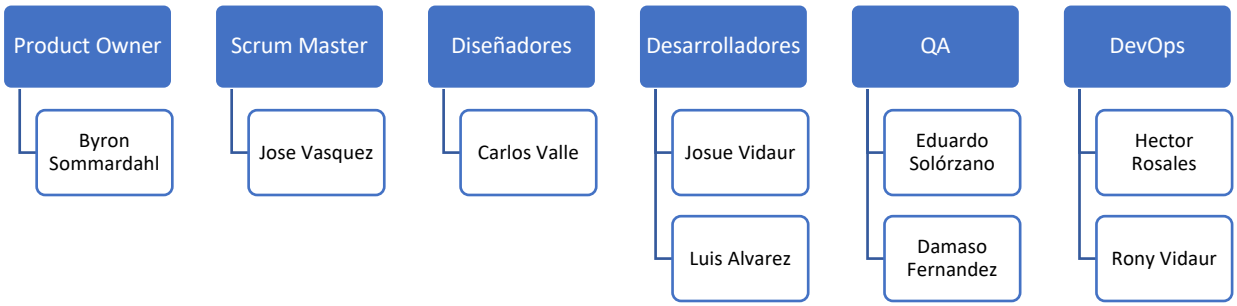


Ilustración 30 Equipo de Desarrollo después de diciembre

Fuente: Propia

4.1.1.1. Definición del Problema

Este proyecto surgió como un proyecto de Byron Sommardahl. Él y su esposa abrieron un Café en la localidad de Santa Ana al sur de Tegucigalpa en el departamento de Francisco Morazán. El Café necesita de un programa que ayude al negocio a tomar las órdenes de los clientes, poder gestionar todos sus recursos y hacer reportes diarios o hacer un reporte entre unas fechas.

4.1.1.2. Descripción

Pisto tiene varios módulos. Primero tenemos a PistoPOS donde los cajeros pueden crear órdenes, administrar productos, categorías, descuentos y administrar cajeros. Luego tenemos a Move donde los usuarios pueden crear productos al igual que Pisto, pero la diferencia es que los productos poseen más campos por ejemplo ingresar el proveedor también se gestionan proveedores, categorías y usuarios.

Ilustración 31 Logo de PistoPOS

Fuente: HeroUnit

PistoPOS y Move comparten la misma base de datos, si hay cambios en la configuración de Pisto por ejemplo cambiar el idioma al Ingles ese cambio de manifiesta también Move. Ambos tienen dos dependencias muy importantes el primero es Cafe Domain donde en el diseño MVC sería el Modelo, este contiene toda la estructura de las entidades, por ejemplo, los productos, cajeros o los descuentos. El segundo es Cafe Data que seria el Controlador en el diseño MVC. Este tiene todos los procedimientos para manipular las entidades de Cafe Domain, es donde está el CRUD de las entidades. FAF(Feed America First) Intake es una versión pequeña de Move donde se puede crear productos y luego generar un código de barra relacionado a ese producto. Después tenemos dos módulos de impresión uno es Cafe Receipt Printer que es usado para imprimir los recibos de las órdenes creadas en PistoPOS. El otro es FAF Intake Printer que es usado para

generar códigos de barra para los productos para exportarlo a PDF o imprimirlo con una impresora ZPL.

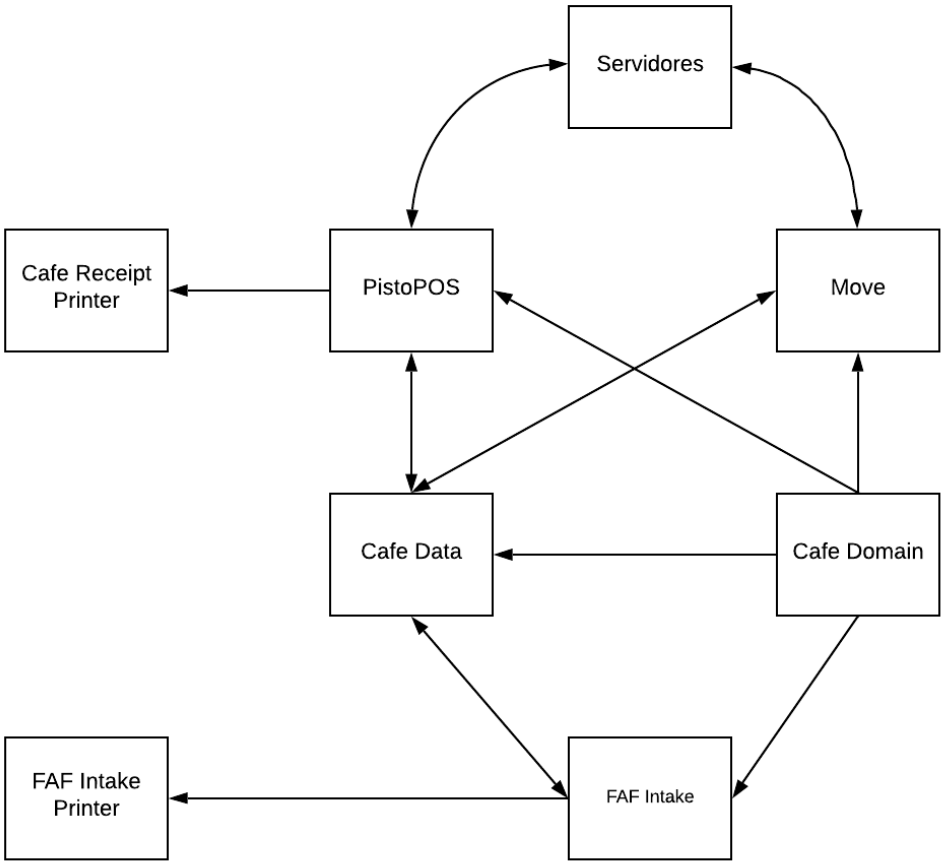


Ilustración 32 Diagrama de los módulos de Pisto

Fuente: Propia

4.1.1.3. Descripción Técnica

A continuación, se presentará una tabla con los detalles técnicos de proyecto o módulos de PistoPOS. Al ser un proyecto usado con React no es necesario de un backend para su funcionamiento.

Tabla 2 Descripción Técnica del Proyecto Pisto

Detalle	Frontend
Plataforma	Navegadores Web
Framework	ReasonReact
UI Framework	Material UI
Sistema Operativo de Desarrollo	MacOS X
Dispositivos Compatibles	Computadoras, Dispositivos Móviles
Motor de Base de Datos	PouchDB
Sistema de Control de Versiones	Git
Alojamiento de Repositorios	Gitlab
Servicio de Host	Heroku, AWS
Editor de Texto	Visual Studio Code
Herramienta de Diseño	Figma

Fuente: Propia

4.1.1.4. Trabajo Realizado

En mi primera semana de la práctica me fui asignado hacer trabajos de pruebas al código para ver en como funcionaba los componentes de los módulos de PistoPOS y Move. Eso ayudo mucho para saber el comportamiento de estos módulos y ver como se relacionaban. Al final de la primera semana fui asignado a trabajar con Boris García para introducirme al lenguaje de programación ReasonML, trabajamos en un error que ocurría en las órdenes cerradas, el usuario podía cambiar la fecha de pago a una fecha anterior a la fecha de creación de la orden. Una validación fue agregada al input de la fecha de pago a para evitar el problema descrito anteriormente.

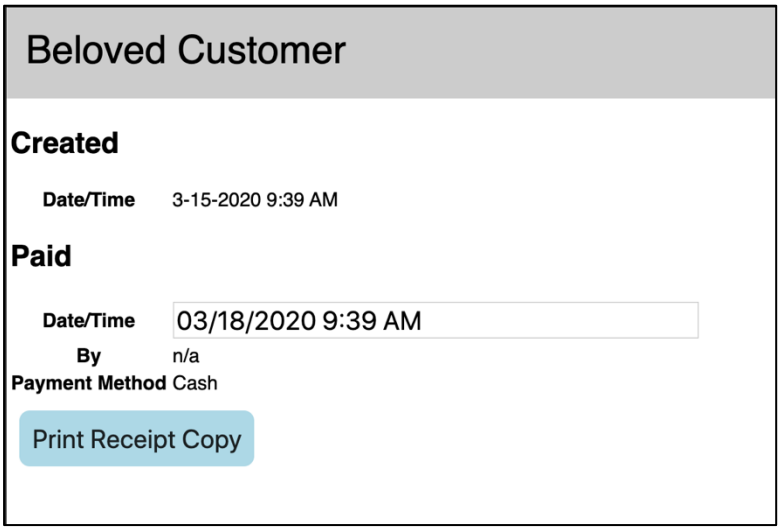


Ilustración 33 Input para el cambio de fecha de pago

Fuente: Propia

Mi siguiente asignación fue en agregar un campo Daily Report llamado onHand para manejar la cantidad de producto que hay en inventario. Para resolver esta tarea se hicieron filtros para conseguir las órdenes que tenían los productos que vendieron en ese día. Se necesita para sumar las cantidades que se vendieron. Después se siguió la misma estrategia para calcular las compras de ese producto. Se hizo un filtro para conseguir las compras que contienes esos productos para luego sumar las cantidades que se vendieron. Ver ilustración 34.

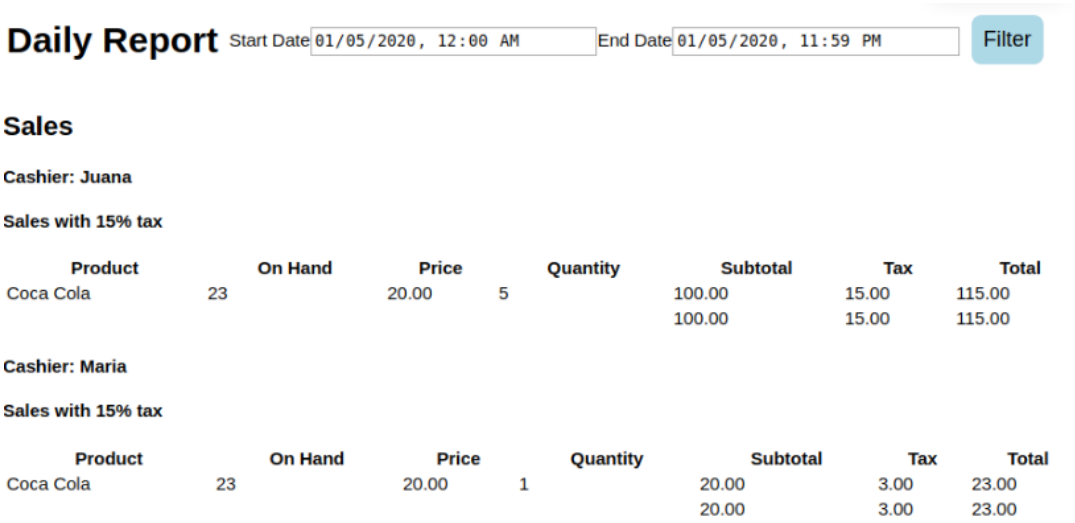


Ilustración 34 Visualización de onHand(inventario)

Fuente: Propia

Durante casi 3 semanas se procedió a actualizar los módulos de Pisto y Move. El motivo de esta actualización fue para solucionar un error que ocurría al momento de compilar un proyecto. De la nada ocurrió un error que decía que el archivo no tenia fin de archivo para seguir con la compilación se debía ingresar saltos de línea o "enters" hasta que la compilación sea un éxito. Esto entorpecía mucho el proyecto porque para cada cambio en el código se tenia que hacer este inadecuado procedimiento. Para saber donde ocurría se actualizaron los módulos por ejemplo se actualizaba el módulo de los órdenes luego el módulo de la administración durante esos procesos se actualizaba las dependencias a la versión más reciente posible. A la más reciente posible porque si actualizamos de una sola vez a la última versión, ya no era compatible con las otras dependencias o se presentaban cambios significativos de código. Al final de las actualizaciones el bug no apareció así que dio por hecho que ya estaba solucionado.

.

Ilustración 35 Error de compilación en Pisto

Fuente: Propia

Luego se tuvo que comprobar el funcionamiento de FAF Printer porque estuvo por mucho tiempo sin actividad y nadie en el equipo no sabia casi nada del funcionamiento de este así que se tuvo que hacer una exploración por el código para saber cuales son las peticiones que recibía el servidor. El principal objetivo era saber si generaba el código de barra por eso tenia que saber que es lo que se debía enviar para cumplir su cometido. Al final resulto que el se generaba correctamente con los datos ingresados.

Ilustración 36 Código de barra generado

Fuente: Propia

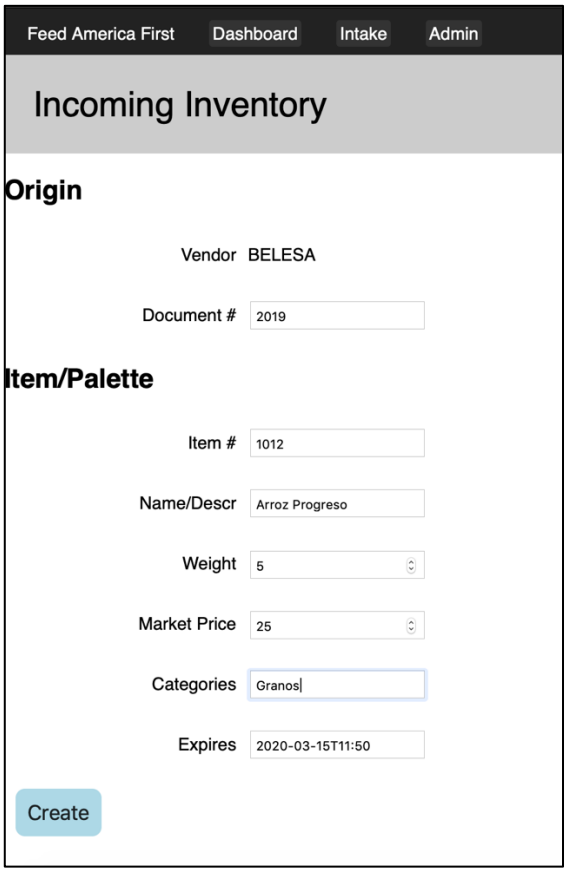


Ilustración 37 Formulario para la creación de productos

Fuente: Propia

Durante el desarrollo se notó que había un problema con los productos y las entidades relacionadas a esta, como las compras y las órdenes. Con el equipo notamos que al borrar un producto si se ingresa a una compra o una orden que posean este producto el programa fallaba. Y este es un problema por que

entorpece los procesos de auditoria que puede tener el negocio. Aunque esto se podía haberse solucionado agregando una validación que revise el producto, pero para eliminar un producto que no se venderá más debería eliminar todo los relacionado con él(órdenes, compras o devoluciones) y no es correcto porque perdemos la transparencia por motivos de posibles auditorias. Lo que se hizo fue agregar un campo que sirva con flag para ocultar los productos en las pantallas de gestión de productos. Con eso es posible ver compras anteriores sin importar si el producto está oculto o no. Esta tarea fue relativamente sencilla solo al momento se debía cambiar la flag(variable bandera) en falso. La función getAll de Productos en Cafe Data fue modificada para que solo retorne los que no están ocultos y agrego otra función para que retorne los productos archivados o no archivados. Aunque getAll no retorna todos los productos en realidad se decidió dejarlo así para mantener la consistencia en el código porque la función getAll es usado en varias partes del Cafe Domain.

La siguiente tarea fue en agrupar las ventas diarias por cajero. Esto fue desarrollado en la pantalla de Daily Report. Para hacer esto debería hacer una agrupación por cajero a las órdenes cerradas. Las órdenes cuando se pagan se guarda el cajero que hizo la orden, así que es fácil extraerlo al contrario del inventario existente. ReasonML nos da la facilidad de hacer la agrupación porque posee una librería que se encarga. ReasonML nos devuelve un mapa donde la key o llave es el condicionante o el campo a agrupar en este caso es el campo cajero. Ya cuando tenemos la agrupación el despliegue de esta en la pantalla es sencillo usando un mapa. Donde un mapa es una especia de foreach donde se le ingresa una lista y después se está creando el HTML para cada venta del cajero. Esto ayudara a visualizar quienes son los cajeros que más venden, también servirá como control.

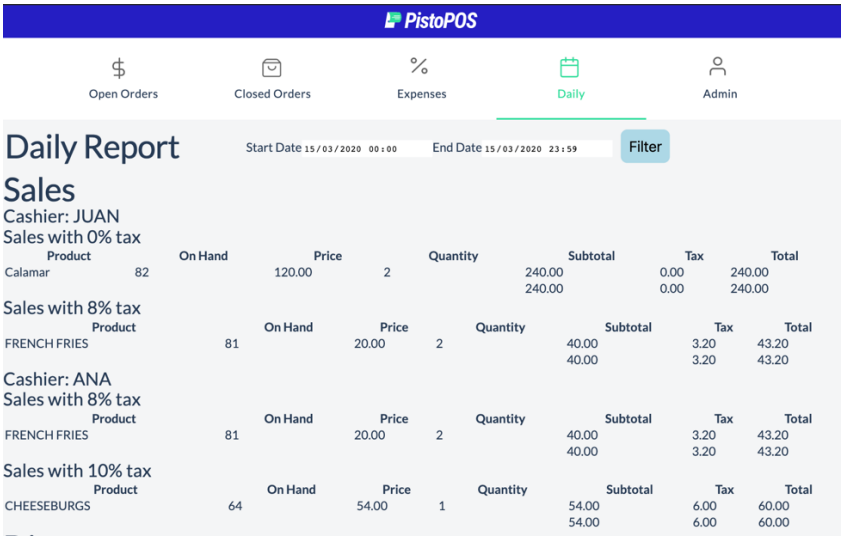


Ilustración 38 Agrupación por cajeros en Daily Report

Fuente: Propia

Luego se le agrego la opción de agregar propinas para el cajero que está atendiendo la orden. Esta opción se presenta al final del proceso de pago de una orden. Para realizar esta tarea se comenzó con agregar un nuevo campo a la entidad Órdenes en el modulo Cafe Domain. Luego se agrego el botón en la pantalla Pay Order para que despliegue de un dialogo que tuvo que ser creado para que el usuario ingrese el monto de la propina y una nota opcional.

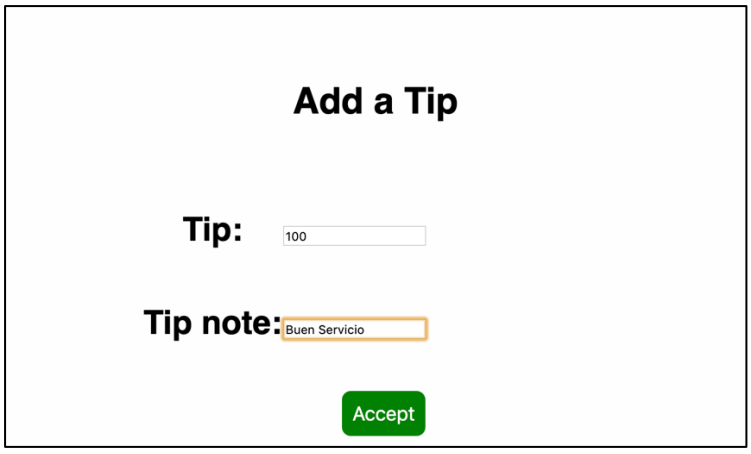


Ilustración 39 Modal para atribuir una propina

Fuente: Propia

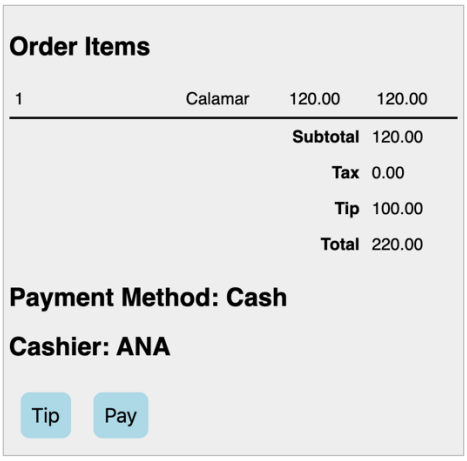


Ilustración 40 Propina en los totales de una venta

Fuente: Propia

Después de esto se agregaron al Daily Report una sección de propinas agrupadas por cajeros. Donde se tuvo que hacer el mismo proceso de agrupar las ventas diarias por cajero y hacer una suma de las propinas que se encuentren en las ordenes.

Tips

Cashier	Total
ANA	100.00
JUAN	20.00

Ilustración 41 Agrupación de las propinas en el Daily Report

Fuente: Propia

Se implementaros los descuentos individuales a los artículos de una orden. Un descuento individual solo puede ser aplicado al articulo que le pertenece y no a toda la orden. Para esto se acordó agregar un nuevo campo a la entidad OrderItem. La relación entre una Order es de una a muchas. Lo que quiere decir que una orden puede terne varios artículos y un articulo a una sola orden. Haciendo esto se hace sencillo saber que artículos tienen descuentos. Para que el usuario pueda ingresar un descuento individual se colocó un pequeño input con la validación donde el numero menor aceptado es 0 y el mayor 100. En la siguiente ilustración 42 se observa un descuento de 50% que solo es aplicable a las papas fritas.

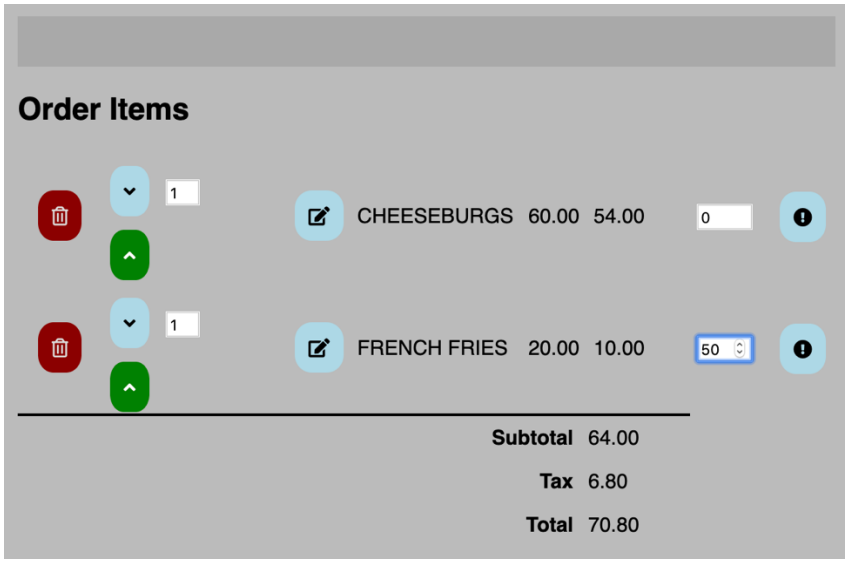


Ilustración 42 Descuentos individuales en el carrito

Fuente: Propia

En el módulo de Move se le agregó la opción de generar un archivo CSV que tome la información que se le está presentando en la pantalla Dashboard. Para realizar esta actividad no se podía usar código de ReasonML, pero el lenguaje permitía ingresar código de JavaScript. La función para generar el archivo debe recibir un JSON debía ser construido. Primero se tenia que ingresar los encabezados del CSV tomando de base los que tiene el dashboard. Luego se debía estar llenando el JSON con los datos para hacer esto se le aprovecho los mapeos que estaban en el render del componente. Luego en la función se extraían los encabezados y concatenan con comas. El siguiente paso era concatenar la data con comas usando un ciclo para esta. Al final se añadió un botón en el header de la pantalla.

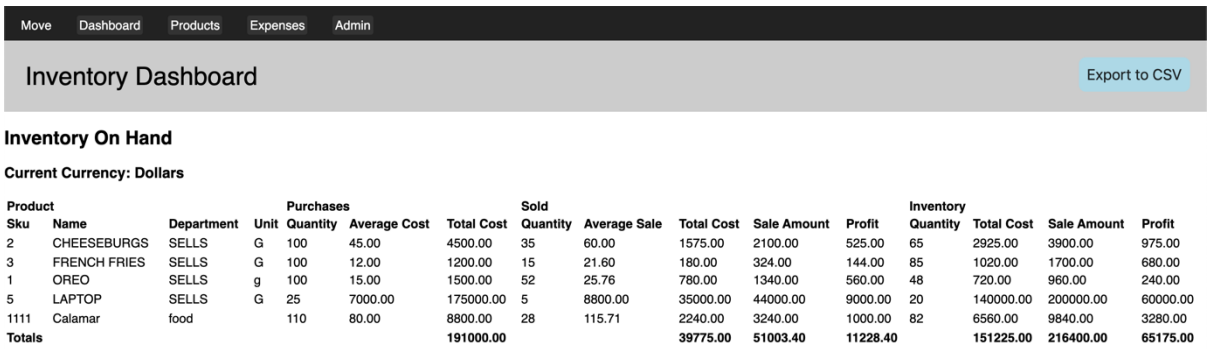


Ilustración 43 Dashboard con el nuevo botón

Fuente: Propia

products

Sku	Name	Department	Unit	Quantity	Average Cost	Total Cost	Sold quantity	Total Sold Cost	Average Sale	Sale Amount	Profit	On Hand	Total Inventory Cost	Inventory Amount	Inventory Profit
2	CHEESEBURGS	SELLS	G	100	45	4500	35	60	1575	2100	525	65	2925	3900	975
3	FRENCH FRIES	SELLS	G	100	12	1200	15	21.6	180	324	144	85	1020	1700	680
1	OREO	SELLS	g	100	15	1500	52	25.76	780	1339.52	559.52	48	720	960	240
5	LAPTOP	SELLS	G	25	7000	175000	5	8800	35000	44000	9000	20	140000	200000	60000
1111	Calamar	food		110	80	8800	28	115.71	2240	3239.88	999.88	82	6560	9840	3280

Ilustración 44 CSV generado

Fuente: Propia

Después del campamento del 10 de noviembre y 11 de noviembre el desarrollo se reanudó. La siguiente tarea corregir un bug en el cálculo de impuesto de las ordenes. En Pisto existe 3 métodos de impuesto, exento donde cuyo producto no se calcula el impuesto, Primeros Totales donde el impuesto ya esta incluido en precio de venta del producto y Primeros Sub Totales donde el impuesto se le añade al precio del producto. Ver la siguiente tabla para guiarse.

Tabla 3 Cálculo de los tres métodos de impuestos

	Artículos	Precio de Venta	Precio Real	Imp. 12%	Total a Pagar
Primero Total	Coca Cola	22	19.36	2.64	22
Primero Subtotal	Oreo	10	10	1.2	11.2
Exento	Baleada	12	12	0	12

Fuente: Propia

Después de buscar en el código se descubrió que las formulas estaban mal escritas. Se procedió a reformular los métodos de cálculos de los artículos que se mostraban en el Daily Report. Después de esa corrección los lugares donde se calcula el impuesto lo están haciendo de manera correcta.

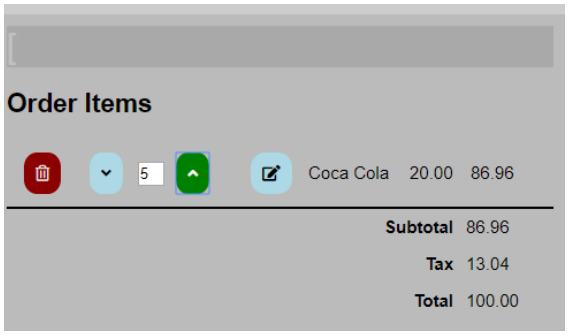
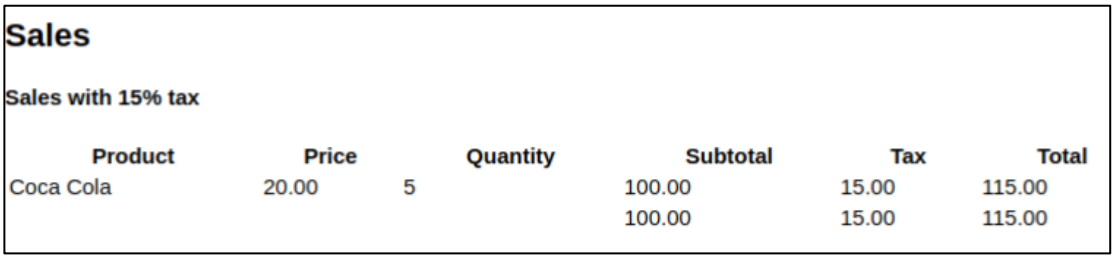


Ilustración 45 Cálculo correcto del impuesto (Carrito de venta)

Fuente: Propia



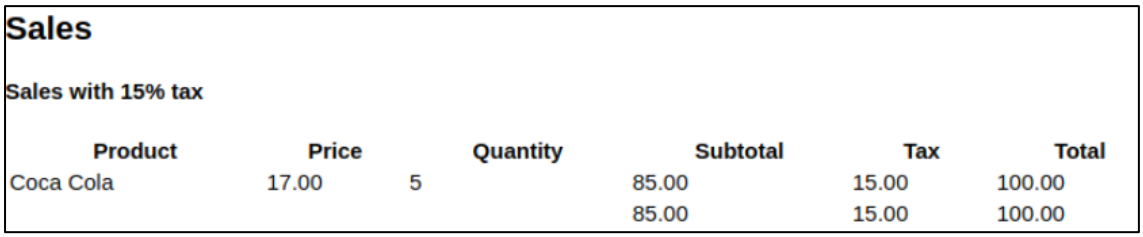
Sales

Sales with 15% tax

Product	Price	Quantity	Subtotal	Tax	Total
Coca Cola	20.00	5	100.00	15.00	115.00
			100.00	15.00	115.00

Ilustración 46 Mal cálculo de impuesto (Daily Report)

Fuente: Propia



Sales

Sales with 15% tax

Product	Price	Quantity	Subtotal	Tax	Total
Coca Cola	17.00	5	85.00	15.00	100.00
			85.00	15.00	100.00

Ilustración 47 Cálculo correcto de impuesto (Daily Report)

Fuente: Propia

En el módulo de Pisto se añadieron filtros por fecha en las pantallas de las órdenes cerradas y en el Daily Report. Antes de esto el usuario no podía consultar información de fechas anterior sin cambiar la fecha del sistema.

Para hacer el filtro se ingresaron dos Date Pickers en las pantallas. Estos pickers modifican la URL añadiendo la fecha inicial y final para enviar por parámetros URL al router de la aplicación. El router se encarga de extraer las fechas en URL y enviar las fechas a componente del reporte para que este haga el filtrado por fechas de las órdenes y al final para mostrar el filtro al usuario.

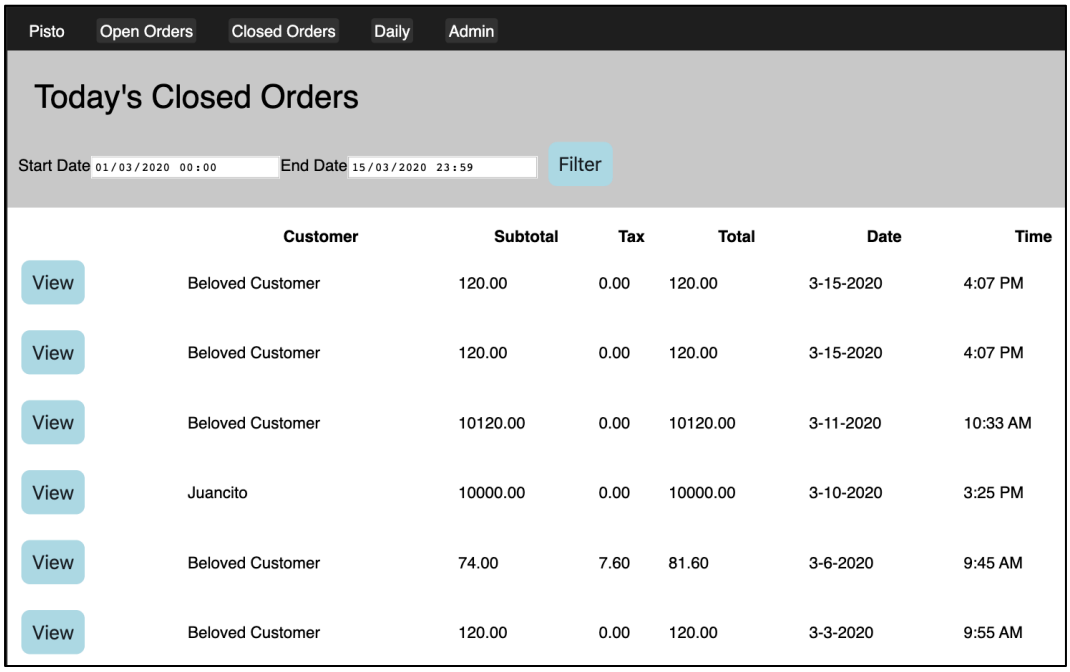


Ilustración 48 Filtro de fechas en Closed Orders

Fuente: Propia

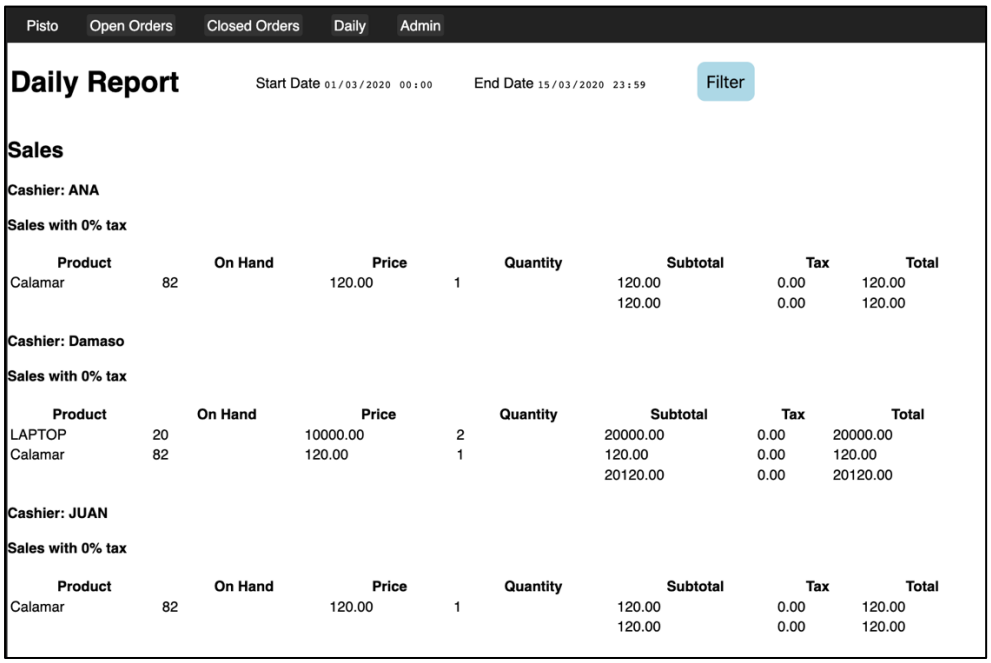


Ilustración 49 Filtro de fechas en Daily Report

Fuente: Propia

Para hacer la mejorar la navegación de usuario en la pantalla de Open Order, se cambió el sistema de etiquetado de los productos donde solo había un nivel estaba

representado por lista de strings por una de dos niveles donde el primero están las categorías y el segundo etiquetas. Por ejemplo, un articulo tiene una categoría llamada zapatos y sus etiquetas son deportivo, americano, hombres. Se borró el campo de Tags que tenia el producto y se añadió dos campos de tipo lista de tipo de la entidad Tag. Ahora un producto puede tener varias categorías y varias etiquetas con la única condición que esas etiquetas pertenezcan a las categorías del producto.

Se agregó la gestión de Tags en los módulos de Pisto y Move donde se pueden crear, leer, modificar y eliminar tags. También se puede asignas etiquetas a las etiquetas así convirtiéndolas en categorías. Se deben cumplir una serie de condiciones para eliminar etiquetas, las etiquetas deben estar sin padre y no deben estar ligadas a uno o varios productos.

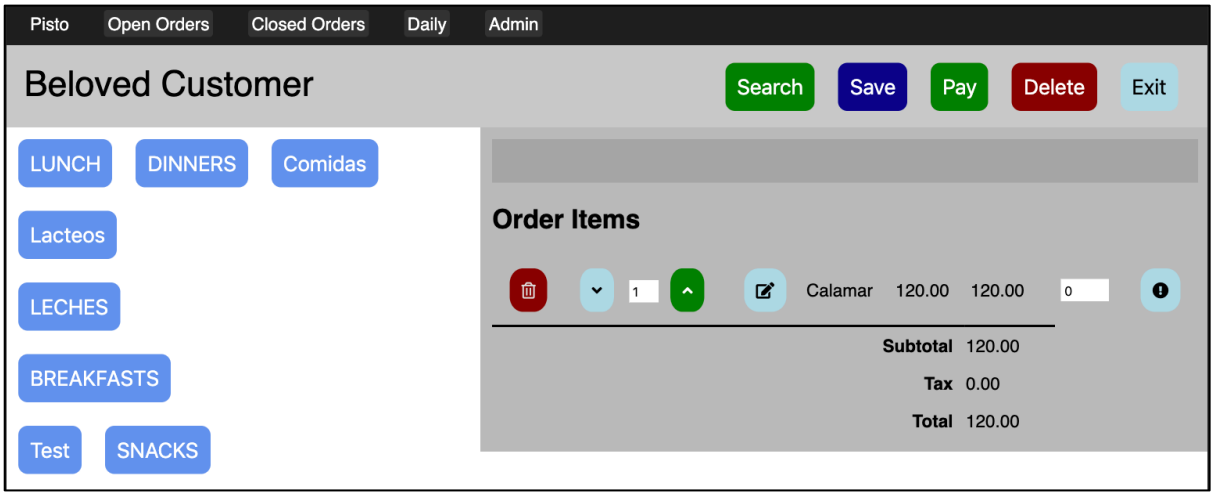


Ilustración 50 Nuevo sistema de etiquetado

Fuente: Propia

Luego se corrigió un error con los pines de los usuarios y cajeros donde se podía ingresar un PIN mayor a cuatro dígitos con los pines deben ser exactamente de 4 dígitos. Para esto se validó la longitud de los pines en los inputs donde se ingresaban por ejemplo al momento de crear o editar un usuario.

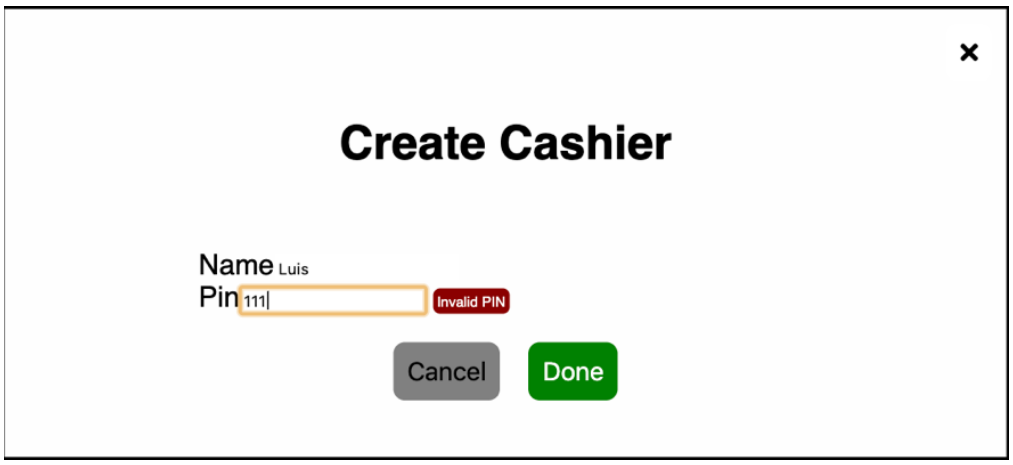


Ilustración 51 Validación de PIN

Fuente: Propia

Se agregaron descuentos individuales a los artículos de las compras. Siguiendo la misma estrategia que se ejecutó en los descuentos individuales de las órdenes. Se creó un campo para guardar el descuento en la entidad Movement donde una compra puede tener varios y un movimiento solo pertenece a una compra. Añadió un input a cada artículo de la compra para que el usuario pueda ingresar un valor entre 0 y 100. Y ese descuento solo puede afectar al artículo asignado.

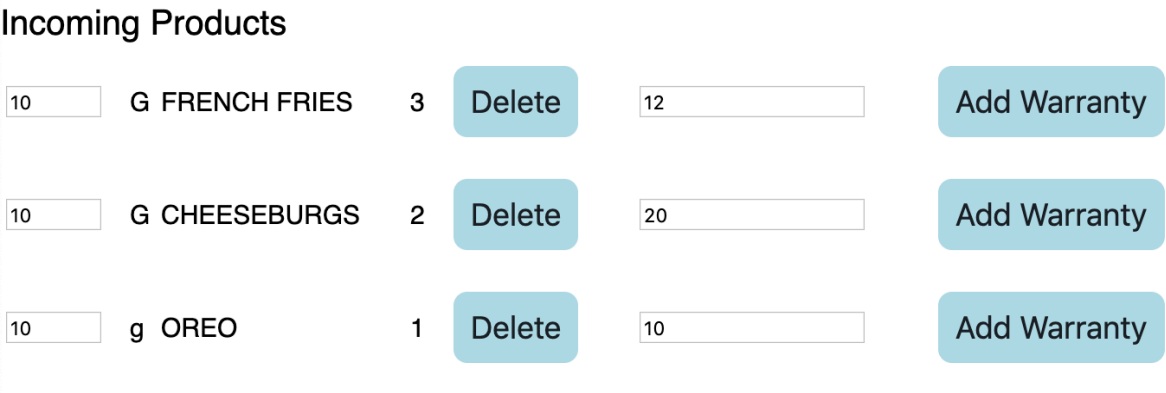


Ilustración 52 Descuentos de 12% 20% y 10%

Fuente: Propia

Aunque esos descuentos no se calculen porque ahora no es la prioridad, el usuario deberá hacer los cálculos manualmente agregando el total comprado, el impuesto y el descuento. En la siguiente ilustración se puede observar como se ejecutaría esto.

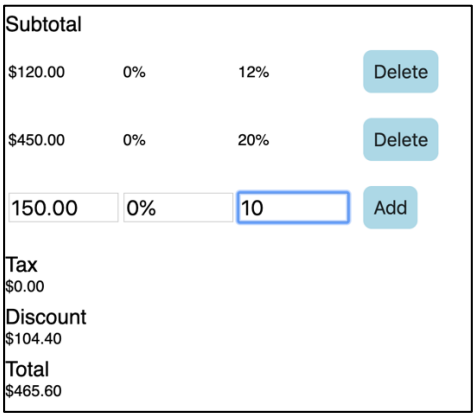


Ilustración 53 Creación de totales en las compras

Fuente: Propia

Se trabajó en la sincronización de la base de datos. El usuario debía ingresar unas credenciales para sincronizarse con otros dispositivos que usen la misma base de datos. La sincronización funcionaba, pero había un problema, no iniciaba sin la intervención del usuario, el usuario debía refrescar para que iniciara el proceso. Para solucionar esto se agregó una funcionalidad para que al momento de hacer clic la página recargue.

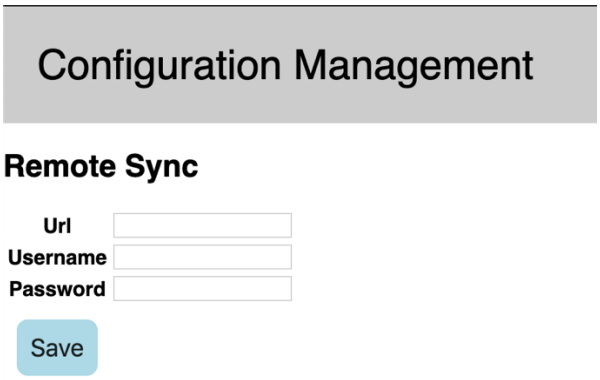


Ilustración 54 Sincronización remota en la configuración de Pisto y Move

Fuente: Propia

Después de la sincronización remota, se agregó la garantía a los productos donde en Move se crean y en Pisto se visualizan. Se agregó una entidad al Cafe Domain llamada Warranty donde la relación entre los productos y las garantías es de uno a uno. Cuando se están haciendo las compras de un producto en ese momento se

agregan. Se despliega un modal para hacer tal acción. Luego el usuario puede eliminarlo o modificarlo usando el mismo modal.

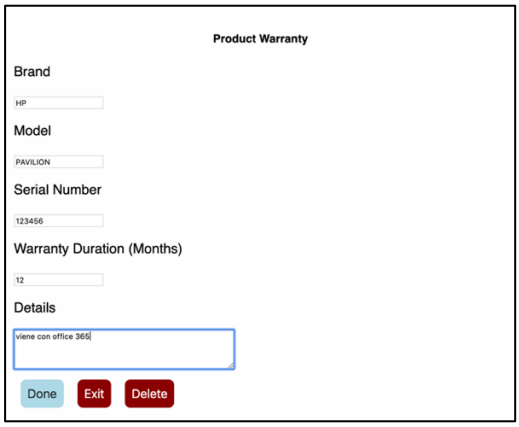


Ilustración 55 Formulario de Garantías

Fuente: Propia

En Pisto se agregó un botón a los artículos de la orden que muestra un modal con los datos donde el cajero debe leerlos al comprador y al momento de imprimir el recibo se hizo un apartado para imprimir la garantía. También se le dio la opción de mostrar la garantía en la vista de las órdenes cerradas.

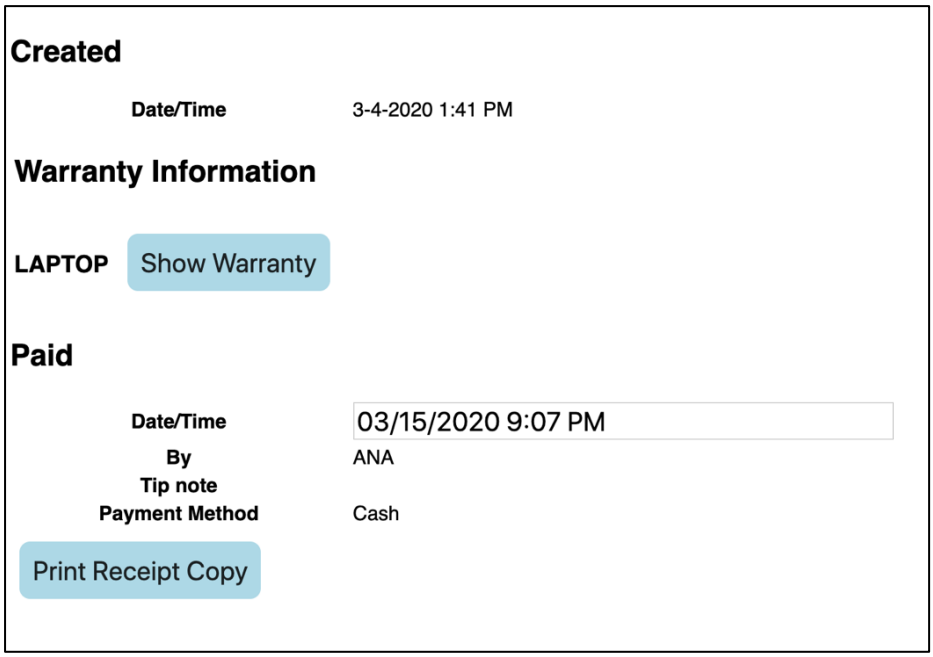


Ilustración 56 Botón de garantía en una orden cerrada

Fuente: Propia

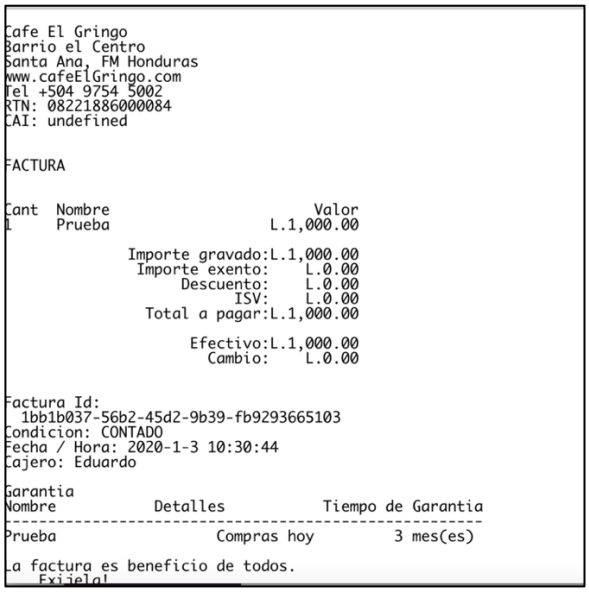


Ilustración 57 Factura con la garantía de un producto

Fuente: Propia

En el mes de diciembre el proceso de cambio del UI de PistoPOS comenzaba. Se tuvo que investigar que UI frameworks teníamos disponible para ReasonReact y cual seria el más sencillo de implementar. No había muchos para el lenguaje ReasonML y muchos de ellos estaban sin terminar y con un gran período de inactividad. La librería más completa fue BS Material UI que es un binding de Material UI escrito en JavaScript y está recibiendo soporte.

Durante los meses de enero, febrero y marzo se ha estado trabajando en la nueva interfaz de Pisto. El proceso consiste en que los diseñadores trabajaban en una pantalla cuando ya estaba lista se hacía una pequeña reunión para dar el visto bueno se procedía a escribir las development tasks para especificar que lo que se trabajará. Estos también servían para hacer pruebas para asegurarse que después diseño todo este bien.

Primero se comenzó con la pantalla Home donde el objetivo era que el usuario aún siga creando ordenes. Se le aplico el nuevo diseño a la pantalla cambiando el encabezado se removió el input para el nombre del cliente para moverlo a un diálogo que se despliega cuando se crea la orden.

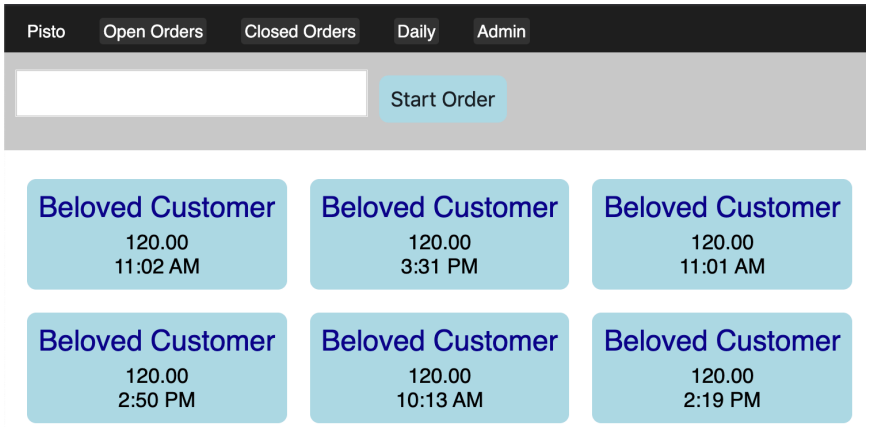


Ilustración 58 Vieja interfaz de creación de órdenes

Fuente: Propia

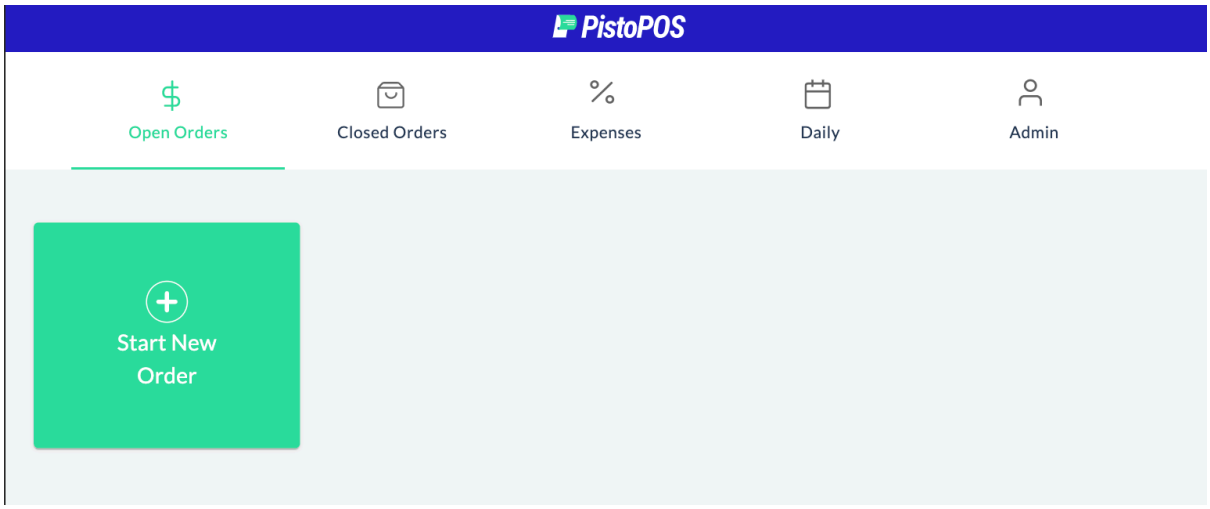


Ilustración 59 Nueva interfaz de creación de órdenes

Fuente: Propia

Luego se hicieron los cambios en las pantallas de las órdenes abiertas. Esta pantalla contiene muchas partes así que dividieron en tareas pequeñas. Lo primero que se hizo fue remover el encabezado porque en esta pantalla no se mostraría luego se añadieron las acciones de la orden. Después se le aplicó el nuevo diseño a las categorías, etiquetas y botones de los productos.

El siguiente paso fue trabajar en el carrito de la orden. En este paso las funciones de la pantalla de Pago pasaron a esta. Ahora el usuario puede agregar artículos usando la nueva interfaz.

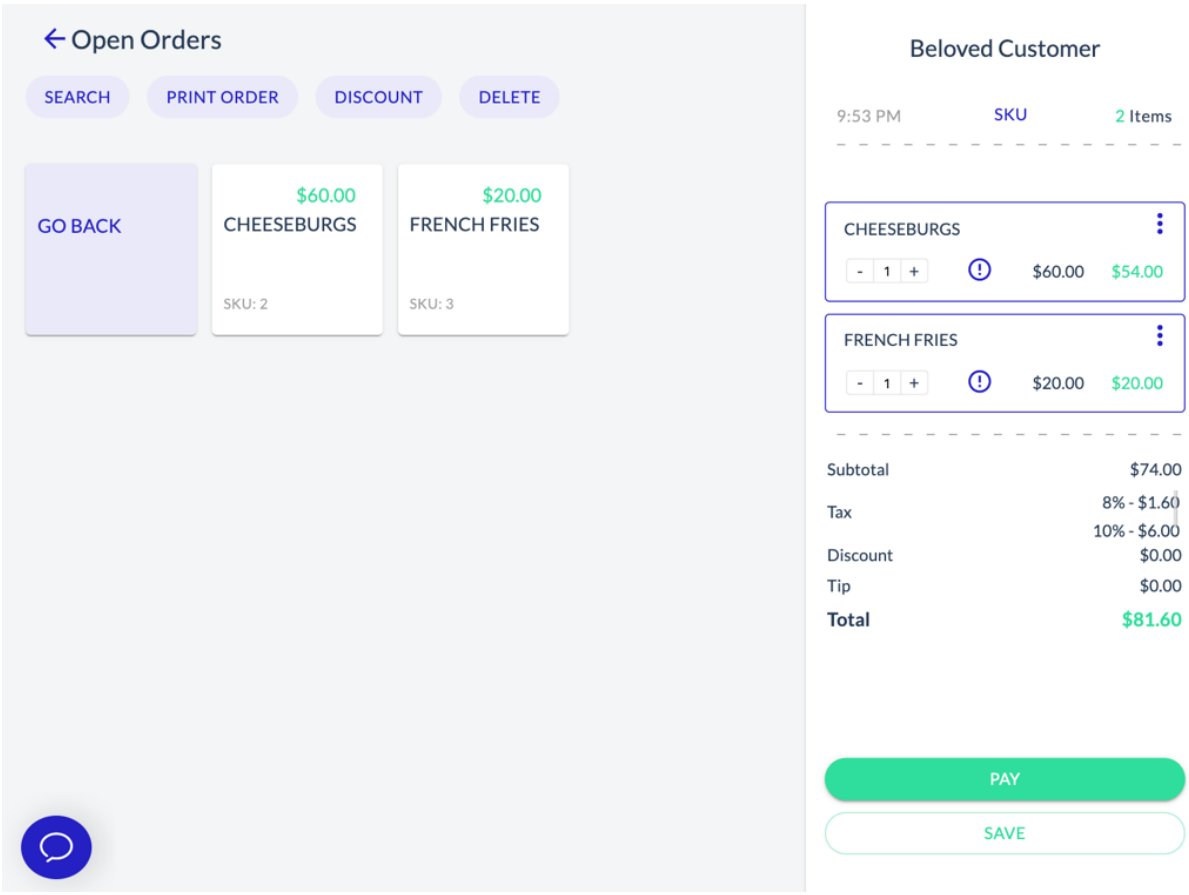


Ilustración 60 Nueva interfaz de creación de órdenes

Fuente: Propia

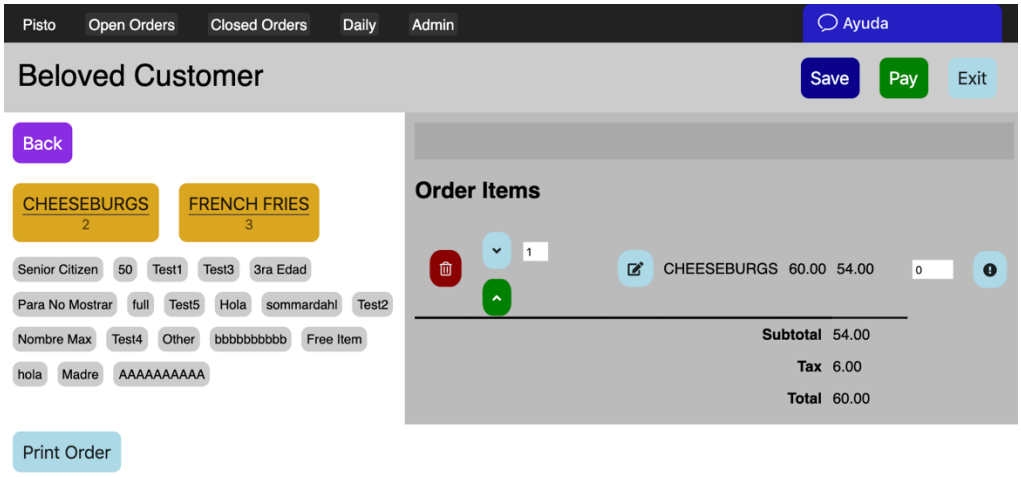


Ilustración 61 Vieja interfaz de creación de órdenes

Fuente: Propia

Después de eso se agregaron varias funcionalidades como hacer pagos con tarjeta de crédito o en efectivo. Se agregó un modal para ingresar las propinas. Se aplicó diseño a las notas que puede ser escritas en un artículo de la orden. En parte superior se agregó un botón que utilizado para mostrar un input para agregar productos usando el campo único SKU.

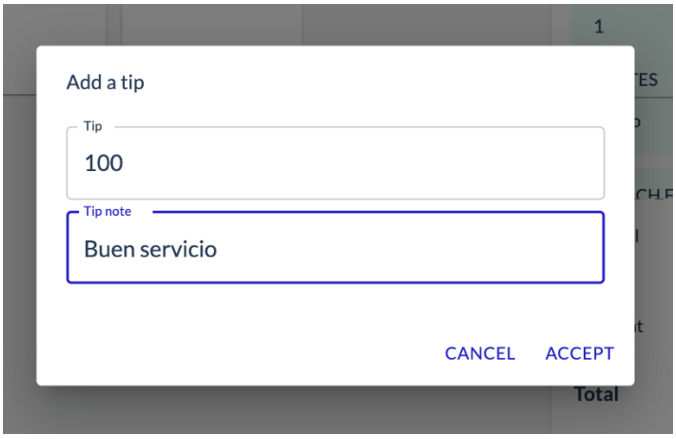


Ilustración 62 Nueva interfaz de creación de propinas

Fuente: Propia

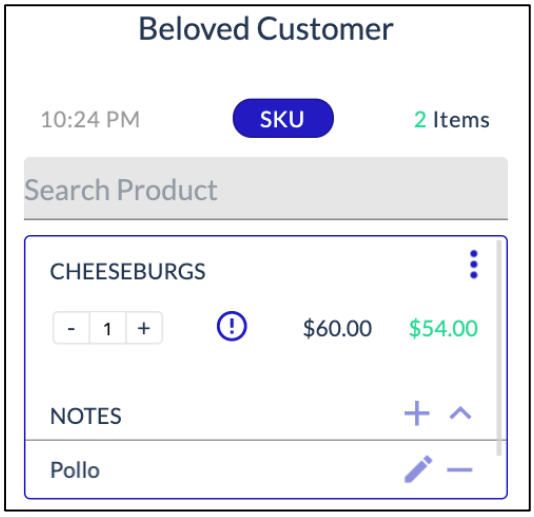


Ilustración 63 Nueva interfaz de manejo de notas

Fuente: Propia

El paso siguiente fue aplicar diseño a los descuentos generales, al modal de eliminación, los descuentos individuales, un de buscador de productos por nombre y el dialogo de la garantía. Con esto el proceso se rediseñó de la pantalla de Open Orders fue terminada con éxito.

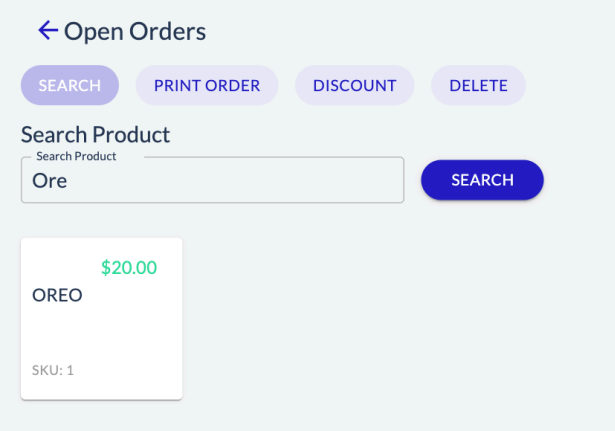


Ilustración 64 Nuevo buscador de productos

Fuente: Propia

La siguiente pantalla que tuvo una transformación fue la pantalla de configuración de la aplicación, pero esto vino acompañado la implementación de uso de moneda en Pisto y Move. Para realizar esto se agregó un campo al controlador de la configuración en el módulo de Cafe Data para manejar la moneda. Luego se debió agregar una función de utilidad que me devuelva el símbolo internacional del dólar o del lempira para ser usando en todo lugar que un numero represente un valor monetario.

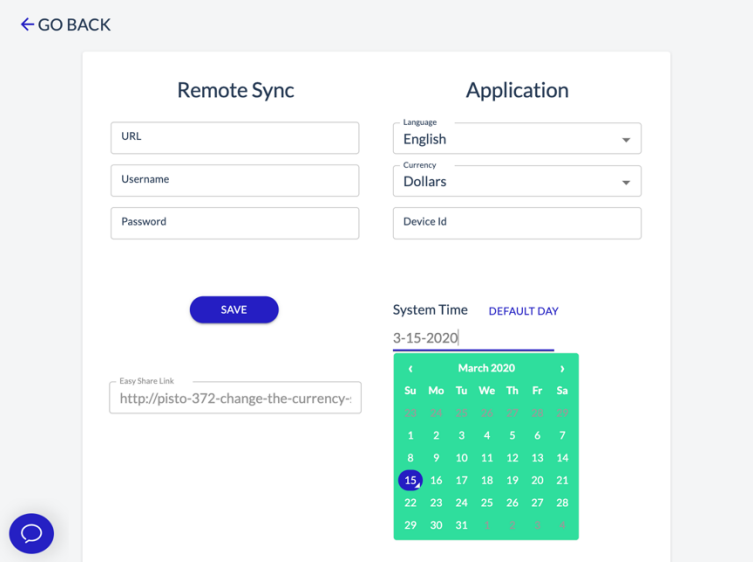


Ilustración 65 Nueva pantalla de configuración

Fuente: Propia

Cuando se finalizó el nuevo diseño de la configuración. Se procedió con la gestión de cajeros, productos y descuentos, cabe destacar estos componentes son muy simples. Estas pantallas constan de funciones de CRUD donde la única diferencia es el formulario para crear o editar. De ahí las demás partes son muy similares. Y la razón es que estos vienen de una clase o componente padre llamado ItemManager donde estos componentes hacen pequeños cambios a las funciones de renderizado. Y el ItemManager solo las invoca usando polimorfismo.

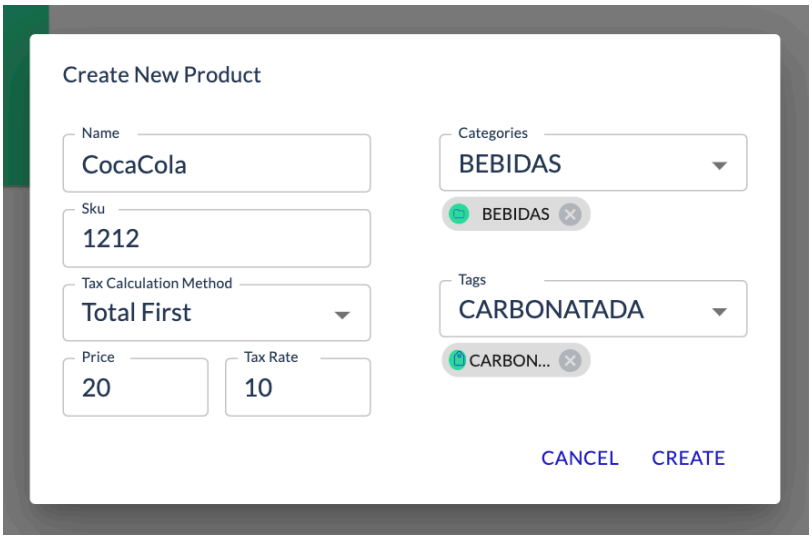


Ilustración 66 Formulario de creación de productos

Fuente: Propia

Ilustración 67 Nueva pantalla de administración de productos

Fuente: Propia

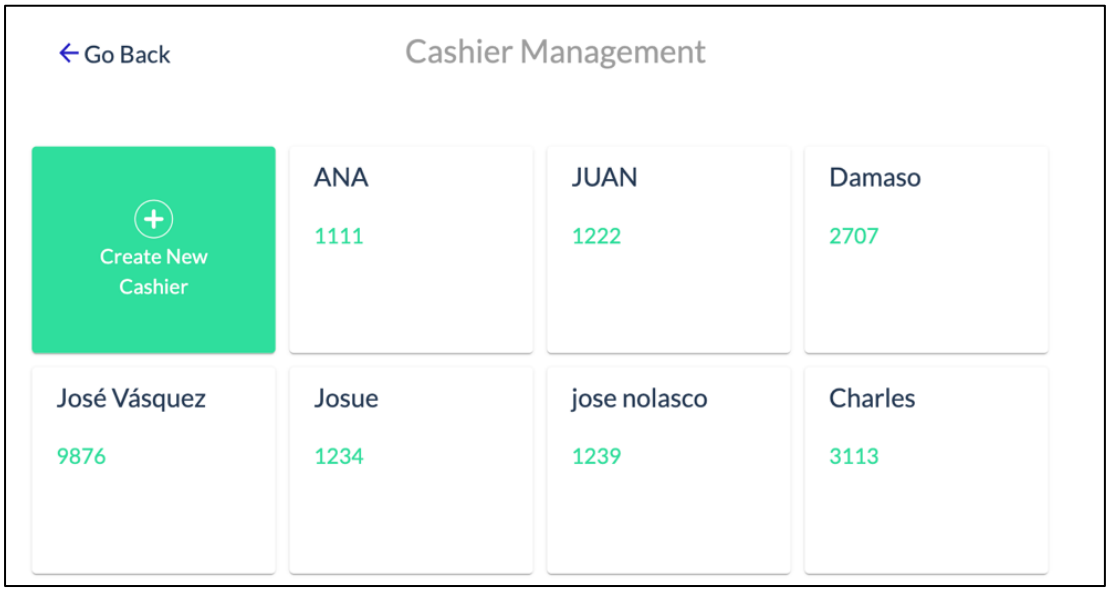


Ilustración 68 Nueva pantalla de administración de cajeros

Fuente: Propia

Ilustración 69 Nueva pantalla de administración de descuentos

Fuente: Propia

4.2. BITÁCORA DE TAREAS REALIZADAS

Tabla 4 Actividades realizadas desde 7 de octubre hasta 3 de noviembre

Semana	Actividades
7 de octubre – 13 de octubre	• Curso introductorio de la organización • Introducción al Proyecto PistoPOS • Investigar sobre el lenguaje de programación ReasonML y el framework ReasonReact • Hacer trabajos de testing para familiarizarme con el proyecto • Se corrigió un error que permitía al usuario cambiar la fecha de pago a una fecha anterior a la fecha de creación.
14 de octubre – 20 de octubre	• Recibiendo más entrenamiento acerca de ReasonML y el framework de ReasonReact • Se le agregaron nuevos campos a la vista del Daily Report de Pisto. El usuario puede ver la cantidad de producto existente. • Se comenzó la actualización de dependencias en Pisto, actualizar código de los componentes y solucionar un bug que aparecía al momento de compilación.
21 de octubre – 27 de octubre	• Se completo la actualización de las dependencias y componentes y corrigió el bug de compilación. • Introducción a los módulos FAF Intake y FAF Printer
28 de octubre – 3 de noviembre	• Se inició y terminó la actualización de Move. Se actualizaron las dependencias, componentes y también se solución el bug que entorpecía la compilación del proyecto. • Se comprobó el funcionamiento de FAF Printer, debía imprimir el código de barra correcto para un producto.

Fuente: Propia

Tabla 5 Actividades realizadas desde 4 de noviembre hasta 1 de diciembre

Semana	Actividades
4 de noviembre – 10 de noviembre	• Se le agregó el campo isArchive a los productos para ocultar los productos que han sido "eliminados" y evitar entidades que tienen relación con ellas fallen. • Las ventas en el Daily Report se agrupan por cajero. • Se implementaron las propinas para una orden en pisto. • Se implementaron descuentos individuales para los artículos de una orden. • Se le agrego la opción de exportar el dashboard de Move a CSV
11 de noviembre – 17 de noviembre	• Se asistió a un campamento de la empresa en Tegucigalpa. El campamento estuvo orientado a compartir con los compañeros y adquirir conocimiento gracias conferencias y talleres. • Solucionar el bug en el cálculo de impuestos. Se tuvo que hacer cambios en la fórmula.
18 de noviembre – 24 de noviembre	• Se agregó la opción de filtrar por fechas en las órdenes cerradas. • Se agregó la opción de filtrar por fechas en el Daily Report.
25 de noviembre – 1 de diciembre	• Se cambió la estructura de los Tags en los productos. Se agregó un nuevo nivel llamado categorías. Un producto puede tener varias categorías y esas categorías puede tener varios Tags. • Se corrigió un bug con el pin de los cajeros y usuarios, estos permitían pines mayores a 4 dígitos.

Fuente: Propia

Tabla 6 Actividades realizadas desde 2 de diciembre hasta 5 de enero

Semana	Actividades
2 de diciembre – 8 de diciembre	• Se agregó la nueva estructura de categorías a Move. • Se solucionó un bug en la nueva estructura de las categorías. Fallaba cuando se borraba un Tag que un producto lo estaba usando.
9 de diciembre – 15 de diciembre	• Se investigó varios frameworks para el UI de Pisto. • Se les agregó descuentos individuales a los artículos de una compra.
16 de diciembre – 22 de diciembre	• Se le agregó sincronización automática de la base de datos de Pisto y Move • Se procedió a agregar la opción de agregar una garantía a los productos en Move y el despliegue de esta en Pisto
23 de diciembre – 29 de diciembre	• Se siguió trabajando en la garantía.
30 de diciembre – 5 de enero	• Se agregó la información de la garantía en el recibo de la orden si los productos lo poseían.

Fuente: Propia

Tabla 7 Actividades realizadas desde 6 de enero hasta 2 de febrero

Semana	Actividades
6 de enero – 12 de enero	• Se eligió Material UI para el UI del proyecto. • Se estudió si es factible usar Material UI con ReasonReact. • Se comenzó a cambiar el UI de proyecto Pisto. Se inició con la pantalla de Home.
13 de enero – 19 de enero	• Se empezó aplicar el nuevo diseño a la pantalla de new/edit Open Order • Se aplicó el nuevo diseño las categorías y etiquetas en la pantalla de new/edit Open Order • Se aplicó el nuevo diseño a la zona del carrito de Open Orders
20 de enero – 26 de enero	• Se aplicó el diseño a las notas de los artículos de una Open Orders. • Se eliminó la pantalla de Pay Order y se movió la funcionalidad de pago en efectivo con la nueva interfaz a pantalla de Open Orders.
27 de enero – 2 de febrero	• Se concluyó en nuevo proceso de pago en efectivo. • Se aplicó el nuevo diseño al modal de la garantía. • Se agregó un modal para agregar un descuento individual a los artículos de la orden. • Se agregó la funcionalidad de pagar con tarjeta de crédito o debito en la página de Open Order.

Fuente: Propia

Tabla 8 Actividades realizadas desde 3 de febrero hasta 1 de marzo

Semana	Actividades
3 de febrero – 9 de febrero	• Se solucionó un bug en las notas de los artículos de una Open Orders. • Se agregó un modal para agregar artículos al carrito usando el campo único SKU. • Se aplicó el nuevo diseño a los descuentos que son aplicables a toda la orden.
10 de febrero – 16 de febrero	• Se aplicó el nuevo diseño al modal de confirmación al eliminar una Open Order. • Se aplicó el nuevo diseño al modal para agregar un articulo al carrito haciendo una búsqueda por nombre.
17 de febrero – 23 de febrero	• Se hizo una reunión con los nuevos integrantes del proyecto para transferir conocimiento. • Se hizo un rebase de las ramas para prepararlas para el deploy.
24 de febrero – 1 de marzo	• Se le agregó la opción de cambiar la moneda a Pisto y Move • Se le aplicó el nuevo diseño a la pantalla de administración. • Se le aplicó el nuevo diseño a la pantalla de configuración de Pisto. • Se le aplicó el nuevo diseño a la pantalla de administración de cajeros de Pisto.

Fuente: Propia

Tabla 9 Actividades realizadas desde 2 de marzo hasta 11 de marzo

Semana	Actividades
2 de marzo – 8 de marzo	• Se le aplicó el nuevo diseño a la pantalla de administración de productos de Pisto. • Se le aplicó el nuevo diseño a la pantalla de administración de descuentos de Pisto.
9 de marzo – 11 de marzo	• Se arregló un bug relacionado a los Date Pickers de los descuentos. Fallaba cuando se cambiaba una fecha, problemas de conversión.

Fuente: Propia

V. CONCLUSIONES

- Se entendió porque las practicas Ágiles son tan importantes para Acklen Avenue y HeroUnit.
- Se agregaron con éxito las nuevas funcionalidades a los módulos de Pisto, agregando nuevas entidades a la base de dato y mejorar funciones ya existentes.
- La nueva interfaz de usuario de PistoPOS comenzó y se desarrolla sin problemas.
- Se adquirió mucho conocimiento gracias al proyecto Pisto y de todas las herramientas involucradas en el desarrollo.
- Se aplicaron las buenas practicas de programación creando código limpio y entendible para futuros desarrolladores.

VI. RECOMENDACIONES

Se recomienda a los futuros desarrolladores investigar todas las tecnologías que están en auge o aquellas que tengan un futuro prometedor por ejemplo el lenguaje ReasonReact esto porque las empresas de software siempre están al pendiente de herramientas que faciliten el desarrollo de software. Los alumnos deben de tener conocimiento de backend como en frontend.

Se recomienda que los alumnos tengan fluidez al momento de hablar ingles para tener una buena comunicación con clientes en el extranjero.

Se recomienda a los futuros desarrolladores tomar cursos sobre UI/UX porque hay muchas variables involucradas como la combinación de colores incluso el uso del lenguaje formal o informal hace un impacto.

Desde mi experiencia adquirida en mi práctica profesional recomiendo a los alumnos tomar cursos sobre automatizaciones por ejemplo pruebas automáticas. Tomar cursos sobre software de integración continua como Travis, CircleCI o Gitlab Pipelines. Estas herramientas ayudan a los desarrolladores a ahorrar tiempo y evitar los trabajos tediosos.

BIBLIOGRAFÍA

Chacon, S., & Straub, B. (2014). *ProGit* (2a ed.). Apress.

David J Anderson, & Andy Carmichael. (2016). *Kanban Esencial Condensado*.

David Kopal. (2018, septiembre 20). Psst! Here's why ReasonReact is the best way to write React. Recuperado el 5 de junio de 2020, de FreeCodeCamp.org website: https://www.freecodecamp.org/news/psst-heres-why-reasonreact-is-the-best-way-to-write-react-5088d434d035/

Facebook. (2019, febrero 10). Components and Props [Informatica]. Recuperado de Components and Props website: https://reactjs.org/docs/components-and-props.html

Facebook. (s/f). Componentes y propiedades – React. Recuperado el 5 de junio de 2020, de https://es.reactjs.org/docs/components-and-props.html

Facebook. (s/f). ReactJS. Recuperado de https://es.reactjs.org/

Fernández, J. (s/f). *Introducción a las metodologías ágiles*. Catanluya, España: Universida Oberta de Catalunya.

Francia, J. (s/f). Sprint Review. Recuperado el 7 de marzo de 2020, de Scrum.org website: https://www.scrum.org/resources/blog/sprint-review-spanish-edition

Haworth, S. (2019, abril 23). Waterfall Vs Agile. Recuperado de

The digital project manager website: https://thedigitalprojectmanager.com/es/agile-frente-a-waterfall/

Hethey, J. M. (2013). *GitLab Repository Management*. Birmingham B3 2PB, UK.: Packt Publishing.

Javatpoint. (s/f). What is Pouchdb—Javatpoint. Recuperado el 5 de junio de 2020, de Www.javatpoint.com website: https://www.javatpoint.com/what-is-pouchdb

Kanbanize. (s/f). Qué es Kanban: Fundamentos | Kanbanize. Recuperado el 21 de marzo de 2020, de Kanban Software for Agile Project Management website: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban/

Knowledgehu Tutorials. (2018, julio 25). Sprint Execution Overview | Sprint Execution Activities. Recuperado el 6 de marzo de 2020, de KnowledgeHut Tutorial website: https://www.knowledgehut.com/tutorials/agile-management/scrum-tutorial

Leonard, T. (2014, junio 6). Python to OCaml: Retrospective—Thomas Leonard's blog. Recuperado el 5 de junio de 2020, de http://roscidus.com/blog/blog/2014/06/06/python-to-ocaml-retrospective/

Mukherjee, P. (s/f). ReactJS - State in React [Informatica]. Recuperado de ReactJS - State in React website:

https://www.geeksforgeeks.org/reactjs-state-react/

Normand, E. (2019, septiembre 23). Is React functional programming? Recuperado de Is React functional programming? website: https://lispcast.com/is-react-functional-programming/

Novoseltseva, E. (2018, enero 25). Katas de código; ser buen programador y crecer profesionalmente. Recuperado el 22 de marzo de 2020, de Apiumhub website: https://apiumhub.com/es/tech-blog-barcelona/katas-de-codigo/

Parra, M. (2016, noviembre 24). ¿Qué es y cómo funciona Trello? - Postedin. Recuperado el 8 de marzo de 2020, de https://www.postedin.com/blog/que-es-y-como-funciona-trello/

Pedro Ventura. (2011, mayo 19). Motores de navegadores web: Gecko, Trident, WebKit y otros. Recuperado el 5 de junio de 2020, de https://www.pedroventura.com/desarrollo-web/motores-de-navegadores-web-gecko-trident-webkit-y-otros/

Perez, E. (s/f). *Introducción a CSS*.

Polanco, J., & Pedersen, A. (s/f). *React In-Depth*. DevelomenpArc.

Radigan, D. (s/f). Waterfall vs agile: How to manage agile programs. Recuperado el 5 de junio de 2020, de Atlassian website: https://www.atlassian.com/agile/project-

management/program

Rafatpanah, R., & D'mello, B. J. (2019). *ReasonML Quick Start Guide*. WOW! eBook.

Ramos Vega, C. (2017, febrero 20). Los artefactos de Scrum. Recuperado el 7 de marzo de 2020, de Blog de Cristina Ramos Vega. website: http://guarenty-group.com/cz/

Riva, M. (2018, junio 5). The rise of ReasonML. Recuperado de The rise of ReasonML website: https://medium.com/openmindonline/better-javascript-with-reasonml-6d5ba70f70e8

Roche, J. (s/f). Artefactos Scrum: Las 3 herramientas clave de gestión. Recuperado el 8 de marzo de 2020, de Deloitte Spain website: https://www2.deloitte.com/es/es/pages/technology/articles/artefactos-scrum.html

Schwaber, K. (s/f). Online Nexus Guide. Recuperado el 5 de junio de 2020, de Scrum.org website: https://www.scrum.org/resources/online-nexus-guide

Sutherland, J., & Schwaber, K. (2017). *La Guía de Scrum*.

The Kanban Board. (2015, diciembre 21). Recuperado el 5 de junio de 2020, de Denver Peak Academy website: https://denpeakacademy.com/2015/12/21/the-kanban-board/

tiThink. (2018, noviembre 14). 7 razones para utilizar React. Recuperado de 7 razones para utilizar React website:

https://www.tithink.com/es/2018/11/14/7-razones-para-utilizar-react/

Vega, A. A. (2017, diciembre 17). Tipos de Componentes en ReactJS. Recuperado de https://medium.com/@alonsus91/tipos-de-componentes-en-reactjs-f387a6f8e2b7