### Puntos de historia

El método de los puntos de historia es la forma de realizar estimaciones de esfuerzo y tiempo a las historis de usuario. También nos sirve para medir la velocidad del sprint en la reunión de retrospectiva.

Se escoge una historia de usuario sencilla, una que todo el mundo entiende, para emplearla como referencia. Esa sería la definición de 1 punto de historia en el proyecto. Para estimas las otras historias de usuario, bastará con compararlo con la historia de referencia.

Se emplea la actividad de planning pocker para determinar el valor de los puntos y el tiempo de las historias de usuario, según el siguiente cuadro. (números de Fibonacci)

|  |  |  |
| --- | --- | --- |
| **SIZE** | **POINTS** | **TIME [Days]** |
| XtraSmall (XS) | 1 | ½ |
| Small (S) | 2 | 1 |
| Medium (M) | 3 | 2 |
| Large (L) | 5 | 3 |
| XtraLarge (XL) | 8 | 5 |

*Tabla 1: Puntos historia*

## MARCO PRÁCTICO

### Modelo del negocio

El sistema de información propuesto según los alcances y limitaciones [Pág. [5](#_bookmark13)], está orientado para cualquier negocio o empresa cuyo rubro principal consiste en la venta de comidas y bebidas; éstas pueden ser restaurant, pensiones, fondas, franquicias de comida rápida, etc.

Como el proyecto no se enfoca en un negocio o empresa en particular, se hace uso del product vision board para comprender mejor la visión del producto que se va a desarrollar.

### Product Vision Board

|  |  |  |  |
| --- | --- | --- | --- |
| **ctrlFOOD** | Administración y control económico en la venta de alimentos. | | |
| **Usuarios** | **Necesidades (requerimientos)** | **Producto** | **Valor** |
| * Restaurants * Pensiones * Fondas * Pollerías * Franquicias de comida rápida | * Controlar las ventas que se realizan * Realizar Informes económicos diarios y mensuales * Realizar reportes estadísticos para publicidad o descuentos | * Sistema de información web * Administración de usuarios y permisos * Gestión de productos * Gestión de clientes * Administración de las ventas * Generación de recibos * Generación de informes económicos * Generación de reportes estadísticos | * Control económico fiable * Reportes estadísticos confiables * Publicidad * Descuentos a clientes potenciales |

*Tabla 2: Product Vision Board del proyecto*

### Roles y stackeholders

Con fines académicos, los roles de Scrum están representados por una sola persona.

* **Product Owner:** Saul Mamani M.
* **Scrum Master:** Saul Mamani M.
* **Development Team**: Saul Mamani M.
* **Client:** Restaurants, Pensiones, Pollerías, Fondas, Franquicias de comida rápida, etc.

### Historias de usuario

Para la determinación de requerimientos y/o características del sistema se recolectan historias de usuario.

|  |  |
| --- | --- |
| **HU1: Ingresar al sistema como administrador** | |
| **Como** | Administrador |
| **Quiero** | Ingresar al sistema con una cuenta y un password |
| **Para** | Tener el acceso a toda la funcionalidad del sistema, incluyendo la funcionalidad de los vendedores |

*Tabla 3: Historia de usuario – Ingresar al sistema como administrador*

|  |  |
| --- | --- |
| **HU2: Ingresar al sistema como vendedor** | |
| **Como** | Vendedor |
| **Quiero** | Ingresar al sistema con una cuenta y un password |
| **Para** | Vender productos y generar informes económicos |

*Tabla 4: Historia de usuario – Ingresar al sistema como vendedor*

|  |  |
| --- | --- |
| **HU3: Gestionar usuarios del sistema** | |
| **Como** | Administrador |
| **Quiero** | Listar, insertar, modificar y eliminar datos de los vendedores |
| **Para** | Que utilicen el sistema según su rol seleccionado |

*Tabla 5: Historia de usuario – Gestionar usuarios del sistema*

|  |  |
| --- | --- |
| **HU4: Administrar perfil de usuario** | |
| **Como** | Vendedor |
| **Quiero** | Ver y actualizar mis datos personales y mi foto de perfil |
| **Para** | Personalizar mi cuenta y cambiar la foto de perfil |

*Tabla 6: Historia de usuario – Administrar perfil de usuario*

|  |  |
| --- | --- |
| **HU5: Cambiar contraseña** | |
| **Como** | Vendedor |
| **Quiero** | Cambiar mi contraseña asignada |
| **Para** | Mantener seguro mi sesión y mis acciones en el sistema |

*Tabla 7: Historia de usuario – Cambiar contraseña*

|  |  |
| --- | --- |
| **HU6: Gestionar productos** | |
| **Como** | Vendedor |
| **Quiero** | Listar, buscar, insertar, modificar y eliminar productos categorizados en comidas y bebidas |
| **Para** | Tener un catálogo completo de las comidas y bebidas al momento de registrar las ventas solicitadas |

*Tabla 8: Historia de usuario – Gestionar productos*

|  |  |
| --- | --- |
| **HU7: Gestionar clientes** | |
| **Como** | Vendedor |
| **Quiero** | Listar, buscar, insertar, modificar y eliminar datos de los clientes |
| **Para** | Registrar los datos del cliente al realizar una venta, llevar un adecuado control de los clientes y generar reportes estadísticos |

*Tabla 9: Historia de usuario – Gestionar clientes*

|  |  |
| --- | --- |
| **HU8: Listar ventas** | |
| **Como** | Vendedor |
| **Quiero** | Listar todas las ventas realizadas por rango de fechas, mostrando el total vendido |
| **Para** | Llevar un control económico de todo lo que he vendido. Además de, re imprimir los recibos, anular y/o restablecer ventas erradas |

*Tabla 10: Historia de usuario – Listar ventas*

|  |  |
| --- | --- |
| **HU9: Registrar ventas** | |
| **Como** | Vendedor |
| **Quiero** | Registrar en el sistema la venta de uno o varios productos del catálogo de comidas o bebidas, y buscar o registrar los datos del cliente |
| **Para** | Llevar un control económico adecuado de todas las ventas realizadas |

*Tabla 11: Historia de usuario – Registrar ventas*

|  |  |
| --- | --- |
| **HU10: Generar número de ticket** | |
| **Como** | Vendedor |
| **Quiero** | Que el sistema genere un número secuencial de ticket de forma diaria |
| **Para** | Llamar al cliente y que recoja su pedido |

*Tabla 12: Historia de usuario – Generar número de ticket*

|  |  |
| --- | --- |
| **HU11: Generar recibo** | |
| **Como** | Vendedor |
| **Quiero** | Que el sistema genere un recibo con los datos de la venta realizada, la información del cliente, y un código QR |
| **Para** | Tener constancia de que la transacción se ha realizado de manera correcta |

*Tabla 13: Historia de usuario – Generar recibo*

|  |  |
| --- | --- |
| **HU12: Anular ventas** | |
| **Como** | Vendedor |
| **Quiero** | Anular las ventas que yo he registrado y que tengan datos erróneos. Las ventas se pueden anular solo el día en el que se realizó la transacción y antes de generar el informe económico |
| **Para** | Que no sean tomados en cuenta a la hora de generar el informe económico diario |

*Tabla 14: Historia de usuario – Anular ventas*

|  |  |
| --- | --- |
| **HU13: Restablecer ventas** | |
| **Como** | Vendedor |
| **Quiero** | Restablecer una venta que yo he anulado previamente por error. Las ventas se pueden restablecer solo el día en que se realizó la transacción y antes de generar el informe económico diario |
| **Para** | Que vuelva a ser tomada en cuenta a la hora de generar el informe económico diario |

*Tabla 15: Historia de usuario – Restablecer ventas*

|  |  |
| --- | --- |
| **HU14: Generar informe económico** | |
| **Como** | Vendedor |
| **Quiero** | Quiero que el sistema genere informes económicos diarios, mensuales y anuales de todas las ventas realizadas |
| **Para** | Llevar un control adecuado de todas las transacciones que se han realizado en un determinado tiempo |

*Tabla 16: Historia de usuario – Generar informe económico*

|  |  |
| --- | --- |
| **HU15: Generar reportes estadísticos** | |
| **Como** | Administrador |
| **Quiero** | Que el sistema genere reportes estadísticos económicos y reportes estadísticos de los clientes potenciales |
| **Para** | Tener una adecuada toma de decisiones, como lanzar y publicar promociones y descuentos especiales |

*Tabla 17: Historia de usuario – Generar reportes estadísticos*

### Product backlog

La pila del producto pendientes de desarrollar está constituida por las historias de usuario, y ordenamos según prioridad de implementación.

|  |  |  |
| --- | --- | --- |
| **Historia de**  **usuario** | **Descripción** | **Prioridad** |
| **HU1** | Ingresar al Sistema como administrador | 1 |
| **HU2** | Ingresar al sistema como vendedor | 1 |
| **HU3** | Gestionar usuarios del sistema | 1 |
| **HU4** | Administrar perfil de usuario | 1 |
| **HU5** | Cambiar contraseña | 1 |
| **HU6** | Gestionar productos | 1 |
| **HU7** | Gestionar clientes | 1 |
| **HU8** | Listar ventas | 2 |
| **HU9** | Registrar ventas | 2 |
| **HU10** | Generar número de ticket | 2 |
| **HU11** | Generar recibo | 2 |
| **HU12** | Anular ventas | 3 |

|  |  |  |
| --- | --- | --- |
| **HU13** | Restablecer ventas | 3 |
| **HU14** | Generar informe económico | 3 |
| **HU15** | Generar reportes estadísticos | 3 |

*Tabla 18: Product backlog*

### Estimación del product backlog

Para estimar el tiempo y el esfuerzo de las historias de usuario se emplea el método de los puntos de historia, y planning pocker para asignar valores a cada historia de acuerdo a la Tabla 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint** | **HU** | **Descripción** | **Size** | **Puntos**  **de historia** | **Tiempo [Días]** |
| **Primer Sprint** | **HU1** | Ingresar al Sistema como administrador | XS | 1 | ½ |
| **HU2** | Ingresar al sistema como vendedor | XS | 1 | ½ |
| **HU3** | Gestionar usuarios del sistema | S | 2 | 1 |
| **HU4** | Administrar perfil | XS | 1 | ½ |
| **HU5** | Cambiar contraseña | XS | 1 | ½ |
| **HU6** | Gestionar productos | S | 2 | 1 |
| **HU7** | Gestionar clientes | S | 2 | 1 |
| **Segundo Sprint** | **HU8** | Listar ventas | S | 2 | 1 |
| **HU9** | Registrar ventas | L | 5 | 3 |
| **HU10** | Generar número de ticket | XS | 1 | ½ |
| **HU11** | Generar recibo | XS | 1 | ½ |
| **Tercer Sprint** | **HU12** | Anular ventas | XS | 1 | ½ |
| **HU13** | Restablecer ventas | XS | 1 | ½ |
| **HU14** | Generar informe económico | M | 3 | 2 |
| **HU15** | Generar reportes estadísticos | M | 3 | 2 |
| **Puntos de historia / Tiempo estimado (Time Boxing)** | | | | **27** | **15** |

*Tabla 19: Product backlog estimado*

De acuerdo al tamaño y a la cantidad de historias de usuario, el tiempo estimado para el desarrollo del sistema es de quince días, las cuales la dividimos en tres iteraciones o sprints de 5 días cada una.

### Definición de hecho

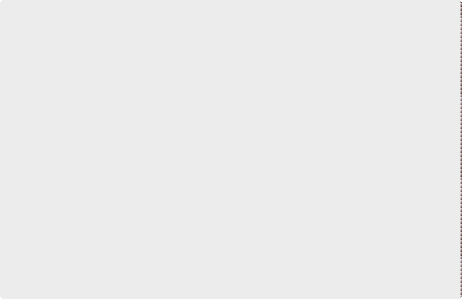
Se establece la “definición de hecho” cuando todas las tareas definidas en el sprint backlog de una iteración se hayan completado, se han verificado los criterios de aceptación, y se ha desplegado el product increment en un servidor de pruebas como Heroku.

### Definición de la arquitectura

Para modelar y analizar la arquitectura del sistema se utiliza el diagrama de componentes y el diagrama de despliegue del Lenguaje de Modelado Unificado.

### Diagrama de componentes

El diagrama de componentes muestra la arquitectura lógica del sistema software y los componentes necesarios para construir la aplicación web.



«web application»

**ctrlFOOD**

«PHP library»

**domPDF 1.0**

**.env**

«PHP framework»

**Laravel 7.2**

**Models**

**Views**

«templates»

**AdminLTE 5.7**

«library»

**jQuery.js 3.2**

«data base»

**MySQL 10.4.8**

**Controllers**

«library»

**vue.js 2**



**Descripción:**

ctrlFOOD: La aplicación web que se va a desarrollar bajo laarquitectura MVC

.env: Archivo de configuración del sistema postgreSQL 11: Gestor de base de datos

vue.js 2: Framework progresivo para construir interfaces de usuario jQuery 1.11: Biblioteca multiplataforma java script

AdminLTE 2: Panel de administración de usuario para Bootstrap domPDF: Librería PHP para la generación de reportes en PDF

Laravel 5.8: Framework de código abierto para desarrollar aplicaciones y servicios web con PHP

*Ilustración 7: Diagrama de componentes Fuente: Elaboración propia*

### Primer sprint

* + 1. **Sprint planning**

El sprint planning muestra **qué3** es lo que se tiene que desarrollar en esta iteración, y los tiempos comprometidos para finalizar cada historia de usuario.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **HU** | **Descripción** | **Puntos de historia** | **Tiempo [Días]** | **Inicio** | **Finalización** |
| **HU1** | Ingresar al Sistema como administrador | 1 | ½ | 02/09/2019 | 02/09/2019 |
| **HU2** | Ingresar al sistema como vendedor | 1 | ½ | 02/09/2019 | 02/09/2019 |
| **HU3** | Gestionar usuarios del sistema | 2 | 1 | 03/09/2019 | 03/09/2019 |
| **HU4** | Administrar perfil | 1 | ½ | 04/09/2019 | 04/09/2019 |
| **HU5** | Cambiar contraseña | 1 | ½ | 04/09/2019 | 04/09/2019 |
| **HU6** | Gestionar productos | 2 | 1 | 05/09/2019 | 05/09/2019 |
| **HU7** | Gestionar clientes | 2 | 1 | 06/09/2019 | 06/09/2019 |
| **Puntos de historia/Tiempo**  **estimado**  **(Time Boxing)** | | **10** | **5** |  | |

*Tabla 20: Primer sprint planning*

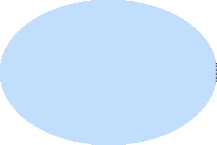
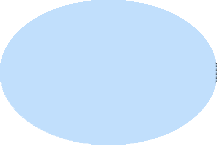
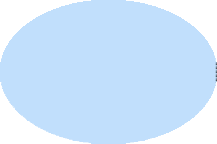
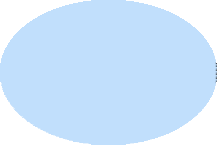
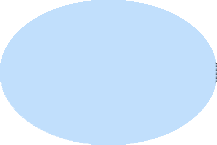
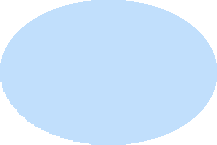
Para completar este sprint se estima un esfuerzo de 10 puntos de historia y tiempo de desarrollo de cinco días.

Para comprender mejor la funcionalidad global del sprint se modela un diagrama de casos de uso, y para modelar el esquema lógico de la base de datos de utiliza un diagrama de clases.

### Diagrama de casos de uso

El diagrama de casos de uso muestra la funcionalidad global del sprint.

3 La palabra **qué** es lo que se tiene que hacer, hace referencia al **análisis** de sistemas dentro el ciclo de vida un proyecto de desarrollo de software.



ctrlFOOD - Primer Sprint

«login»

**Iniciar Sesión**

**Administrar Perfil**

**Vendedor**

«CRUD»

**Gestionar clientes** «extend»

«CRUD»

**Gestionar productos**

**Cambiar contraseña**

**Administrador**

«CRUD»

**Gestionar usuarios**

*Ilustración 8: Diagrama de casos de uso primer sprint Fuente: Elaboración propia*



### Diagrama de clases persistentes

Se utiliza el diagrama de clases para modelar el esquema lógico de la base de datos.



* email: char
* email\_verified\_at: timestamps
* fotografia: char
* name: char
* password: char

«enum»

* rol: Rol

**User**

**Product**

«derive»



**Client**

* nit: int
* razon\_social: char



«dataType»

**Categoria**

* Bebida: char
* Comida: char
* fotografia: char
* nombre: char
* precio: float

«enum»

* categoria: Categoria

«derive»



«dataType»

**Rol**

* Administrador: char
* Vendedor: char

*Ilustración 9: Diagrama de clases primer sprint Fuente: Elaboración propia*

### Especificación de casos de uso

Para la especificación de los casos de uso utilizamos historias de usuario detallados.

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Iniciar Sesión | **Historia de usuario:** HU1, HU2 | | |
|  | **Actor:** Vendedor | **Depende de:** HU3 | |
| **Descripción:** Ingresar al sistema con una cuenta y un  password | | |
| **Criterio de Aceptación:**   * El actor ingresa su email y una contraseña y presiona el botón Aceptar. * El sistema busca las credenciales en la base de datos. * Una vez verificada las credenciales, el sistema abre la pantalla principal de acuerdo al rol del actor. Si es vendedor tiene acceso solo a la funcionalidad mostrada en diagrama de casos de uso, y si es administrador tiene acceso a toda la funcionalidad   del sistema. | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 21: Caso de uso iniciar sesión*

|  |  |  |
| --- | --- | --- |
| **Caso de uso:** Gestionar usuarios | **Historia de usuario:** HU3 | |
|  | **Actor:** Administrador | **Depende de:** |
| **Descripción:** Crear, modificar, eliminar y listar  usuarios del sistema | |
| **Criterio de Aceptación:**   * El sistema muestra la lista de todos los usuarios registrados y un menú con las siguientes opciones:   **Nuevo**   * El actor presiona el botón Nuevo * El sistema abre un formulario con los datos del usuario en base a la clase “User” del diagrama de clases * El actor llena los datos y presión el botón Guardar * El sistema inserta el registro en la base de datos y lanza un mensaje   **Modificar**   * El actor selecciona un registro presiona el botón Modificar * El sistema abre un formulario con los datos   del usuario seleccionado | |

|  |  |  |
| --- | --- | --- |
|  | * El actor modifica uno o más campos y presiona el botón Guardar * El sistema modifica los datos y lanza un mensaje   **Eliminar**   * El actor selecciona un registro presiona el botón Eliminar * El sistema muestra un mensaje de confirmación * El actor confirma el mensaje * El sistema elimina el registro de la base de datos | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | **Versión:**  1.0 |

*Tabla 22: Caso de uso gestionar usuarios*

|  |  |  |
| --- | --- | --- |
| **Caso de uso:** Gestionar clientes | **Historia de usuario:** HU7 | |
|  | **Actor:** Vendedor | **Depende de:** |
| **Descripción:** Crear, modificar, eliminar y listar  clientes | |
| **Criterio de Aceptación:**   * El sistema muestra la lista de todos los clientes registrados y un menú con las siguientes opciones:   **Nuevo**   * El actor presiona el botón Nuevo * El sistema abre un formulario con los datos del cliente en base a la clase “Client” del diagrama de clases * El actor llena los datos y presión el botón Guardar * El sistema inserta el registro en la base de datos y lanza un mensaje   **Modificar**   * El actor selecciona un registro presiona el botón Modificar * El sistema abre un formulario con los datos del cliente seleccionado * El actor modifica uno o más campos y presiona el botón Guardar * El sistema modifica los datos y lanza un mensaje | |

|  |  |  |
| --- | --- | --- |
|  | **Eliminar**   * El actor selecciona un registro presiona el botón Eliminar * El sistema muestra un mensaje de confirmación * El actor confirma el mensaje * El sistema elimina el registro de la base de datos | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | **Versión:**  1.0 |

*Tabla 23: Caso de uso gestionar clientes*

|  |  |  |
| --- | --- | --- |
| **Caso de uso:** Gestionar productos | **Historia de usuario:** HU6 | |
|  | **Actor:** Vendedor | **Depende de:** |
| **Descripción:** Crear, modificar, eliminar y listar  producto | |
| **Criterio de Aceptación:**   * El sistema muestra la lista de todos los productos registrados y un menú con las siguientes opciones:   **Nuevo**   * El actor presiona el botón Nuevo * El sistema abre un formulario con los datos del producto en base a la clase “Product” del diagrama de clases * El actor llena los datos, carga una imagen no más de 2MB de tamaño y presión el botón Guardar * El sistema inserta el registro en la base de datos y lanza un mensaje   **Modificar**   * El actor selecciona un registro presiona el botón Modificar * El sistema abre un formulario con los datos del producto seleccionado * El actor modifica uno o más campos y presiona el botón Guardar * El sistema modifica los datos y lanza un mensaje   **Eliminar**   * El actor selecciona un registro presiona el botón Eliminar * El sistema muestra un mensaje de   confirmación | |

|  |  |  |
| --- | --- | --- |
|  | * El actor confirma el mensaje * El sistema elimina el registro de la base de datos | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | **Versión:**  1.0 |

*Tabla 24: Caso de uso gestionar productos*

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Administrar perfil | **Historia de usuario:** HU4 | | |
|  | **Actor:** Vendedor | **Depende de:** HU3 | |
| **Descripción:** Administrar datos del usuario logeado | | |
| **Criterio de Aceptación:**   * Al actor presiona el botón Perfil de usuario de la pantalla principal * El sistema llena los datos del usuario logeado * El actor modifica uno o más datos, carga una foto de no más de 2MB, y presiona Aceptar * El sistema actualiza la información del usuario en   la base de datos, y refresca el sistema. | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 25: Caso de uso Administrar perfil*

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Cambiar contraseña | **Historia de usuario:** HU5 | | |
|  | **Actor:** Vendedor | **Depende de:** HU3, HU4 | |
| **Descripción:** Cambiar contraseña de un usuario  logado | | |
| **Criterio de Aceptación:**   * El sistema muestra la pantalla para cambiar contraseña * El actor llena la contraseña actual, la nueva contraseña y la verificación de la contraseña * El sistema verifica las credenciales en la base de datos, actualiza la contraseña, y lanza un mensaje de éxito.   o Las contraseñas deben estar protegidas por un algoritmo de cifrado. | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 26: Caso de uso Cambiar contraseña*

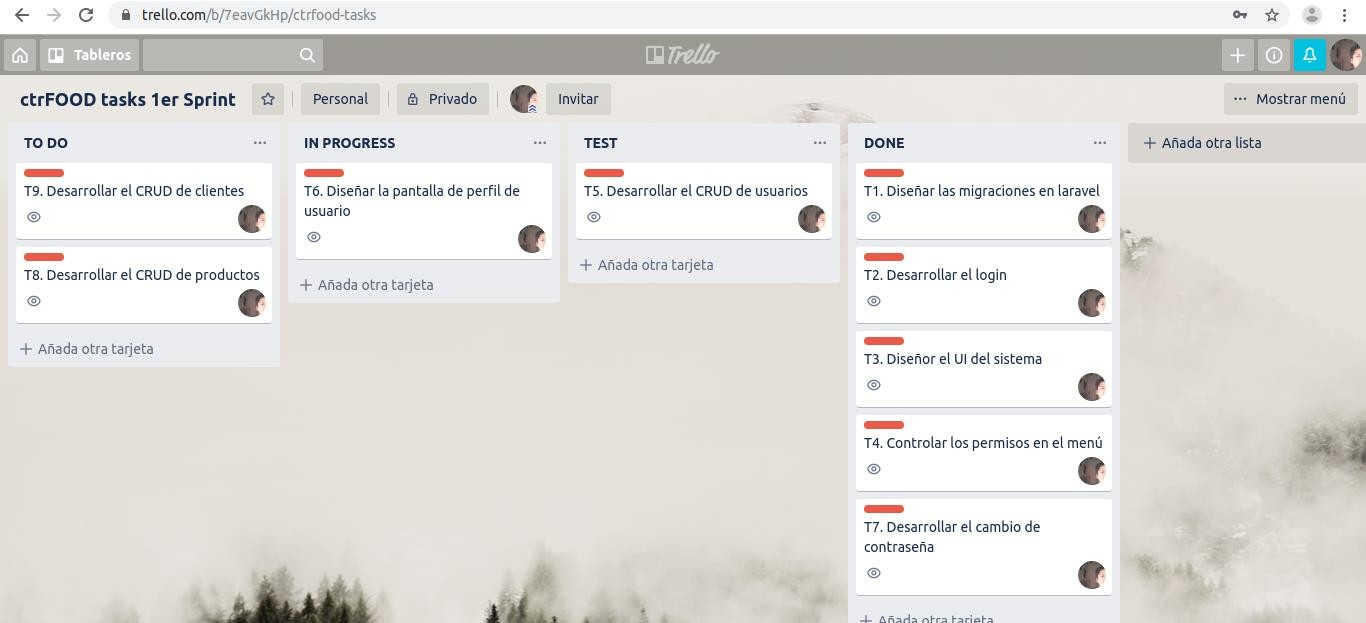
### Sprint backlog

Para determinar **cómo4** se va a desarrollar el sprint, se identifican una o varias tareas por historia de usuario, las cuales son agrupadas en un sprint backlog y asignadas a los miembros del equipo de desarrollo.

|  |  |  |
| --- | --- | --- |
| **HU** | **Descripción** | **Tareas** |
| **HU1** | Ingresar al Sistema como administrador | T1. Diseñar las migraciones en Laravel T2. Desarrollar el Login. |
| **HU2** | Ingresar al sistema como vendedor | T3. Diseñar el IU del sistema  T4. Controlar los permisos en el menú |
| **HU3** | Gestionar usuarios del sistema | T5. Desarrollar el CRUD de usuarios |
| **HU4** | Administrar perfil | T6. Diseñar la pantalla de perfil de usuario |
| **HU5** | Cambiar contraseña | T7. Desarrollar el cambio de contraseña |
| **HU6** | Gestionar productos | T8. Desarrollar el CRUD de productos |
| **HU7** | Gestionar clientes | T9. Desarrollar el CRUD de clientes |

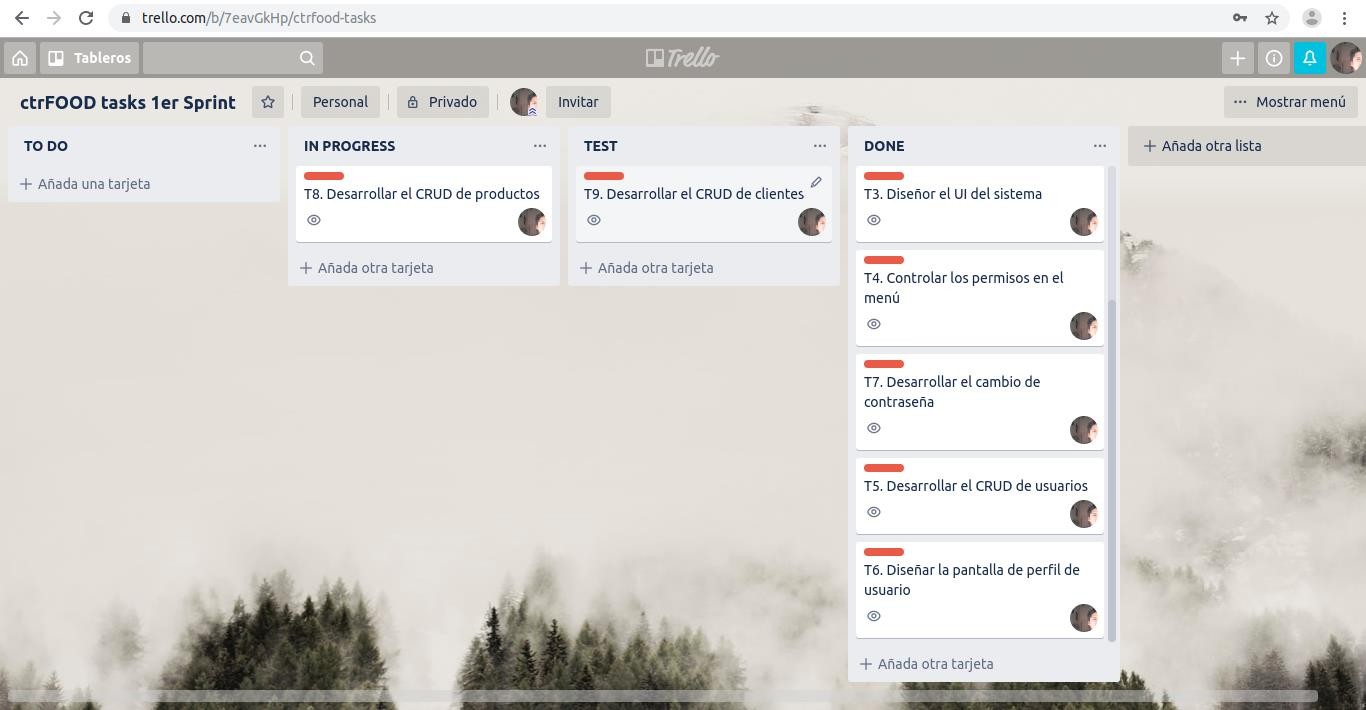
*Tabla 27: Primer sprint backlog*

### Sprint execution

Para controlar la ejecución del sprint y el avance de las tareas, empleamos un tablero Kanban con la herramienta Trello.

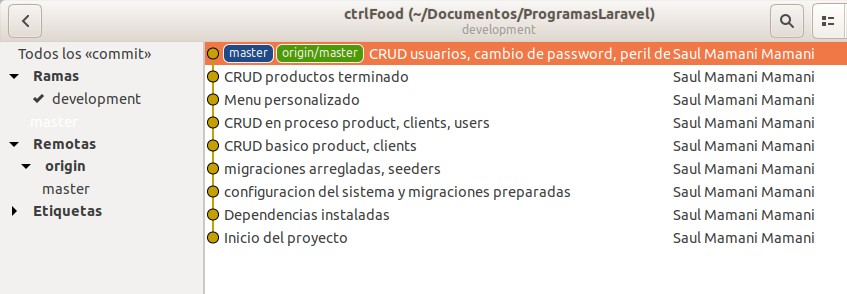
*Ilustración 10: Tablero Kanban primer sprint Fuente: Elaboración propia*

4 La palabra **cómo** se va a desarrollar, hace referencia al **diseño** de sistemas dentro el ciclo de vida un proyecto de desarrollo de software.



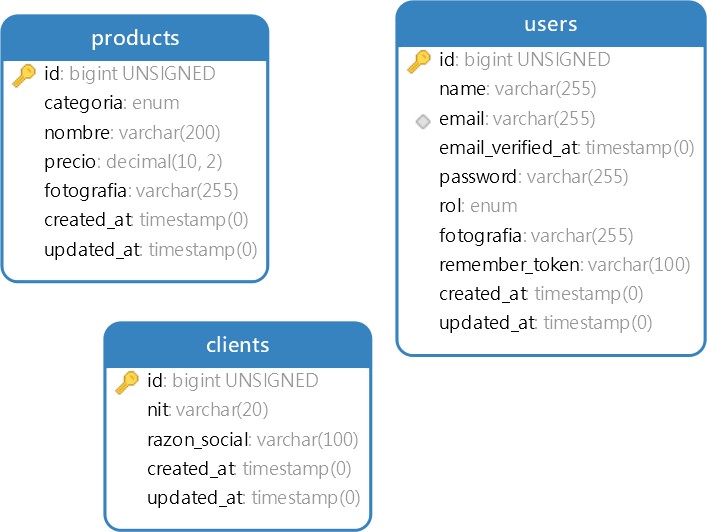
*Ilustración 11: Tablero Kanban primer sprint Fuente: Elaboración propia*

Para controlar el avance del código fuente se utiliza el sistema de control de versiones GIT, el cual se encuentra en GitHub bajo en el siguiente repositorio: <http://github.com/saulmamani/ctrlFood>



*Ilustración 12: Control de versiones del primer sprint Fuente: Elaboración propia*

El modelo relacional de la base de datos que se ha implementado en base al diagrama de clases persistentes.



*Ilustración 13: Modelo relacional de la base de datos del primer sprint Fuente: Elaboración propia*

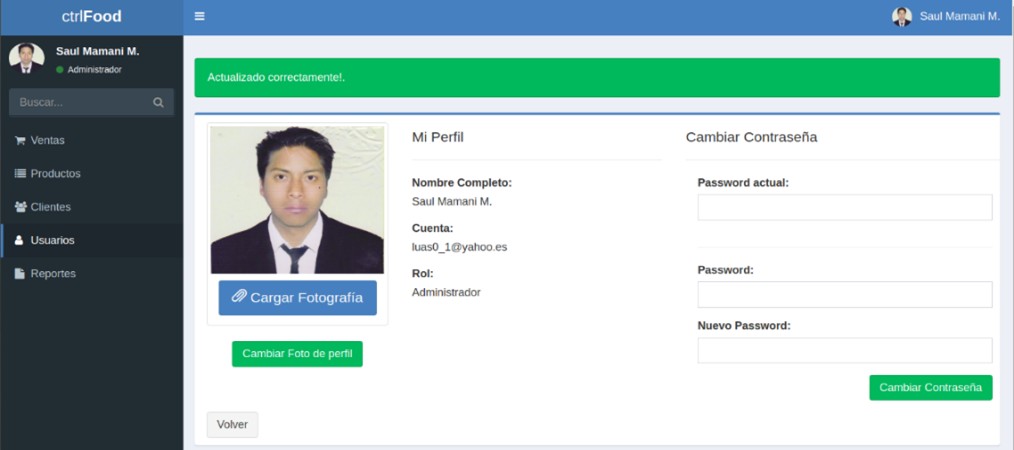
### Sprint review

El **product increment** representa los entregables realizados al product owner y al cliente, estos entregables reflejan el cumplimiento del primer sprint.

Se ha desplegado el sistema en la siguiente dirección URL, a modo de prueba para su revisión: [https://ctrlfood.herokuapp.com](https://ctrlfood.herokuapp.com/)



*Ilustración 14: Pantalla Login Fuente: Elaboración propia*



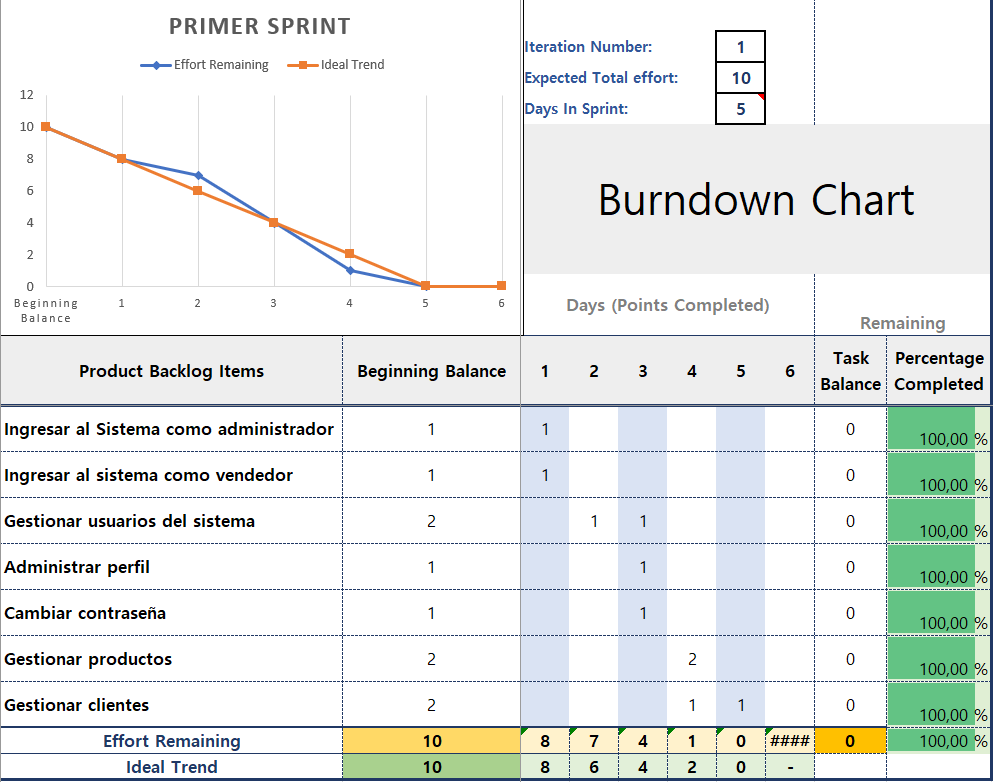
*Ilustración 15: Pantalla administrar perfil de usuario Fuente: Elaboración propia*



*Ilustración 16: Pantalla producto Fuente: Elaboración propia*

### Sprint retrospective

El desarrollo del primer sprint se concluyó de manera adecuada cumpliendo las tareas, los tiempos establecidos, y la definición de hecho establecido al inicio del sprint. Esta evolución se puede apreciar en el gráfico burndown.



*Ilustración 17: Seguimiento de historias primer sprint Fuente: Elaboración propia*

*Ilustración 18: Burndown chart primer sprint*

Primer sprint

10 8

8

6 8

4

2

0

Beginning Balance

6

6,4

4

3

4,8

2

3,2

0

1

2

3

1,6

4

4,44089E5-16

Days

Effort Remaining

Ideal Trend Lineal (Effort Remaining)

Story Points

### Segundo sprint

* + 1. **Sprint planning**

El sprint planning muestra **qué** es lo que se tiene que desarrollar en esta iteración, y los tiempos comprometidos para finalizar cada historia de usuario.

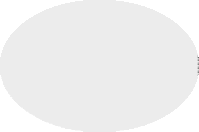
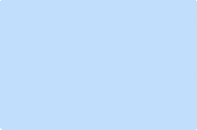
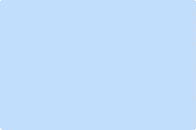
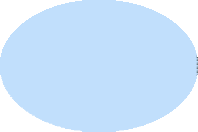
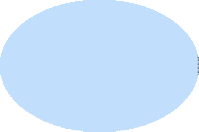
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **HU** | **Descripción** | **Puntos**  **de historia** | **Tiempo [Días]** | **Inicio** | **Finalización** |
| **HU8** | Listar ventas | 2 | 1 | 08/09/2019 | 09/09/2019 |
| **HU9** | Registrar ventas | 5 | 3 | 10/09/2019 | 12/09/2019 |
| **HU10** | Generar nro. de ticket | 1 | ½ | 13/09/2019 | 13/09/2019 |
| **HU11** | Generar recibo | 1 | ½ | 13/09/2019 | 13/09/2019 |
| **Puntos de historia / Tiempo**  **estimado (Time Boxing)** | | **9** | **5** |  | |

*Tabla 28: Segundo sprint planning*

Para completar este sprint se estima un esfuerzo de 9 puntos de historia y tiempo de desarrollo de cinco días.

### Diagrama de casos de uso

El diagrama de casos de uso muestra la funcionalidad global del sprint.



ctrlFOOD - Segundo Sprint

«CRUD»

**Gestionar clientes**

*(from Primer Sprint)*

«extend»

**Vendedor**

*(from Primer Sprint)*

«include»

«include»

«extend»

**Generar número de**

**ticket**

**Administrador**

*(from Primer Sprint)*

**Generar recibo**

*extension points*

imprimir recibo

**Listar ventas**

*extension points*

Nuevo cliente

**Registrar ventas**

*Ilustración 19: Diagrama de casos de uso segundo sprint*

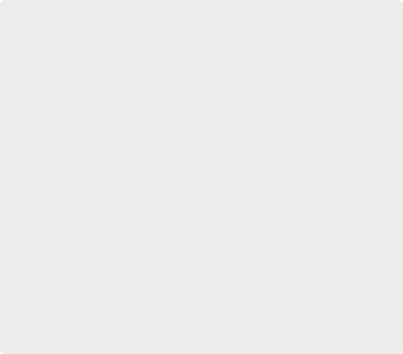
### Diagrama de clases persistentes

Se emplea el diagrama de clases para modelar el esquema lógico de la base de datos.

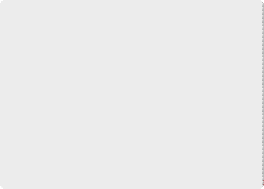
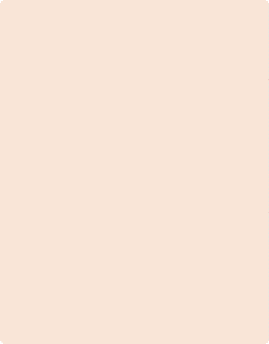
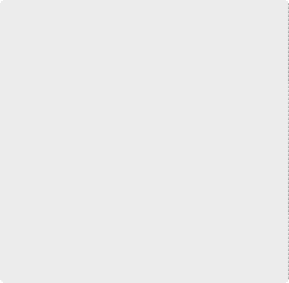


|  |  |
| --- | --- |
|  | «dataType»  **Rol** |
| «derive»  1  +Vendedor \* |
| * Administrador: char * Vendedor: char |

**User**



* email: char
* email\_verified\_at: timestamps
* fotografia: char
* name: char



* password: char

«enum»

* rol: Rol

«dataType»

**Categoria**

* + Bebida: char
  + Comida: char
* concepto: char

**Sale**

* estado: boolean
* fecha: date
* nit: int
* numero: int
* numero\_ticket: int
* razon\_social: char

1

0..1 1

**Client**

* nit: int
* razon\_social: char

«derive»

**Product**

* + fotografia: char
  + nombre: char
  + precio: float

«enum» 1

* + categoria: Categoria

1..\*

**Detail**

1 - cantidad: int

- precio: float

*Ilustración 20: Diagrama de clases segundo sprint Fuente: Elaboración propia*

### Especificación de casos de uso

Para la especificación de los casos de uso utilizamos historias de usuario detallados.

|  |  |  |
| --- | --- | --- |
| **Caso de uso:** Listar ventas | **Historia de usuario:** HU8 | |
|  | **Actor:**  Vendedor | **Depende de:** HU9 |
| **Descripción:** Muestra la lista de las  ventas realizadas | |
| **Criterio de Aceptación:**   * El sistema muestra la lista de todas las ventas realizadas y un formulario de búsquedas por rango de fechas, estado, y clientes. | |

|  |  |  |
| --- | --- | --- |
|  | * El actor llena los datos del formulario y pulsa el botón Buscar * El sistema actualiza el listado   según los parámetros de búsqueda | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | **Versión:**  1.0 |

*Tabla 29: Caso de uso Listar ventas*

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Registrar ventas | **Historia de usuario:** HU9 | | |
|  | **Actor:** Vendedor | **Depende de:**  HU6 | |
| **Descripción:** Registro de ventas de uno  más productos | | |
| **Criterio de Aceptación:**   * El sistema muestra la pantalla de ventas con el catálogo de los productos habilitados para vender. * El actor selecciona uno o más productos y los añade al carrito de ventas, además puede modificar las cantidades en el carrito. * El sistema calcula el subtotal y el total del monto a cobrar en bolivianos * El actor llena los datos de la venta, en base a las clases “Sale” y “Detail” del diagrama de clases. * Al llenar el campo Nit el sistema busca la razón social asociada a ese Nit, si lo encuentra lo llena en el campo correspondiente; y si no lo encuentra deja en campo vacío. * El actor presiona el botón Finalizar venta. * El sistema **genera un número de ticket,** inserta la nueva venta en la base de datos y prepara la   impresión del recibo. | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 30: Caso de uso Registrar ventas*

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Generar número de ticket | | **Historia de usuario:** HU10 | |
| **Actor:** Vendedor | | **Depende de:** HU9 | |
| **Descripción:** Genera el número de ticket de la transacción | | | |
| **Criterio de Aceptación:**   * Al registrar una nueva venta, el sistema genera un número de ticket que se va incrementando de uno en uno, y empieza en 1 al inicio de cada jornada.   *\* esta historia de usuario consiste en un algoritmo y no tiene una evidente interfaz de usuario* | | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 31: Caso de uso Generar número de ticket*

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Generar recibo | **Historia de usuario:** HU11 | | |
|  | **Actor:** Vendedor | **Depende de:** HU9, HU10 | |
| **Descripción:** Generación o impresión de la nota de  venta o recibo | | |
| **Criterio de Aceptación:**   * Después de realizar una venta, el sistema genera y muestra en una pantalla el recibo con todos los datos de la venta correspondiente, y crea un código QR con los datos del cliente, el número y el total vendido. * El actor presiona el botón imprimir * El sistema imprimir el recibo en una impresora pre instalada. | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 32: Caso de uso Generar recibo*

### Sprint backlog

Para determinar **cómo** se va a desarrollar el sprint, se identifican una o varias tareas por historia de usuario, las cuales son agrupadas en un sprint backlog y asignadas a los miembros del equipo de desarrollo.

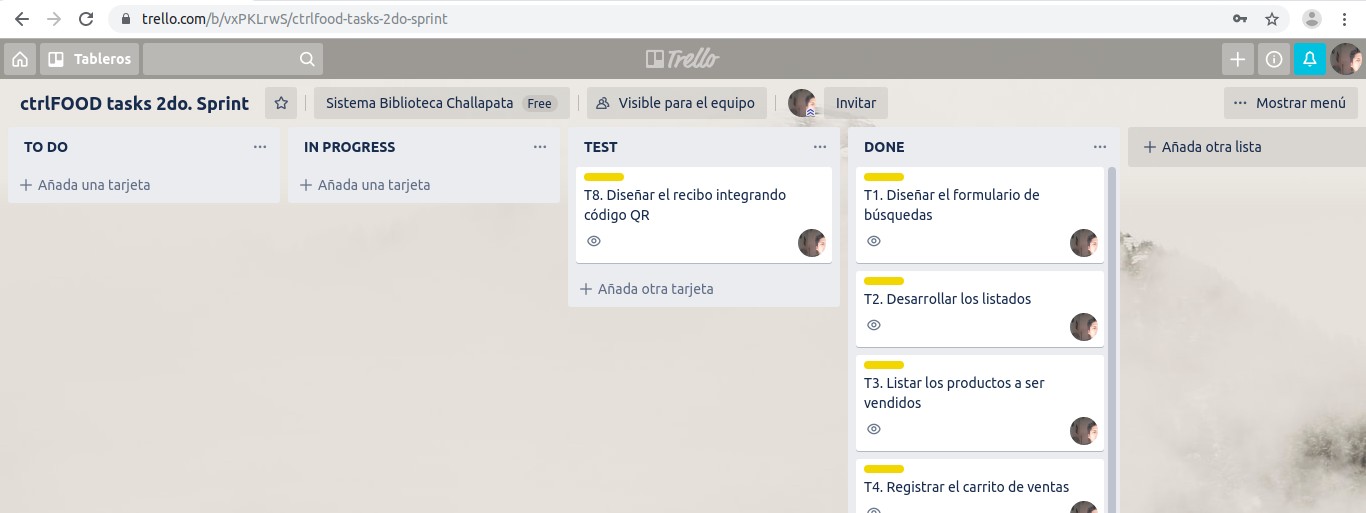
|  |  |  |
| --- | --- | --- |
| **HU** | **Descripción** | **Tareas** |
| **HU8** | Listar ventas | T1. Diseñar el Formulario de búsquedas  T2. Desarrollar los listados |
| **HU9** | Registrar ventas | T3. Listar los productos a ser vendidos T4. Registrar en el carrito de ventas T5. Calcular los totales  T6. Registrar la venta |
| **HU10** | Generar nro. de ticket | T7. Generar el número de ticket |
| **HU11** | Generar recibo | T8. Diseñar el recibo integrando un código QR |

*Tabla 33: Segundo sprint backlog*

### Sprint execution

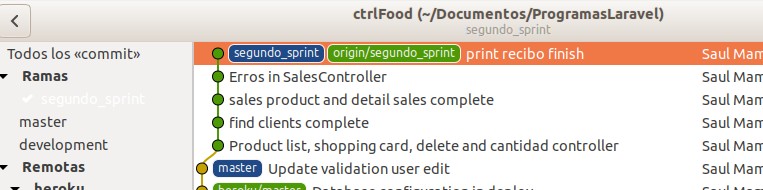
Para controlar la ejecución del sprint y el avance de las tareas, empleamos un tablero Kanban con la herramienta Trello.

*Ilustración 21: Tablero Kanban segundo sprint Fuente: Elaboración propia*

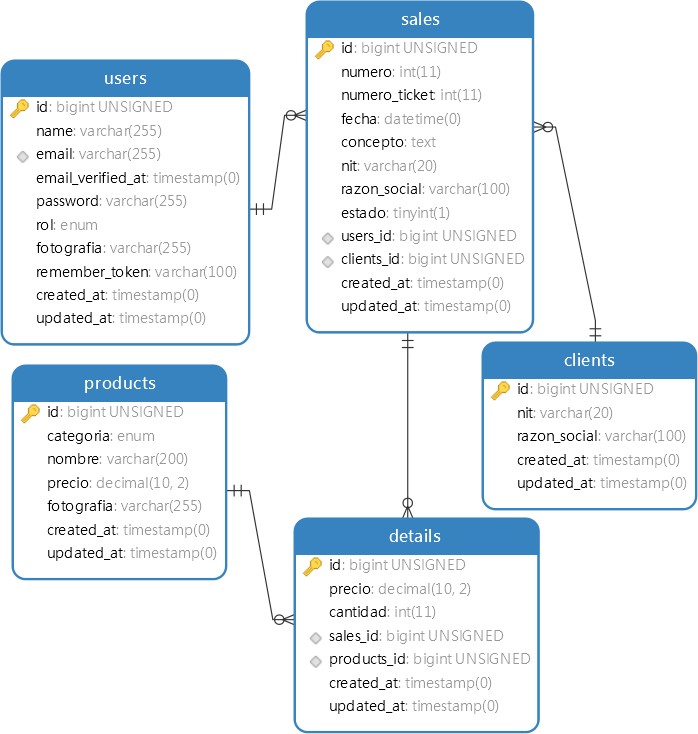


*Ilustración 22: Tablero Kanban segundo sprint*

Para controlar el avance del código fuente se utiliza el sistema de control de versiones GIT, el cual se encuentra en GitHub bajo en el siguiente repositorio: [http://github.com/saulmamani/ctrlFood](http://github.com/saulmamani/ctrlfood)



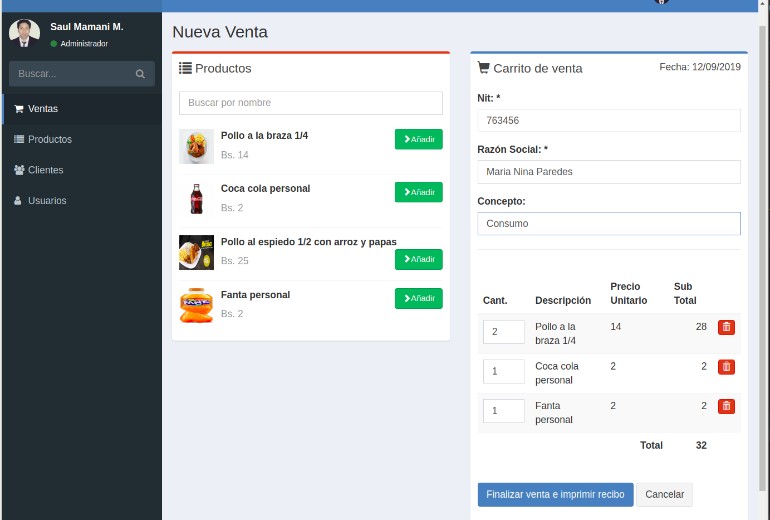
*Ilustración 23: Control de versiones del segundo sprint Fuente: Elaboración propia*

El modelo relacional de la base de datos se ha implementado en base al diagrama de clases persistentes.

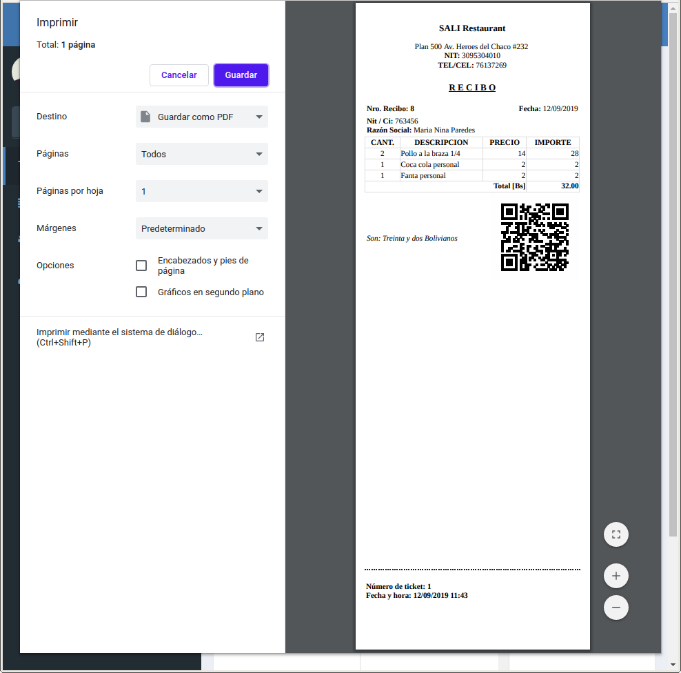
*Ilustración 24: Modelo relacional de la base de datos del segundo sprint*

### Sprint review

El **product increment** representa los entregables realizados al product owner y al cliente, estos entregables reflejan el cumplimiento del segundo sprint.

Se ha desplegado el sistema en la siguiente dirección URL, a modo de prueba para su revisión: [https://ctrlfood.herokuapp.com](https://ctrlfood.herokuapp.com/)

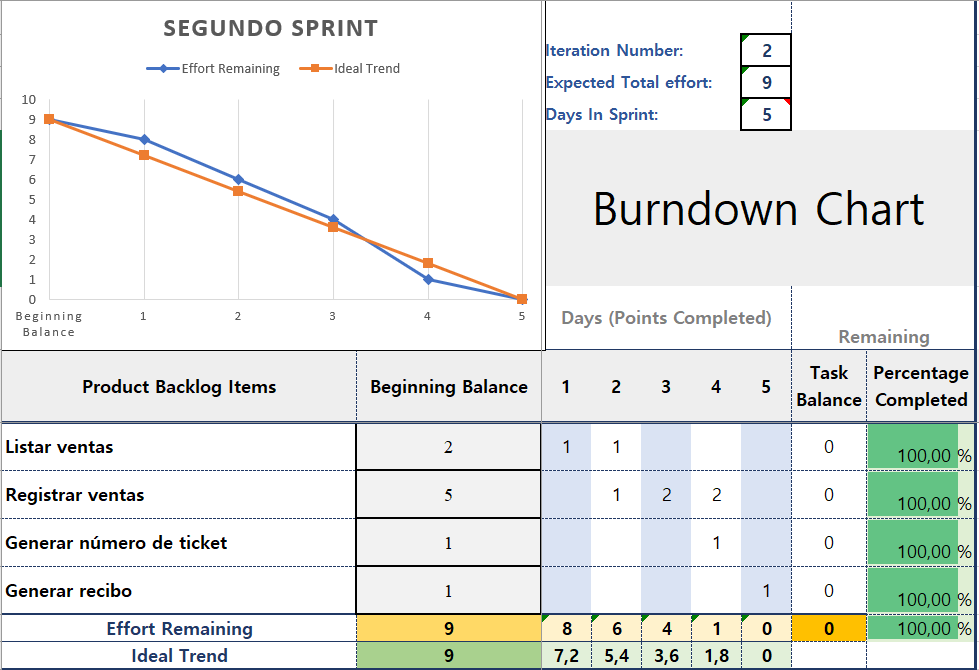
*Ilustración 25: Pantalla registrar ventas Fuente: Elaboración propia*



*Ilustración 26: Pantalla generar recibo*

### Sprint retrospective

El desarrollo del segundo sprint se concluyó de manera adecuada cumpliendo las tareas, los tiempos establecidos, y la definición de hecho establecido al inicio del sprint. Esta evolución se puede apreciar en el gráfico burndown.



*Ilustración 27: Seguimiento de tareas segundo sprint Fuente: Elaboración propia*

*Ilustración 28: Burndown chart segundo sprint*

Segundo sprint

10

8

8

8

6

6

6,4

4

4

4,8

3

2

2 3,2

0

0

Beginning Balance

1

2

3

1,6

4

4,44089E5-16

Days

Effort Remaining

Ideal Trend Lineal (Effort Remaining)

Story Points

### Tercer sprint

* + 1. **Sprint planning**

El sprint planning muestra **qué** es lo que se tiene que desarrollar en esta iteración, y los tiempos comprometidos para finalizar cada historia de usuario.

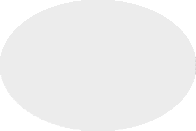
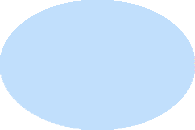
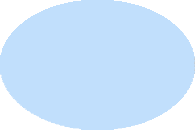
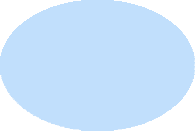
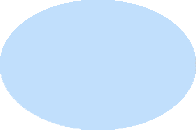
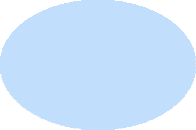
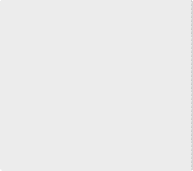
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **HU** | **Descripción** | **Puntos**  **de historia** | **Tiempo [Días]** | **Inicio** | **Finalización** |
| **HU12** | Anular ventas | 1 | ½ | 16/09/2019 | 16/09/2019 |
| **HU13** | Restablecer ventas | 1 | ½ | 16/09/2019 | 16/09/2019 |
| **HU14** | Generar informe económico | 3 | 2 | 17/09/2019 | 18/09/2019 |
| **HU15** | Generar reportes  estadísticos | 3 | 2 | 19/09/2019 | 20/09/2019 |
| **Puntos historia/Tiempo estimado**  **(Time Boxing)** | | **8** | **5** |  | |

*Tabla 34: Tercer sprint planning*

Para completar este sprint se estima un esfuerzo de 8 puntos de historia y tiempo de desarrollo de cinco días.

### Diagrama de casos de uso

El diagrama de casos de uso muestra la funcionalidad global del sprint.



ctrlFOOD - Tercer Sprint

«extend»

*(from Segundo Sprint)*

**Anular ventas**

**Vendedor**

***Generar reportes***

«extend»

**Restablecer ventas**

**Informes económicos**

«include»

**Administrador**

*(from Primer Sprint)*

**Reportes estadísticos**

**Generar número de**

**ticket**

*(from Segundo Sprint)*

*extension points* imprimir recibo anular ventas restablecer ventas

**Listar ventas**



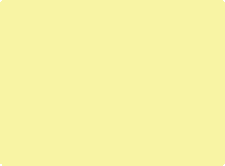
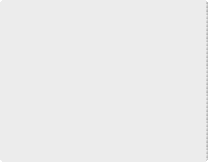
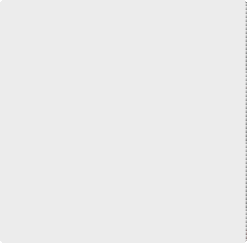
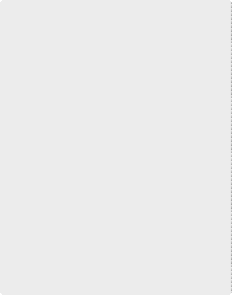
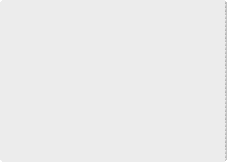
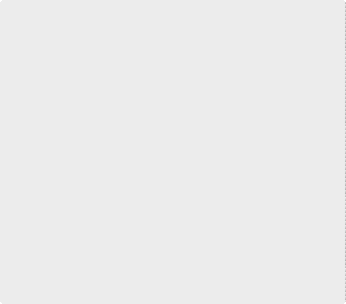
*(from*

*Primer Sprint)*

*Ilustración 29: Diagrama de casos de uso tercer sprint*

### Diagrama de clases persistentes

Se emplea el diagrama de clases para modelar el esquema lógico de la base de datos.



«derive»

1

+Vendedor

\*

«dataType»

**Categoria**

-

-

Bebida: char

Comida: char

0..1

1

«derive»

1

«derive»

**Product**

1..\*

-

-

-

fotografia: char

nombre: char precio: float

«enum»

categoria: Categoria

**Detail**

«dataType»

**Estado**

1

-

1 -

cantidad: int

precio: float

-

-

Activo: boolean

Anulado: boolean

-

* nit: int
* razon\_social: char

**Client**

* concepto: char
* estado: Estado
* fecha: date
* nit: int
* numero: int
* numero\_ticket: int
* razon\_social: char

**Sale**

* Administrador: char
* Vendedor: char
* email: char
* email\_verified\_at: timestamps
* fotografia: char
* name: char
* password: char

«enum»

* rol: Rol

«dataType»

**Rol**

**User**

*Ilustración 30: Diagrama de clases tercer sprint Fuente: Elaboración propia*

### Especificación de casos de uso

Para la especificación de los casos de uso utilizamos historias de usuario detallados.

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Anular ventas | **Historia de usuario:** HU12 | | |
|  | **Prioridad:** Baja | **Depende de:** HU8 | |
| **Descripción:** Anulación de ventas erróneas | | |
| **Criterio de Aceptación:**   * El sistema muestra la lista de ventas realizadas de la fecha actual, con el botón para Anular * El actor pulsa el botón Anular * El sistema cambia el estado de la venta en la base de datos y recarga la lista.   o Las ventas anuladas no son contabilizadas tomadas en cuenta para el reporte económico. | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 35: Caso de uso Anular ventas*

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Restablecer ventas | **Historia de usuario:** HU13 | | |
|  | **Prioridad:** Baja | **Depende de:** HU12 | |
| **Descripción:** Restablece ventas anuladas por error | | |
| **Criterio de Aceptación:**   * El sistema muestra la lista de ventas anuladas de la fecha actual. * El actor pulsa el botón Restablecer * El sistema cambia el estado de la venta en la base de datos y recarga la lista. | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 36: Caso de uso Restablecer ventas*

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** Generar reportes | **Historia de usuario:** HU14, HU15 | | |
|  | **Prioridad:** Baja | **Depende de:** | |
| **Descripción:** Generar informes y reportes estadísticos | | |
| **Criterio de Aceptación:**  **Reporte económico**   * En la lista de ventas (HU8) el sistema muestra un botón para imprimir el reporte, y el actor pulsa el botón. * El sistema genera el reporte económico en PDF.   **Reporte estadístico**   * El actor pulsa el botón Reportes del menú principal * El sistema genera dos reportes estadísticos:   + Gráfica del total de las ventas realizadas agrupadas por mes   + Lista de clientes potenciales (clientes con más compras realizadas) | | |
| **Elaborado por:**  Saul Mamani | **Responsable:**  Saul Mamani | | **Versión:**  1.0 |

*Tabla 37: Caso de uso Generar reportes*

### Sprint backlog

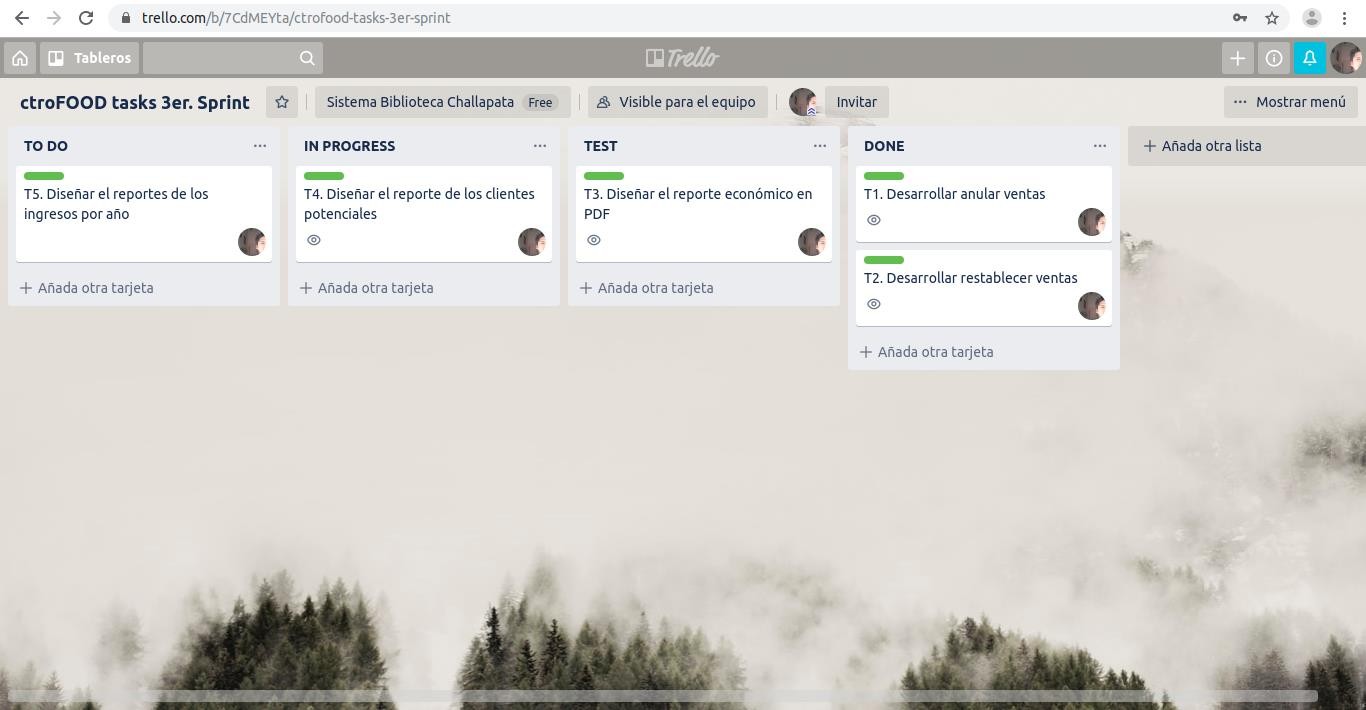
Para determinar **cómo** se va a desarrollar el sprint, se identifican una o varias tareas por historia de usuario, las cuales son agrupadas en un sprint backlog y asignadas a los miembros del equipo de desarrollo.

|  |  |  |
| --- | --- | --- |
| **HU** | **Descripción** | **Tareas** |
| **HU12** | Anular ventas | T1. Desarrollar anular ventas |
| **HU13** | Restablecer ventas | T2. Desarrollar restablecer ventas |
| **HU14** | Generar informe económico | T3. Diseñar el reporte económico en PDF |
| **HU15** | Generar reportes estadísticos | T4. Diseñar el reporte de los clientes potenciales  T5. Diseñar el reporte de ingresos económicos por año |

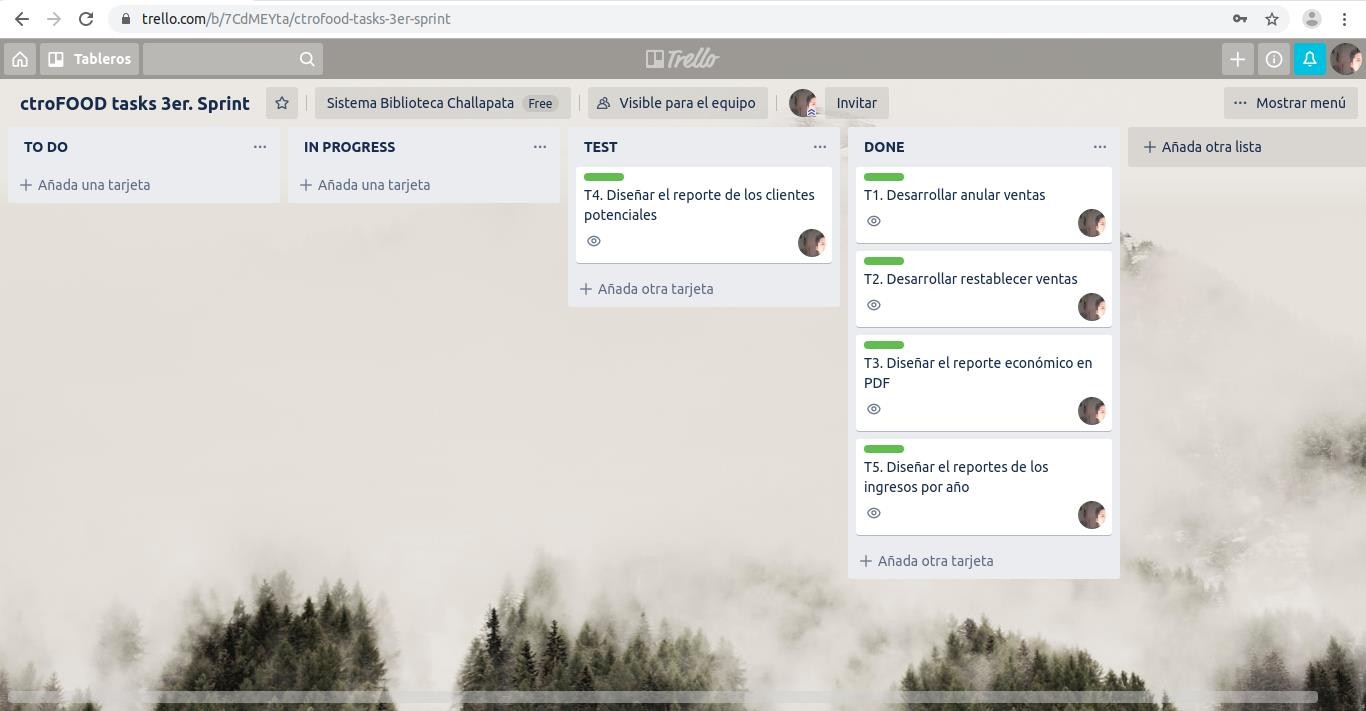
*Tabla 38: Tercer sprint backlog*

### Sprint execution

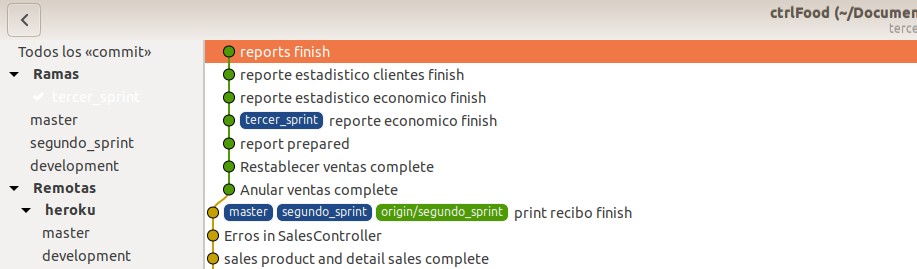
Para controlar la ejecución del sprint y el avance de las tareas, empleamos un tablero Kanban con la herramienta Trello.



*Ilustración 31: Tablero Kanban tercer sprint Fuente: Elaboración propia*

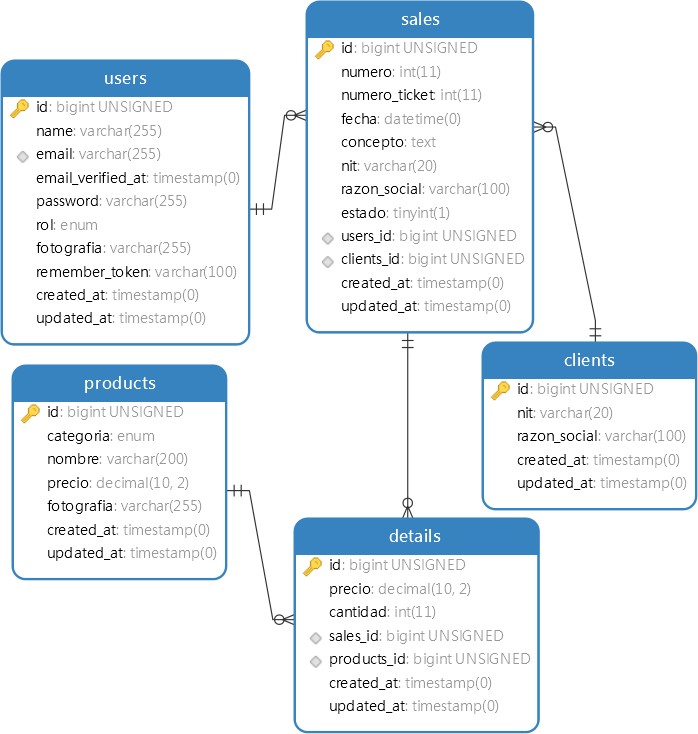


*Ilustración 32: Tablero Kanban tercer sprint Fuente: Elaboración propia*

Para controlar el avance del código fuente se utiliza el sistema de control de versiones GIT, el cual se encuentra en GitHub bajo en el siguiente repositorio: [http://github.com/saulmamani/ctrlFood](http://github.com/saulmamani/ctrlfood)

*Ilustración 33: Control de versiones del tercer sprint Fuente: Elaboración propia*

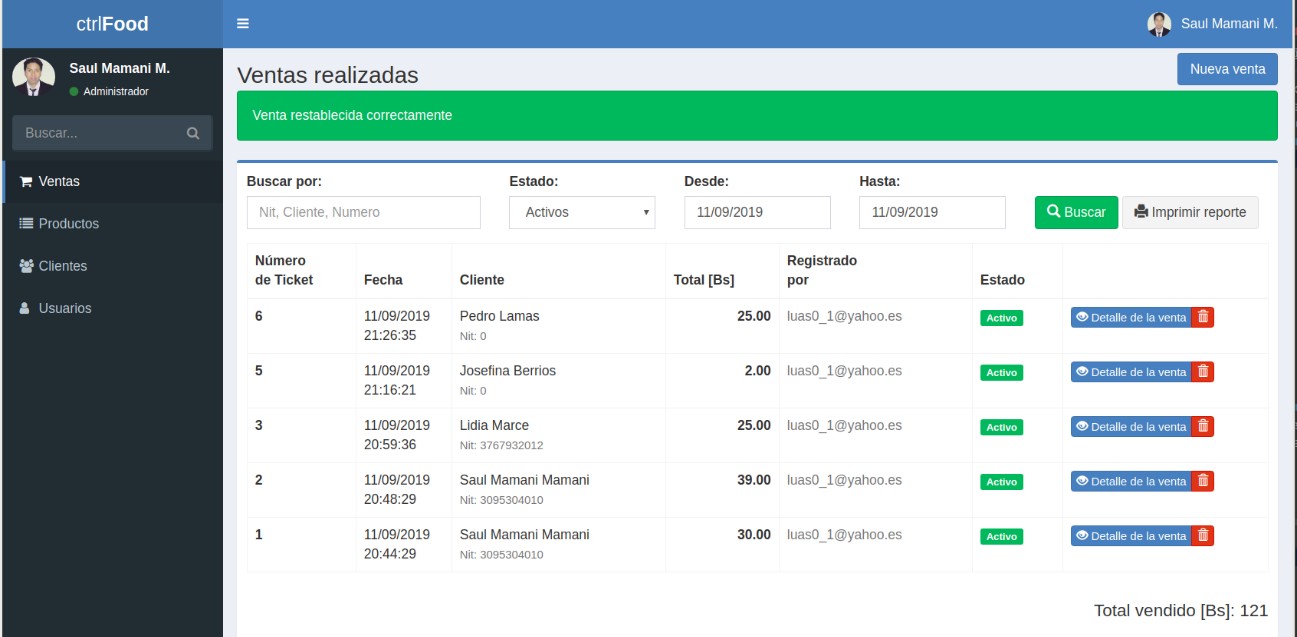
El modelo relacional de la base de datos que se ha implementado en base al diagrama de clases persistentes.

Se puede observar que no ha sufrido cambios desde el sprint anterior, por lo tanto, se concluye que el base de datos fue concluido en el segundo sprint.

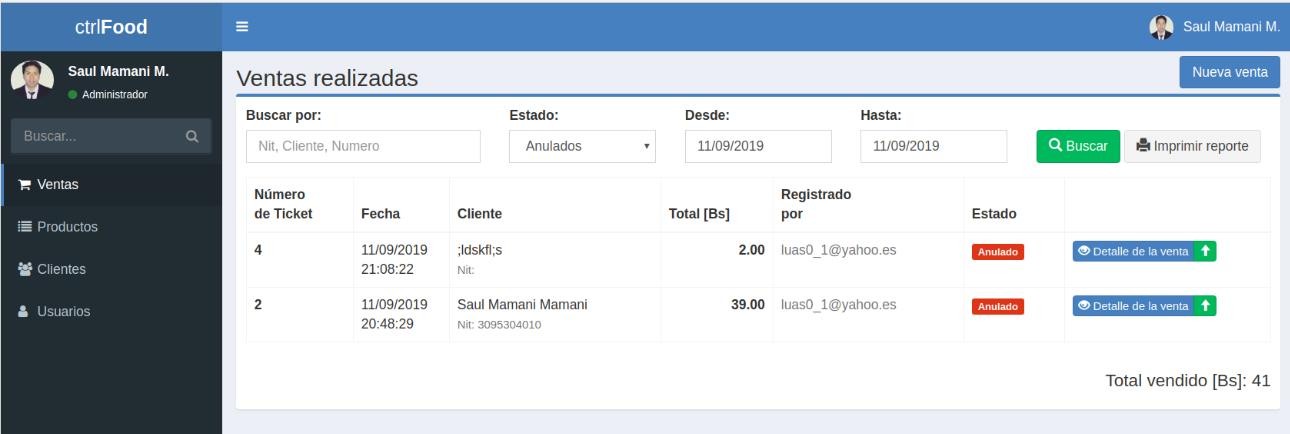
*Ilustración 34: Modelo relacional de la base de datos del tercer sprint Fuente: Elaboración propia*

### Sprint review

El **product increment** representa los entregables realizados al product owner y al cliente, estos entregables reflejan el cumplimiento del tercer sprint.

Se ha desplegado el sistema en la siguiente dirección URL, a modo de prueba para su revisión: [https://ctrlfood.herokuapp.com](https://ctrlfood.herokuapp.com/)

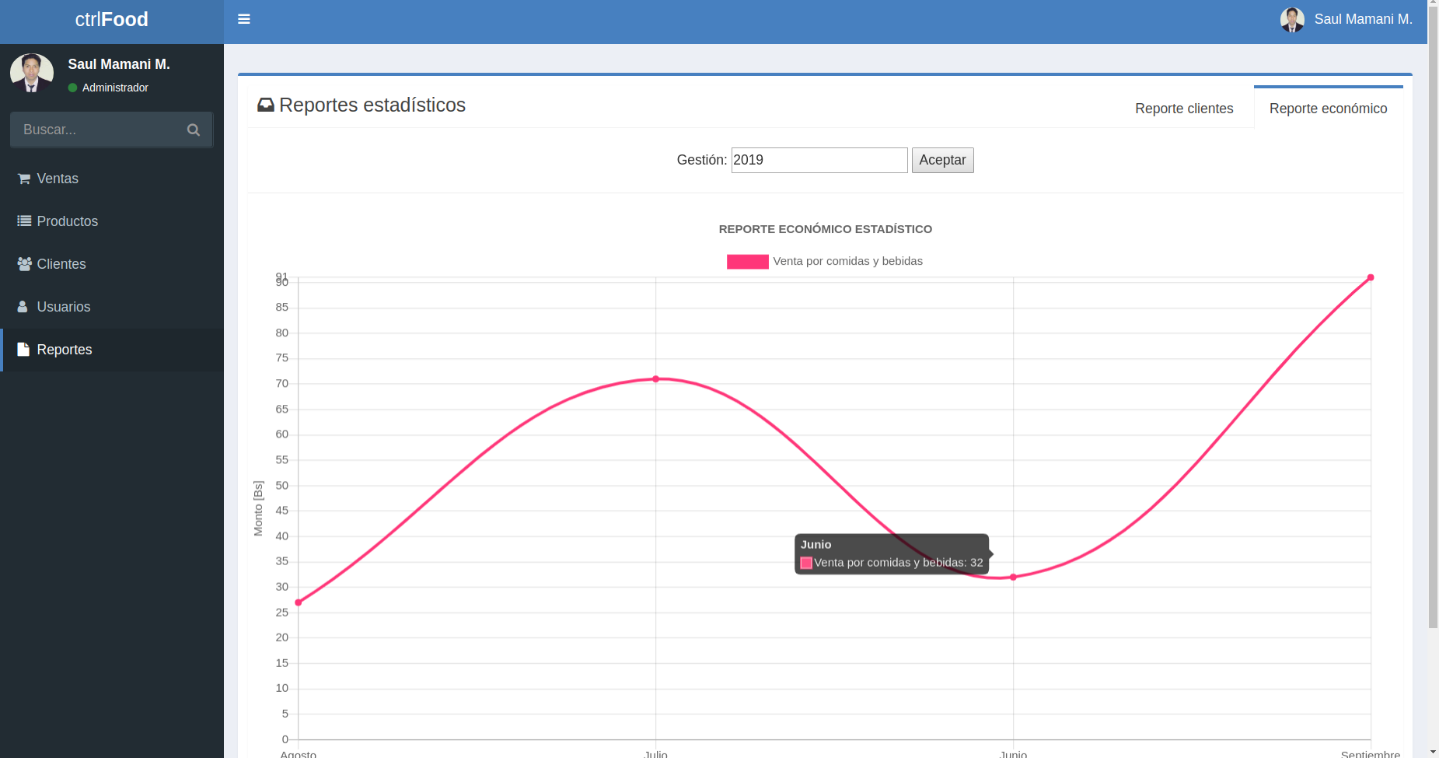
*Ilustración 35: Pantalla restablecer ventas Fuente: Elaboración propia*



*Ilustración 36: Pantalla ventas anuladas Fuente: Elaboración propia*



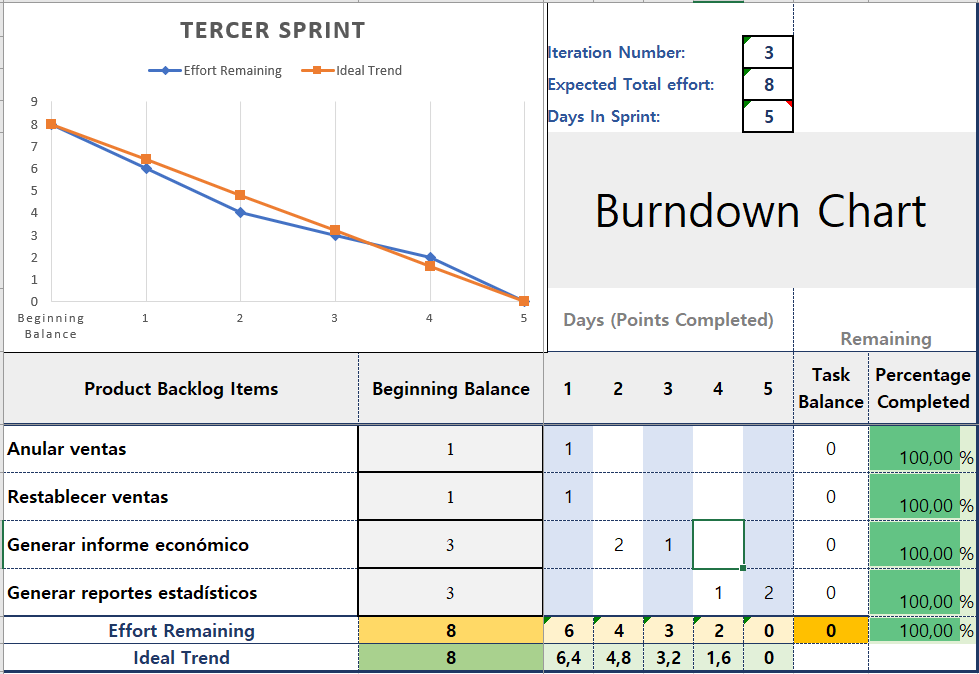
*Ilustración 37: Pantalla reporte de ventas Fuente: Elaboración propia*



*Ilustración 38: Pantalla reporte estadístico Fuente: Elaboración propia*

### Sprint retrospective

El desarrollo del tercer sprint se concluyó de manera adecuada cumpliendo las tareas, los tiempos establecidos, y la definición de hecho establecido al inicio del sprint. Esta evolución se puede apreciar en el gráfico burndown.



*Ilustración 39: Seguimiento de tareas tercer sprint Fuente: Elaboración propia*

*Ilustración 40: Burndown chart tercer sprint*

Tercer sprint

10 8

8

6 8

4

2

0

Beginning Balance

6

6,4

4

3

4,8

2

3,2

0

1

2

3

1,6

4

4,44089E5-16

Days

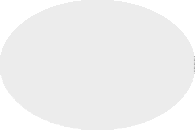
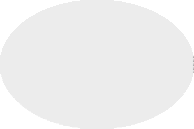
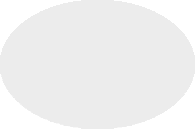
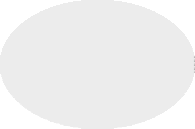
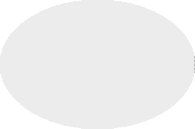
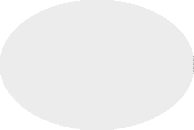
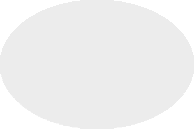
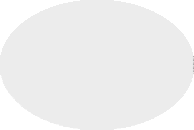
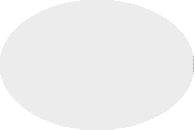
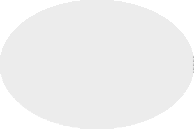
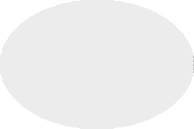
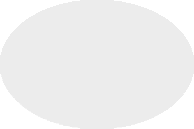
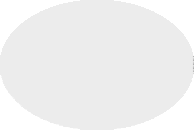
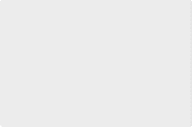
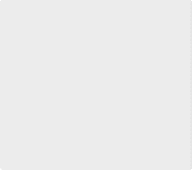
Effort Remaining

Ideal Trend Lineal (Effort Remaining)

Story Points

### Diagrama general de casos de uso

Se puede ver la funcionalidad global del sistema, y su interacción con los actores, en el siguiente diagrama de casos de uso.



ctrlFOOD

«login»

**Iniciar Sesión**

**Administrar Perfil**

*(from Primer Sprint)*

«CRUD»

**Gestionar clientes**

*(from Primer Sprint)*

«extend»

*(from Primer Sprint)*

«extend»

**Cambiar contraseña**

**Vendedor**

*(from Primer Sprint)*

«include»

*(from Segundo Sprint)*

**Generar número de**

**ticket**

«include»

*(from Segundo Sprint)*

«extend»

*(from Segundo Sprint)*

**Generar recibo**

«extend»

*(from Segundo Sprint)*

«include»

«CRUD»

**Gestionar productos**

«extend»

*(from Primer Sprint)*

**Restablecer ventas**

**Anular ventas**

**Informes económicos**

*(from Tercer Sprint)*

*(from Tercer Sprint)*

*(from Tercer Sprint)*

«CRUD»

**Gestionar usuarios**

***Generar reportes***

*(from Primer Sprint)*

*(from Tercer Sprint)*

**Administrador**

*(from Primer Sprint)*

**Reportes estadísticos**

*(from Tercer Sprint)*

*extension points* imprimir recibo anular ventas restablecer ventas

**Listar ventas**

*extension points*

Nuevo cliente

**Registrar ventas**



*(from*

*Primer Sprint)*

*Ilustración 41: Diagrama general de casos de uso*

### Diagrama de despliegue

El diagrama de despliegue muestra la arquitectura física de red y los servidores donde el software será puesto en producción, además de, las terminales de los clientes que van a utilizar el sistema.



**deployment Diagrama de despliegue**

**Apache2: Web Server**

tcp/ip

1

1

tcp/ip

1

1

tcp/ip

tcp/ip

1

\*

**Administrador: PC**

**Vendedor: PC**

«device»

**router**

**MySQL 10.4.8: Data Base Server**

**ctrlFOOD**

*Ilustración 42: Diagrama de despliegue Fuente: Elaboración propia*

## CONCLUSIONES Y RECOMENDACIONES

Después de haber culminado las diferentes etapas del desarrollo de la monografía, se puede llegar a las siguientes conclusiones y realizar las algunas recomendaciones:

### Conclusiones

Es posible trabajar con una metodología ágil como scrum en combinación con una notación de modelado visual como UML, sin dañar el valor del manifiesto ágil que dicta: “Software funcionando sobre documentación exhaustiva”, utilizando solamente los diagramas necesarios que ayuden a comprender mejor la funcionalidad, las interacciones, y la lógica del sistema.

UML, al ser una notación de modelado visual, mejora la comprensión del problema y el objetivo de las iteraciones, además, mejora la comunicación entre los diferentes miembros del equipo scrum y los stackeholders, puesto que con una notación visual se puede observar mejor los distintos niveles de abstracción de un sistema.

Se han cumplido también las tareas planteadas al principio del proyecto. [Pág. [3](#_bookmark4)]

* Con la ayuda del product vision board se logró comprender la visión del producto que fue desarrollado como ejemplo para la monografía.
* El ámbito del sistema fue determinado con las historias de usuario escritas por el product owner.
* Se ha cumplido con la planificación descrita en el product backlog, siguiendo los plazos estimados dentro las iteraciones y las entregas establecidas en el despliegue.
* Se han modelado diagramas de casos de uso, para tener una visión global de la funcionalidad dentro de cada iteración o de cada sprint, y diagramas de clases para representar la lógica de la base de datos. Además, se ha modelado el diagrama de componentes y el diagrama de despliegue para mostrar la arquitectura del sistema, la arquitectura física, los servidores, y la tecnología que se necesita para la puesta en producción del sistema.
* Para controlar el avance de las tareas identificadas en el sprint backlog de cada iteración, se ha utilizado un tablero virtual Kanban, apoyándonos en la herramienta online Trello.
* Se tiene un historial completo de todos los cambios realizados al código del sistema con la herramienta de control de versiones Git.
* El sistema fue desarrollado de acuerdo a lo planificado y se encuentra desplegado Heroku como un servidor de pruebas. [https://ctrlfood.herokuapp.com](https://ctrlfood.herokuapp.com/)

### Recomendaciones

Si se va utilizar una notación de modelado visual como UML con una metodología de desarrollo ágil, se recomienda aplicar solo los diagramas que sean estrictamente necesarios para resolver y comprender mejor el problema; de esta forma no dañará los valores y los principios del manifiesto ágil.

## BIBLIOGRAFÍA

* Conallen, J. (2011). *Building Web Applications with UML.* Boston: Addinson-Wesley.
* Group, O. M. (13 de 09 de 2019). *omg*. Obtenido de omg: https://[www.omg.org/spec/UML/About-UML/](http://www.omg.org/spec/UML/About-UML/)
* Hiromoto, H. (31 de 12 de 2017). *scrumorganico*. Obtenido de scrumorganico: https://medium.com/scrumorganico
* Manifesto, A. (13 de 09 de 2019). *agilemanifesto*. Obtenido de agilemanifesto: https://agilemanifesto.org/iso/es/manifesto.html
* Mercurial, A. (5 de 9 de 2019). *Agile Mercurial*. Obtenido de Agile Mercurial: https://agile- mercurial.com/excel-templates/burndown-chart/
* Moreno, G. R. (2004). *UML con Rational Rose.* Lima: MEGABYTE SAC.
* Rubin, K. S. (2013). *Essential Scrum.* Boston: Addison Wesley.
* Rumbaugh, J., Jacobson, I., & Booch, G. (2007). *El Lenguaje de Modelado Unificado 2.0.*

Madrid: Perarson Addison Wesley.

* Rumbaugh, J., Jacobson, I., & Booch, G. (2007). *El lenguaje de Modelado Unificado Manual de Referencia.* Madrid: PEARSON Addison Wesley.
* Saiz, J. (2018 de 09 de 13). *js*. Obtenido de js: [http://jorgesaiz.com](http://jorgesaiz.com/)
* Sommerville, I. (2005). *Ingenieria de Software.* Madrid: PEARSON Addison Wesley.