**UNIVERSIDAD TECNOLÓGICA PRIVADA DE SANTA CRUZ FACULTAD DE EDUCACIÓN Y TECNOLOGÍA**

**CARRERA: LICENCIATURA EN INGENIERÍA DE SISTEMAS**



**SISTEMA DE GESTIÓN DE PEDIDOS DE PASTELES ARTESANALES PERSONALIZADOS PARA “PASTELES ÚNICOS”**

**DOCENTE:** Ing. Dennis Delgado

**MATERIA:** Desarrollo de sistemas y Bases de Datos

**INTEGRANTES:**

Jheyson Cuellar

Iden Ticlla Choque

**SANTA CRUZ - BOLIVIA 2023**

[**ENUNCIADO DEL CASO DE ESTUDIO 8**](#_9cav6saoubtr)

[**I. ASPECTOS GENERALES 10**](#_rmxxb58l7iy)

[1.1 Titulo del proyecto 10](#_qcqrz8sgmxbe)

[1.2 Introducción 10](#_eoblqvaf73hf)

[1.3 Planteamiento del problema 11](#_599615569h8k)

[1.4 Justificaciones 11](#_pfexxnaejmsg)

[1.4.1 Justificación económica 11](#_mxez6eckmckv)

[1.4.2 Justificación social 12](#_w9ocexklctc)

[1.4.3 Justificación técnica 12](#_d0j6nzv4r4xx)

[1.5 Delimitaciones 12](#_d741zoazba2u)

[1.5.1 Delimitación Temporal 12](#_md5bwi7emgre)

[1.5.2 Delimitación Geográfica 13](#_nu1vchwa8m17)

[1.5.3 Delimitación Científica 13](#_kr8ly0b3hdj)

[1.6 Objetivos 13](#_v7rfj2htnh7x)

[1.6.1 Objetivo General 13](#_sdse7evmlbkf)

[1.6.2 Objetivos específicos 13](#_hgbr8cv7g3q8)

[**II. Marco Teórico 14**](#_6j32nbr2w7hr)

[2.1 El manifiesto ágil 14](#_gn09dhuih38i)

[2.2 Metodología 15](#_ptf3848xgxig)

[2.3 Xtreme Programming (XP) 15](#_3ebi98ktudm8)

[2.3.1 Características de la metodología XP 16](#_y2k1pjci59x7)

[2.4 Fases de XP 17](#_6xxfjl4i6umn)

[2.4.1 Fase de planificación 18](#_mbijepinmw8z)

[2.4.2 Fase de diseño 18](#_p2va6cdx1vlq)

[2.4.3 Fase de codificación 18](#_ygnncupjx7fe)

[2.4.4 Fase de pruebas 19](#_mylctvyxtuzq)

[2.5 Los valores de XP 19](#_vehjh7hzfnin)

[2.6 Prácticas de XP 20](#_nar56xb43o64)

[2.6.1 Juego de la planificación 21](#_sgu8gv22wwjy)

[2.6.2 Entregas pequeñas 22](#_3hgq1k3lg8kc)

[2.6.3 Desarrollo dirigido por pruebas 22](#_ibwx8y47qrh2)

[2.6.4 Refactorización 22](#_1nfzncy5ix5)

[2.6.5 Programación en parejas 22](#_c27pi5pgt3ed)

[2.6.6 Propiedad colectiva del código 23](#_y1t2ybmdr33q)

[2.6.7 Integración continua 23](#_89t6yua6bn9)

[2.6.8 40 horas por semana 23](#_vmxx2z7z1o3u)

[2.6.9 Metáfora 23](#_kr7o4cj5ntjy)

[2.6.10 Diseño simple 23](#_8mhsdzcn8mr8)

[2.6.11 Cliente in-situ 23](#_r61ht13xirv5)

[2.6.12 Estandares de programacion 24](#_r41znujbnx95)

[2.7 Ciclo de vida de XP 24](#_wj0k0ohphoc2)

[2.7.1 Fase de exploración 24](#_wbcwmuptoe9v)

[2.7.2 Fase de planificación de la entrega 24](#_wd49u9v18dzs)

[2.7.3 Fase de Iteraciones 24](#_t5malu8slltw)

[2.7.4 Fase de producción 25](#_puo0mko4bjiv)

[2.7.5 Fase de mantenimiento 25](#_tde7may57ctt)

[2.7.6 Fase de muerte del proyecto 25](#_upnpdanbguxr)

[2.8 Roles de XP 26](#_2h2r9gnzplxn)

[2.8.1 Clientes 26](#_4nu0vkj5b2wo)

[2.8.2 Programadores 26](#_wm3g5v330jma)

[2.8.3 Testers 26](#_rfrrhoz19nlw)

[2.8.4 Coach 27](#_vhhgy7bkllc5)

[2.8.5 Manager 27](#_csy0jdrblv7d)

[2.9 Marco de trabajo SCRUM 27](#_2mb49effp6vt)

[2.9.1 Proceso 28](#_9h4ggoseywyk)

[2.9.1.1 Planificación de la iteracion 28](#_2k8byewhj2l8)

[2.9.1.2 Ejecucion de la iteracion 29](#_vipb3b8lg1b)

[2.9.1.3 Inspeccion y adaptacion 30](#_f0xgo19iajw3)

[2.9.2 Roles de SCRUM 30](#_dmsdjqq0jx64)

[2.9.2.1 Product owner 30](#_p3rap72l2tm9)

[2.9.2.2 Scrum Master 31](#_aojklyr5zbb)

[2.9.2.3 Equipo de desarrollo 31](#_noyjsgha85k9)

[2.9.3 Hitos de SCRUM 31](#_pbbte57hl3s0)

[2.9.3.1 Sprint 31](#_3rib28f4nm04)

[2.9.3.2 Sprint planning 32](#_2q274djocgdv)

[2.9.3.3 Daily meeting 32](#_hdie4m668ej)

[2.9.3.4 Sprint review 33](#_9vavcymxgn1r)

[2.9.3.5 Sprint retrospective 33](#_pt81iyyo911i)

[2.9.4 Herramientas de SCRUM 34](#_jxkumwn51jjd)

[2.9.4.1 Product backlog 34](#_ffljcwrdzgz9)

[2.9.4.2 Sprint backlog 34](#_qih106fy3m2v)

[2.10 Historias de usuario 35](#_jmfn0egysebu)

[2.10.1 Elementos de una historia de usuario 35](#_iaklsqgzsoss)

[2.10.1.1 Identificador (ID) de la historia: 35](#_nv3g2ans79fg)

[2.10.1.2 Rol 35](#_dj8jno2ev5vb)

[2.10.1.3 Característica / Funcionalidad (Feature) 36](#_vc0hv7l4x4c0)

[2.10.1.4 Razón / Resultados 36](#_vj72vfej4jix)

[2.11 El dueño del producto (Product Owner) 36](#_t9r30jluwmwv)

[2.12 El Scrum Master 37](#_u1w6e16gymdr)

[2.13 Lista de producto (Product backlog) 38](#_h25jvcz5dmuk)

[2.14 Planning Poker 38](#_756jd4ysr63h)

[2.15 Arquitectura del sistema 39](#_y8u04vam0lym)

[2.15.1 MVC (Modelo Vista Controlador) 40](#_gynyu6mmwo16)

[2.16 Lenguajes 41](#_zfgzcuev66j9)

[2.16.1 Lenguaje de bajo nivel 41](#_sm3k02a47x7r)

[2.16.2 Lenguaje de alto nivel 41](#_r64zomdxqrco)

[2.16.3 Python 41](#_wati32mhmafc)

[2.16.4 Javascript 42](#_wqo8xxhz7i0x)

[2.17 Frameworks 42](#_o9ncjrj3ryv4)

[2.17.1 Django 43](#_agrrb71yrze1)

[2.17.2 Bootstrap 43](#_kiiofpounx5e)

[2.18 Base de datos 44](#_mj3wldio4c7c)

[2.18.1 Mysql 44](#_y617dzcyublh)

[2.19 IDE 44](#_6qss5tock5wg)

[2.19.1 Visual Studio Code 44](#_j6cjaw50p7fr)

[**III. Ingeniería del proyecto 45**](#_p32q2aauu3r)

[3.1 Fase de exploración 45](#_unnokmowmmzo)

[3.1.1 Requisitos no funcionales 45](#_5psjvko4bga)

[3.1.2 Diagrama de clases conceptuales 46](#_yk7kznmafu4b)

[3.1.3 Product Backlog - Historias de usuario 47](#_iygw7z6p9t7a)

[3.2 Fase de planificación de la entrega (Release Plan) 51](#_o8ehvz9wbo2y)

[3.2.1 Priorización de las historias de usuario 51](#_ipn2zioya2l5)

[3.2.2 Estimación de esfuerzo por puntos 52](#_5dqky87769g5)

[3.3 Fase de Iteraciones: (para cada sprint) 53](#_flplqvm54q24)

[3.3.1 Sprint 1 : Gestionar Productos - Gestionar Clientes - Registrar Quejas 53](#_nuzsnapml21m)

[3.3.1.1 Sprint Backlog 53](#_pgwffjj3barz)

[3.3.1.2 Burn Down Chart 53](#_bh4882e5o94c)

[3.3.1.3 Pruebas de aceptación 53](#_61ze8htx9600)

[3.3.1.4 Incremento 53](#_bdjt5sti2dxp)

[3.3.2 Sprint 2 : Realizar Pedido - Ver Listado de pedidos - Ver pedido 54](#_eiuhwypmts80)

[3.3.2.1 Sprint Backlog 54](#_mwjvu53h609c)

[3.3.2.2 Burn Down Chart 54](#_jeeu0yyaxt78)

[3.3.2.3 Pruebas de aceptación 54](#_do7gc52l0vkd)

[3.3.2.4 Incremento 54](#_bo6h2ia02433)

[3.3.3 Sprint 3 : Gestionar venta 55](#_r3agrv109nyl)

[3.3.3.1 Sprint Backlog 55](#_5xj67rancgf2)

[3.3.3.2 Burn Down Chart 55](#_1va7qlt83an7)

[3.3.3.3 Prueba de aceptacion 55](#_lt8gcw71z5dl)

[3.3.3.4 Incremento 55](#_53deszr5syh9)

[3.4 Fase de producción: 56](#_4owem8mn16xj)

[3.4.1 Diagrama de clases de diseño 56](#_p1ece6j58ypz)

[3.4.2 Normalización de la base de datos 56](#_1469irpf4gja)

[3.4.3 Modelo de datos relacional (modelo de base de datos lógico/físico) 56](#_1qjoaiz3vsip)

[3.4.4 Diseño de reportes 56](#_5nv4lcllv1cx)

[3.4.5 Triggers 58](#_xy6nbdp1moln)

[3.4.6 Consultas más complejas 59](#_juy1vlnun0rd)

[3.4.7 Procedimientos almacenados 60](#_ktlcaktxuyr2)

[3.5 Fase de mantenimiento 61](#_vebi5qqatjrp)

[3.5.1 Plan de backup de base de datos 61](#_7amzen6tcvim)

[3.6 Fase de muerte del proyecto 61](#_wd6h6g5czzyq)

[3.6.1 Rendimiento del sistema 61](#_z8rfgo3q98j3)

[3.6.2 Confiabilidad del sistema 61](#_jnaenblpt4i6)

[**IV. CONCLUSIONES 62**](#_5qzjg5mso3jd)

[**V. RECOMENDACIONES 62**](#_z7k70kw146ov)

[**VI. BIBLIOGRAFÍA 62**](#_gsnfw0zayod9)

[**VII. ANEXOS 63**](#_y9t5403m5wj5)

# ENUNCIADO DEL CASO DE ESTUDIO

**SISTEMA DE GESTIÓN DE PEDIDOS DE PASTELES ARTESANALES PERSONALIZADOS PARA “PASTELES ÚNICOS”**

**SITUACIÓN PROBLEMÁTICA**

PASTELES UNICOS, se caracteriza por crear diseños personalizados, de todo tipo de pasteles, los cuales son realizados artesanalmente.

El mercado de pasteles personalizados ha crecido mucho en los últimos 3 años, ahora, los clientes quieren que sus pasteles sean decorados con un tema que los identifique plenamente, por este motivo, eligen el modelo, ya sea de revistas o en la mayoría de las veces de internet.

PASTELES UNICOS, realiza los pedidos, con el diseño exacto que solicita el cliente, además de tener muy buen sabor y ser frescos, es decir, horneados la una noche antes de la entrega.

El problema con el que tiene que lidiar, es con la gestión de pedidos, se realizan tantos pedidos por día, que ya es evidente que se requiere un Sistema Informático de Gestión de Pedidos.

**LÓGICA DE NEGOCIO**

Para la Gerencia, es importante que los pedidos no se confundan, por tanto, se debe registrar el nombre del cliente, la fecha de pedido, la fecha de entrega del pedido, capturar la imagen del tema elegido para que sean exactamente con los colores que solicita, la imagen o imágenes, el tipo de pasteles que requiere, torta, cup cakes, galletas grandes, pequeñas, medianas, alfajores, tamaño, sabores y tipo de chocolate que lo cubra, cakes pops, brownies, entre otros pasteles que se pueden realizar a pedido.

Los diseños que van sobre los pasteles, no son realizados en papel, al contrario, son hechos a mano, en un tipo de pasta comestible, por tanto, el pastel es el mismo diseño elegido, por ejemplo, si se eligió el tema de “granja de animales”, las galletas tienen forma de vacas, chanchos, pollos, todo esto realizado de manera tan perfecta que parecen reales y es por eso el éxito de la pastelería.

Una vez que el cliente, fue hasta la pastelería, hizo el pedido, dejó la o las imágenes del tema de su preferencia, debe cancelar el 80% del monto, el 20% restante, al momento de la entrega.

No se hacen entregas a domicilio, por tanto, en la fecha fijada, el cliente debe pasar a recoger su pedido.

El Gerente necesita varios reportes, entre ellos, la cantidad de pedidos, por tipo de pastel que tienen por semana y por mes.

# I. ASPECTOS GENERALES

## 1.1 Titulo del proyecto

Sistema de gestión de pedidos de pasteles artesanales personalizados para “pasteles únicos”.

## 1.2 Introducción

El presente trabajo fue realizado para tener una mejor gestión de los pedidos de la empresa “Pasteles Únicos”, así poder facilitar la administración de la información de sus clientes y los requisitos de sus pedidos.

Se realizará utilizando la tecnología a nuestro alcance modernizando así su forma de administrar la información, con estas herramientas se pretende desarrollar un sistema de gestión fácil y moderno para la manipulación de sus pedidos con una interfaz sencilla e interactivas con los usuarios.

Como metodología a elegir se usará el método XP (Extreme Programming) que es una metodología cuyo objetivo es crear sistemas de alta calidad, basados en una estrecha interacción con los clientes, pruebas constantes y ciclos de desarrollo cortos.

El trabajo será realizado actualmente solo la empresa que se encuentra en la ciudad de Santa Cruz De La Sierra, pero a lo largo de su crecimiento podrá ser aplicado a las diferentes posibles sucursales en el departamento de Santa Cruz.

## 1.3 Planteamiento del problema

PASTELES UNICOS, se caracteriza por crear diseños personalizados, de todo tipo de pasteles, los cuales son realizados artesanalmente.

El mercado de pasteles personalizados ha crecido mucho en los últimos 3 años, ahora, los clientes quieren que sus pasteles sean decorados con un tema que los identifique plenamente, por este motivo, eligen el modelo ya sea de revistas o en la mayoría de la veces de internet

El problema con el que tiene que lidiar, es con la gestión de pedidos, se realizan tantos pedidos por día, que ya es evidente que se requiere un sistema moderno la cual pueda facilitar el registro de pedidos y sus clientes que lo solicitan.

Otro problema secundario a resolver es que la empresa necesita saber la valoración de los clientes con el servicio y atención de la empresa para poder en base a ello mejorar su calidad de producto y servicio.

## 1.4 Justificaciones

### 1.4.1 Justificación económica

El presente proyecto pretende realizar un sistema moderno que registra todas las transacciones monetarias realizadas por los clientes lo que permitirá una mejor gestión y control de los recursos económicos a corto y largo plazo así, facilitar cualquier auditoría anual y en cada cambio de administración.

### 1.4.2 Justificación social

Mejorar la accesibilidad y la comodidad del servicio para los clientes, lo que podría mejorar su preferencia. La implementación de un sistema de gestión de pedidos, podría mejorar la experiencia de los clientes y fomentar la fidelización de los mismos, lo que podría fortalecer la relación entre la pastelería y los clientes, y mejorar su imagen en la comunidad.

### 1.4.3 Justificación técnica

Una mejor gestión del servicio de pedidos que ofrece, así como una mejor interacción con los clientes, lo que podría mejorar la calidad del servicio. La implementación de un sistema de control de pedidos y peticiones detalladamente da un mejor control y seguimiento de los clientes y sus pedidos, lo que podría mejorar la gestión de los recursos.

## 1.5 Delimitaciones

### 1.5.1 Delimitación Temporal

El desarrollo del sistema web de gestión de pedidos para la empresa Pedidos Únicos, será desarrollado en un periodo de tiempo de una semana, la cual se trabajará en un lapso de 4 horas por día, durante los 6 días de la semana, teniendo en cuenta que se está cursando la modalidad de examen de grado

|  |  |
| --- | --- |
| fecha inicio | fecha final |
| 24/02/23 | 03/03/23 |

También se estima que el software completamente desarrollado requiere un alrededor de 6 meses de trabajo para poder implementarlo completamente teniendo en cuenta las posibles modificaciones y exigencias futuras del cliente

|  |  |
| --- | --- |
| fecha inicio | fecha final |
| 24/02/23 | 24/08/23 |

### 1.5.2 Delimitación Geográfica

La presente investigación se delimita geográficamente en la ciudad de Santa Cruz dentro de la provincia Andrés Ibáñez ya que la empresa “Pasteles Únicos” se encuentra en la ciudad de Santa Cruz de la Sierra sobre el 4to anillo entre la av. Beni y la av. principal.

### 1.5.3 Delimitación Científica

El contenido de la presente investigación, orientará su desarrollo al área de programación, específicamente al área de desarrollo aplicaciones web usando la herramienta Django y como Gestor de base de datos MySql.

## 1.6 Objetivos

### 1.6.1 Objetivo General

Desarrollar una aplicación web con el lenguaje de programación Python y utilizando su Framework de desarrollo web Django tomando como caso de estudio la empresa de “Pasteles únicos”.

### 1.6.2 Objetivos específicos

* Identificar las historias de usuario requeridas en el sistema.
* Diseñar la base de datos.
* Diseñar los módulos que serán desarrollados.
* Implementar la base de datos.
* Desarrollar los controladores para consumir los datos de DB.
* Desarrollar la interfaz del sistema usando html, css y javascript.

# II. Marco Teórico

## 2.1 El manifiesto ágil

El Agile Manifiesto para el desarrollo de software ágil es una proclamación de cuatro reglas vitales y doce principios que sirven de introducción y guía para las personas en el software de gestión ágil. Aplicado al desarrollo de software, esta metodología se centra en mantener el código simple e ir testeando los códigos que el propio software vaya sacando. Este nuevo modelo se inventó para agilizar la gestión de proyectos y llegar a sustituir al modelo en cascada (Waterfall model), un modelo más lineal y secuencial.

Existen 4 reglas vitales dentro de este manifiesto:

* Individuos e interacciones sobre procesos y herramientas.
* Software funcionando sobre documentación extensiva.
* Colaboración con el cliente sobre negociación contractual.
* Respuesta ante el cambio sobre seguir un plan.

Principios detrás del Manifiesto Ágil:

* Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
* Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
* Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
* Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
* Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
* El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
* El software funcionando es la medida principal de progreso.
* Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
* La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
* La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
* Las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados.

## 2.2 Metodología

Para la planificación y gestión del proyecto se seguirá las fases que plantea la metodología Xtreme Programming (XP) y el marco de trabajo SCRUM.

## 2.3 Xtreme Programming (XP)

Extreme Programming (XP) surge como una nueva manera de encarar proyectos de software, proponiendo una metodología basada esencialmente en la simplicidad y agilidad. Las metodologías de desarrollo de software tradicionales (ciclo de vida en cascada, evolutivo, en espiral, iterativo, etc.) aparecen, comparados con los nuevos métodos propuestos en XP, como pesados y poco eficientes. La crítica más frecuente a estas metodologías “clásicas” es que son demasiado burocráticas.

Los métodos ágiles cambian significativamente algunos de los énfasis de las metodologías “clásicas”:

* Los métodos ágiles son adaptables en lugar de predictivos.
* Los métodos ágiles son orientados a la gente y no orientados al proceso.

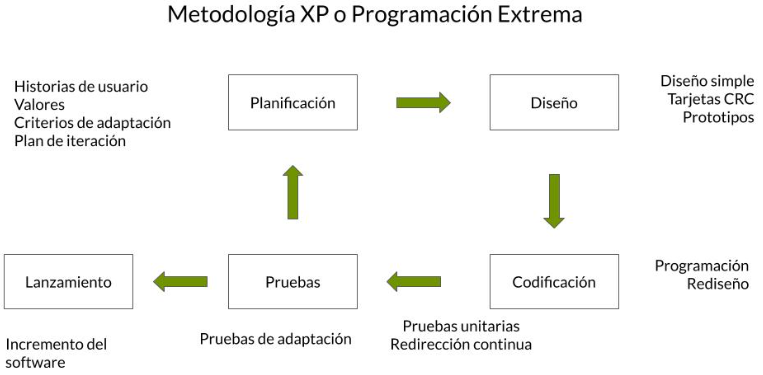
XP fue introducida como metodología ágil de desarrollo de software sobre finales de los 1990s.

La metodología XP define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance. Además, se especifica que, de estas cuatro variables, sólo tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto).

### 2.3.1 Características de la metodología XP

* Comunicación constante entre el cliente y el equipo de desarrollo.
* Respuesta rápida a los cambios constantes.
* La planificación es abierta con un cronograma de actividades flexible.
* Los requisitos del cliente y el trabajo del equipo del proyecto son los principales factores de éxito del mismo.

## 2.4 Fases de XP



*Figura 1: Fases de XP*

*Fuente: Elaboración Propia*

### 2.4.1 Fase de planificación

En esta etapa, se identificarán las historias de usuario. Estas son tarjetas donde se detallan las funcionalidades específicas del software a desarrollar. A la misma ficha, asignaremos un número y un título para poder ubicarla fácilmente más adelante, y también especificaremos a las personas que se encargaran de su concreción. Las historias de usuario se determinan basándonos en las necesidades del cliente. Cada función o historia de usuario se divide según su prioridad y se descompone en versiones. La planificación se revisa cada dos semanas, aproximadamente, para completar las entregas que el cliente debe examinar.

### 2.4.2 Fase de diseño

Se buscará que sea un código sencillo, con el flujo indispensable para hacer funcionar la historia del usuario y considerando siempre su experiencia. El diseño incluye una planificación de programación colaborativa, flexible e integral, para luego pasar a la siguiente fase, donde se evaluará una versión de prueba.

Para un diseño de software orientado a objetos, se crearán tarjetas CRC (Clase-Responsabilidad-Colaboración). Estas tarjetas describen las clases utilizadas en la programación de una historia.

### 2.4.3 Fase de codificación

Comienza la fase de programación. Este proceso de la metodología de programación extrema está pensado para que sea universal. De entrada, se trabaja en parejas frente al mismo ordenador. La meta es obtener un código de propiedad colectiva (recordemos que la metodología XP busca evitar la personalización de códigos a manos de un solo programador, así todo el equipo puede avanzar de forma simultánea y tener conocimiento del progreso).

### 2.4.4 Fase de pruebas

Una de las características de la metodología XP es el cambio constante, por eso cuando el código de una función está listo se somete a una serie de pruebas unitarias continuas, con el objetivo de corregir fallas periódicamente. XP trabaja con tiempos relativamente cortos, por lo que el control automatizado y constante es muy importante. Muchas veces, es el propio cliente quien cumple las funciones de Tester cuando tiene conocimientos de programación.

## 2.5 Los valores de XP

XP tiene valores y principios que deben tenerse en cuenta y practicarlos en el tiempo que dura el proyecto así mismo XP está sostenida sobre valores y principios propios de las metodologías ágiles, Existen cuatro valores importantes ya que son los pilares en las metodologías livianas.

* **La comunicación**: En esta metodología es muy importante que exista un ambiente de colaboración y comunicación. Esto es importante que exista dentro del equipo de desarrollo, así como con el cliente mismo. En XP la interacción con el cliente es tan estrecha, que es considerado como parte del equipo de desarrollo.
* **La simplicidad**: Este valor se aplica en todos los ámbitos de la programación extrema. Desde diseños muy sencillos donde lo más importante es la funcionalidad necesaria que requiere el dueño del producto es decir el cliente, hasta la simplificación del código mediante la refactorización del mismo. La programación extrema no usa sus recursos para las actividades complejas, sólo se desarrolla lo que el cliente demanda, de la forma más sencilla posible. Crearemos algo de lo cual podamos sentirnos orgullosos y que pueda mantenerse en el largo plazo a costos razonables.
* **Feedback**: El feedback es una parte elemental de la comunicación. Dado que los cambios son inevitables es importante obtener el feedback. Ya que se utilizará el feedback para acercarnos a nuestros objetivos. Adaptamos nuestros procesos al proyecto y no a lo contrario
* **Coraje**: El coraje es la fuerza de voluntad que puede desarrollar una persona, sin miedo al fracaso, para superar impedimentos y que a veces se manifiesta cómo la tendencia a la acción
* **Respeto**: El respeto es importante, debemos tener respeto por nuestro trabajo y el trabajo de los demás, si no tenemos respeto por los demás XP no funciona. Cada uno aporta valor incluso si es simple entusiasmo. Los desarrolladores respetan al cliente y viceversa.

## 2.6 Prácticas de XP

XP sugiere usar 12 prácticas mientras desarrollamos software. Cómo XP está definido por valores y principios, sus prácticas también los representan y pueden agruparse en cuatro grupos:



*Figura 2: Prácticas de XP*

*Fuente: Elaboración Propia*

### 2.6.1 Juego de la planificación

Se tiene que tener una comunicación frecuente entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo ya sea en hora o semanas requeridas para la implementación de las historias de usuario, las historias de usuario vendrían a ser los requisitos de nuestro sistema y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.

### 2.6.2 Entregas pequeñas

Producir rápidamente versiones del sistema que sean operativas, es decir un gran proyecto lo dividimos en secciones pequeñas y vamos haciendo esas entregas operativas aunque no cuenten con toda la funcionalidad del sistema. No deben de tardar más de 3 meses, van desde una semana a 90 días.

### 2.6.3 Desarrollo dirigido por pruebas

La producción de código está dirigida por las pruebas unitarias. Establecidas por el cliente antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema.

### 2.6.4 Refactorización

La refactorización es la actividad constante de reestructuración del código.

* Evitar duplicación del código
* Mejorar su legibilidad
* Hacerlo más simple
* Hacerlo más flexible, para facilitar cambios posteriores
* Se mejora la estructura interna del código sin alterar su comportamiento externo

### 2.6.5 Programación en parejas

Toda la producción del código debe realizarse con trabajo en parejas de programadores, esto nos beneficia mucho ya que cómo bien sabemos dos cabezas piensan mejor que una.

### 2.6.6 Propiedad colectiva del código

Cualquier programador puede cambiar cualquier parte del código en cualquier momento, por ejemplo, si en algún momento un programador se enferma o se va de vacaciones otro podrá tomar su lugar y realizar los cambios correspondientes.

### 2.6.7 Integración continua

Cada pieza de código es integrada en el sistema una vez que esté lista (desarrollo incremental). El sistema puede llegar a ser integrado y construido varias veces en un mismo día.

### 2.6.8 40 horas por semana

Trabajo máximo de 40 horas semanales. Todos los integrantes del equipo de desarrollo no deben trabajar más de 40 horas semanales, si hacen horas extras no deben ser dos semanas seguidas. El trabajo extra desmotiva el equipo. Alguien cansado no es productivo

### 2.6.9 Metáfora

El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo.

### 2.6.10 Diseño simple

Lo mejor siempre es el diseño simple, fácil y sencillo posible.

### 2.6.11 Cliente in-situ

El cliente presente y disponible todo el tiempo para el equipo, el conduce constantemente el trabajo hacia lo que aportará más valor para el negocio, de modo inmediato. Comunicación oral más efectiva que la escrita. Es más importante sentar al cliente con el equipo de desarrollo.

### 2.6.12 Estándares de programación

XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de la programación para mantener el código legible.

## 2.7 Ciclo de vida de XP

### 2.7.1 Fase de exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la entrega del producto.

### 2.7.2 Fase de planificación de la entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

### 2.7.3 Fase de Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

### 2.7.4 Fase de producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

### 2.7.5 Fase de mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

### 2.7.6 Fase de muerte del proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

## 2.8 Roles de XP

En cuanto a los agentes implicados en el Extreme Programming, los equipos de un proyecto de esta tipología y magnitud tienen normalmente los siguientes roles:

### 2.8.1 Clientes

Establecen las prioridades y marca el proyecto. Suelen ser los usuarios finales del producto y quiénes marcan las necesidades. El rol de cliente es tan crucial cómo el de un desarrollador, el cliente es el que debería saber que programar, mientras que el desarrollador es el que sabe cómo programar.

Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuales se implementan en cada iteración centrándose en aportar mayor valor al negocio.

### 2.8.2 Programadores

Este rol es el más importante de la programación extrema ya que estos son los que se encargarán de desarrollar el Extreme Programming. Estos tienen cómo responsabilidad: estimar historias, definir tareas, estimar tareas, escribir pruebas de unidad, realizar unit testing, refactorización. El programador escribe las pruebas unitarias y produce el código del sistema.

### 2.8.3 Testers

Se encargan de ayudar al cliente sobre los requisitos del producto. su comunicación con el cliente será vital para alinear resultados con los requisitos estimados. Este rol ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para las pruebas.

### 2.8.4 Coach

Los Coach realizan una tarea fundamental: el asesoramiento y orientación continua tanto para el equipo de trabajo como para los clientes. Son la guía del proyecto, para que todos sepan bien qué, cómo y cuándo hacerlo. Asesoran al resto de componentes del equipo y marcan el rumbo del proyecto. Es el responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

### 2.8.5 Manager

El responsable de coordinar comunicaciones entre las distintas partes, clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es la coordinación.

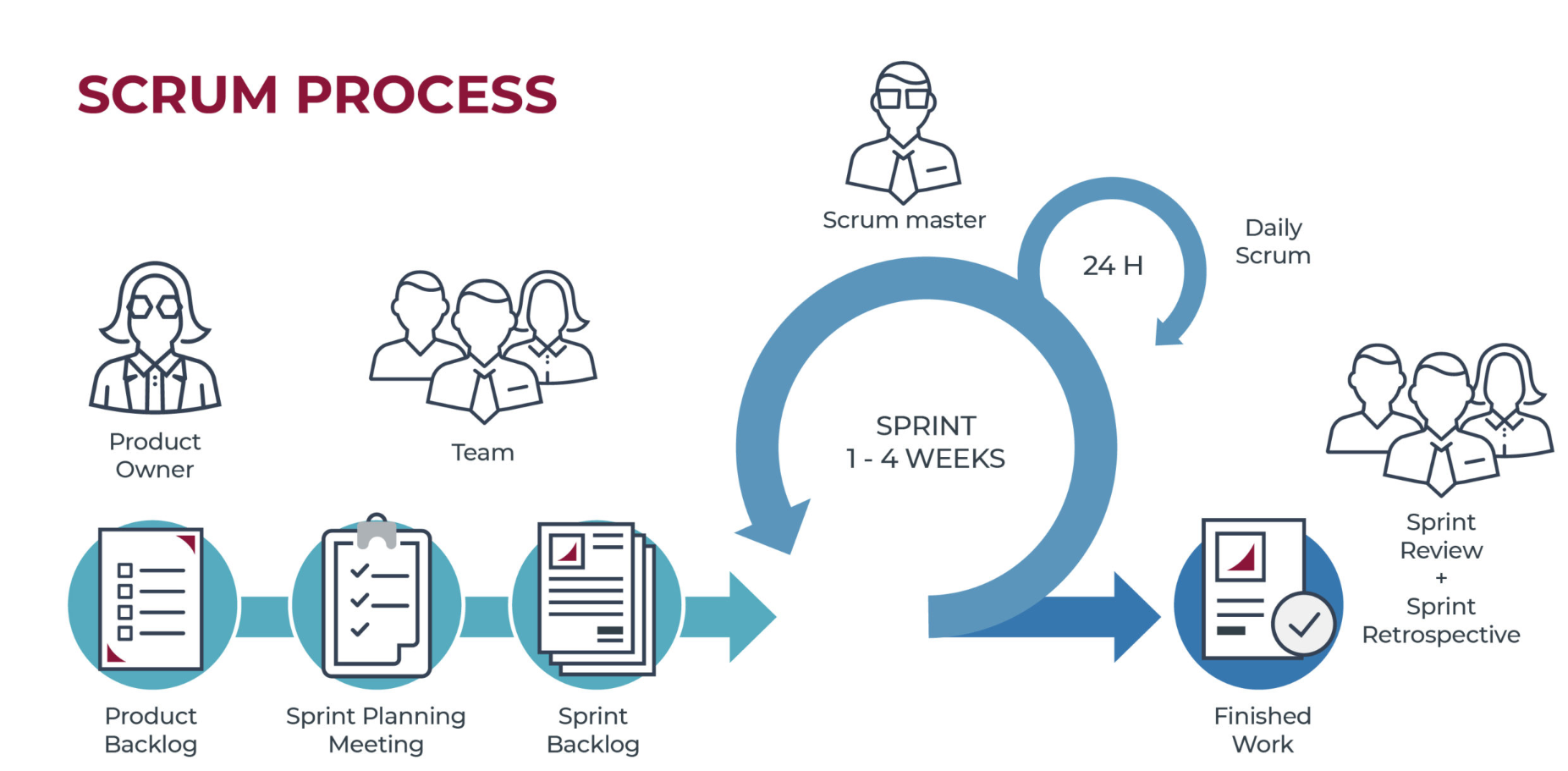
## 2.9 Marco de trabajo SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

### 2.9.1 Proceso

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas, límite máximo de feedback de producto real y reflexión). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.



*Figura 3: El proceso de SCRUM*

*Fuente: Elaboración Propia*

#### 2.9.1.1 Planificación de la iteración

El primer día de la iteración se realiza la reunión de planificación de la iteración. Tiene dos partes:

* Selección de requisitos (2 horas). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que prevé que podrá completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.
* Planificación de la iteración (2 horas). El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos seleccionados. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se autoasignan las tareas, se autoorganizan para trabajar incluso en parejas (o grupos mayores) con el fin de compartir conocimiento (creando un equipo más resiliente) o para resolver juntos objetivos especialmente complejos.

#### 2.9.1.2 Ejecución de la iteración

Cada día el equipo realiza una reunión de sincronización (15 minutos), normalmente delante de un tablero físico o pizarra (Scrum Taskboard). El equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con la previsión de objetivos a mostrar al final de la iteración. En la reunión cada miembro del equipo responde a tres preguntas:

* ¿Qué he hecho desde la última reunión de sincronización para ayudar al equipo a cumplir su objetivo?
* ¿Qué voy a hacer a partir de este momento para ayudar al equipo a cumplir su objetivo?
* ¿Qué impedimentos tengo o voy a tener que nos impiden conseguir nuestro objetivo?

Durante la iteración el Facilitador (Scrum Master) se encarga de que el equipo pueda mantener el foco para cumplir con sus objetivos.

* Elimina los obstáculos que el equipo no puede resolver por sí mismo.
* Protege al equipo de interrupciones externas que puedan afectar el objetivo de la iteración o su productividad.

Durante la iteración, el cliente junto con el equipo refina la lista de requisitos (para prepararlos para las siguientes iteraciones) y, si es necesario, cambian o planifican los objetivos del proyecto (10%-15% del tiempo de la iteración) con el objetivo de maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

#### 2.9.1.3 Inspección y adaptación

El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

* Revisión (demostración) (1,5 horas). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente puede realizar las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.
* Retrospectiva (1,5 horas). El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Facilitador se encargará de eliminar o escalar los obstáculos identificados que estén más allá del ámbito de acción del equipo.

### 2.9.2 Roles de SCRUM

#### 2.9.2.1 Product owner

Es el responsable de maximizar el valor del trabajo del equipo de desarrollo. La maximización del valor del trabajo viene de la mano de una buena gestión del Product Backlog, el cual explicaremos más adelante.

El Product Owner es el único perfil que habla constantemente con el cliente, lo que le obliga a tener muchos conocimientos sobre el negocio. Para finalizar, un equipo Scrum debe tener solo un Product Owner y este puede ser parte del equipo de desarrollo.

#### 2.9.2.2 Scrum Master

Es el responsable de que las técnicas Scrum sean comprendidas y aplicadas en la organización. Es el manager de Scrum, un líder que se encarga de eliminar impedimentos o inconvenientes que tenga el equipo dentro de un sprint (que ya revisaremos en detalle más adelante), aplicando las mejores técnicas para fortalecer el equipo de marketing digital. Dentro de la organización, el Scrum Master tiene la labor de ayudar en la adopción de esta metodología en todos los equipos.

#### 2.9.2.3 Equipo de desarrollo

Son los encargados de realizar las tareas priorizadas por el Product Owner. Es un equipo multifuncional y autoorganizado. Son los únicos que estiman las tareas del product backlog, sin dejarse influenciar por nadie. Los equipos de desarrollo no tienen sub-equipos o especialistas. La finalidad de esto es transmitir la responsabilidad compartida si no se llegan a realizar todas las tareas de un sprint.

### 2.9.3 Hitos de SCRUM

#### 2.9.3.1 Sprint

El sprint es el corazón de Scrum, es el contenedor de los demás hitos del proceso. Todo lo que ocurre en una iteración para entregar valor está dentro de un sprint. La duración máxima es de un mes, el tiempo se determina en base al nivel de comunicación que el cliente quiere tener con el equipo. Los sprints largos pueden hacer que se pierda feedback valioso del cliente y poner en peligro el proyecto.

#### 2.9.3.2 Sprint planning

En esta reunión todo el equipo Scrum define qué tareas se van a abordar y cuál será el objetivo del sprint. La primera reunión que se hace en el sprint puede llegar a tener una duración de 8 horas para sprints de un mes. El equipo se hace las siguientes preguntas:

* ¿Qué se va a hacer en el sprint? En base a ello, se eligen tareas del Product backlog.
* ¿Cómo lo vamos a hacer? El equipo de desarrollo define las tareas necesarias para completar cada ítem elegido del Product Backlog.

La definición de qué se va a hacer implica que el equipo tenga un objetivo y se encuentre comprometido con la entrega de valor que se hará al cliente al final del sprint. A esto se le llama sprint goal. El resultado de esta reunión es el sprint goal y un sprint backlog (que revisaremos más adelante).

#### 2.9.3.3 Daily meeting

Es una reunión diaria dentro del sprint que tiene como máximo 15 minutos de duración. En ella deben participar, sí o sí, el equipo de desarrollo y el Scrum Master. El Product Owner no tiene necesidad de estar presente. En esta reunión diaria el equipo de desarrollo hace las siguientes tres preguntas:

* ¿Qué hice ayer?
* ¿Qué voy a hacer hoy?
* ¿Tengo algún impedimento que necesito que me solucionen?

Esta reunión es la más oportuna para poder inspeccionar el trabajo y poder adaptarse en caso de que haya cambio de tareas dentro de un sprint.

#### 2.9.3.4 Sprint review

La revisión del valor que vamos a entregar al cliente se hace en esta reunión, al final de cada sprint. Su duración es de 4 horas para sprints de un mes, y es la única reunión de Scrum a la que puede asistir el cliente. En ella el Product Owner presenta lo desarrollado al cliente y el equipo de desarrollo muestra su funcionamiento. El cliente valida los cambios realizados y además brinda feedback sobre nuevas tareas que el Product Owner tendrá que agregar al Product backlog.

#### 2.9.3.5 Sprint retrospective

La retrospectiva es el último evento de Scrum, tiene una duración de 3 horas para Sprints de un mes, y es la reunión del equipo en la que se hace una evaluación de cómo se ha implementado la metodología Scrum en el último sprint. Es una gran oportunidad para el equipo Scrum de inspeccionarse a sí mismo, proponiendo mejoras para el siguiente sprint.

El resultado: una lista de mejoras que debe aplicar el siguiente día, ya que, al finalizar la retrospectiva, inmediatamente comienza un nuevo sprint, que incluye el sprint planning, daily meeting, sprint review y el ya mencionado sprint retrospective.

### 2.9.4 Herramientas de SCRUM

#### 2.9.4.1 Product backlog

Básicamente, el product backlog es el listado de tareas que engloba todo un proyecto. Cualquier cosa que debamos hacer debe estar en el product backlog y con un tiempo estimado por el equipo de desarrollo.

La responsabilidad exclusiva de ordenar el product backlog es del Product Owner, que se encuentra en constante comunicación con el cliente para asegurarse de que las prioridades están bien establecidas.

La ordenación también es 100% responsabilidad del Product Owner, por lo que las tareas que están más arriba deben de ser las de mayor prioridad.

El equipo de desarrollo elige tareas del product backlog en el sprint planning para generar tanto el sprint backlog como el sprint goal.

#### 2.9.4.2 Sprint backlog

Es el grupo de tareas del product backlog que el equipo de desarrollo elige en el sprint planning junto con el plan para poder desarrollarlas. Debe ser conocido por todo el equipo, para asegurarse de que el foco debe estar en este grupo de tareas.

El sprint planning no cambia durante el sprint, solo se permite cambiar el plan para poder desarrollarlas.

## 2.10 Historias de usuario

Las “Historias de usuarios” (“User stories”) sustituyen a los documentos de especificación funcional, y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar.

Las historias de usuario deben tener el mínimo detalle como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.

Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

Las historias de los usuarios a menudo se escriben en tarjetas o notas adhesivas y se organizan en paredes o mesas para facilitar la planificación y el debate.

### 2.10.1 Elementos de una historia de usuario

A continuación, se realiza la descripción de cada uno de estos elementos:

#### 2.10.1.1 Identificador (ID) de la historia:

Código que identifica inequívocamente a la historia en el Proyecto que se esté desarrollando. El formato debe ser elegido por el equipo.

#### 2.10.1.2 Rol

Es el rol que está desempeñando el usuario cuando utiliza la funcionalidad que se está describiendo. Debe ser lo más específico posible, describiendo el rol o actor que se está desempeñando. El enunciado puede escribirse como se sigue: Yo como un [Rol], Desempeñando el rol de [Rol], Como un [Rol], entre otros. Por ejemplo:

* Yo como cliente registrado.
* Desempeñando el rol de cliente registrado.
* Como un cliente registrado.

#### 2.10.1.3 Característica / Funcionalidad (Feature)

Representa la función que el rol quiere o necesita hacer en el sistema que se está desarrollando. Puede diferenciarse entre acciones obligatorias u opcionales, utilizando la palabra puede o necesita para describir la acción. Por ejemplo:

* Necesito realizar búsquedas de productos por categorías.
* Puedo seleccionar una categoría para ver el número de productos que tiene asociado.

#### 2.10.1.4 Razón / Resultados

Lo que el rol necesita lograr al ejecutar la acción. Este es el resultado de ejecutar la acción desde el punto de vista del rol. Este punto puede ser opcional, pues la historia puede documentarse sólo con la definición del rol y la acción (sin definir la consecuencia).

## 2.11 El dueño del producto (Product Owner)

El Product Owner es el encargado de optimizar y maximizar el valor del producto, siendo la persona encargada de gestionar el flujo de valor del producto a través del Product Backlog. Adicionalmente, es fundamental su labor como interlocutor con los stakeholders y sponsors del proyecto, así como su faceta de altavoz de las peticiones y requerimientos de los clientes. Si el Product Owner también juega el rol de representante de negocio, su trabajo también aportará valor al producto.

## 2.12 El Scrum Master

Es el responsable del cumplimiento de las reglas de un marco de scrum técnico, asegurando que se entienden en la organización, y se trabaje conforme a ellas. Proporciona la asesoría y formación necesaria al dueño del producto y el equipo. Realiza su trabajo con un modelo de liderazgo servil: al servicio y en ayuda del equipo y del dueño del producto. Proporciona:

* Asesoría y formación al equipo para trabajar de forma autoorganizada y con responsabilidad de equipo.
* Revisión y validación de la pila del producto.
* Moderación de las reuniones.
* Resolución de impedimentos que en el sprint pueden entorpecer la ejecución de las tareas.
* Gestión de las dificultades de dinámica de grupo que se puedan generar en el equipo.
* Configuración, diseño y mejora continua de las prácticas de scrum en la organización.
* Respeto de la organización y los implicados, con las pautas de tiempos y formas de scrum

## 2.13 Lista de producto (Product backlog)

La Lista de Producto es una lista ordenada de todo lo que podría ser necesario en el producto, y es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto (Product Owner) es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación. Una lista de productos nunca está completa. solo refleja los requisitos conocidos y mejor entendidos al principio. La Lista de Producto evoluciona a medida que el producto y el entorno en el que se usará también lo hacen. La Lista de Producto es dinámica.

El Product Backlog o pila de producto en un proyecto que sigue la metodología Scrum consiste en una lista con todos los requerimientos iniciales del producto que se va a desarrollar. Se trata de una lista dinámica, que irá evolucionando a medida que lo hace el producto y el entorno del proyecto. La finalidad de crear esta lista no es otra que identificar las necesidades del producto para lograr su máxima utilidad.

## 2.14 Planning Poker

Es una práctica ágil, para conducir las reuniones en las que se estima el esfuerzo y la duración de tareas. James Grenning ideó este juego de planificación para evitar discusiones dilatadas que no terminan de dar conclusiones concretas. El modelo inicial de Grenning consta de 7 cartas, con los números 1,2,3,5,7,10 e infinito.

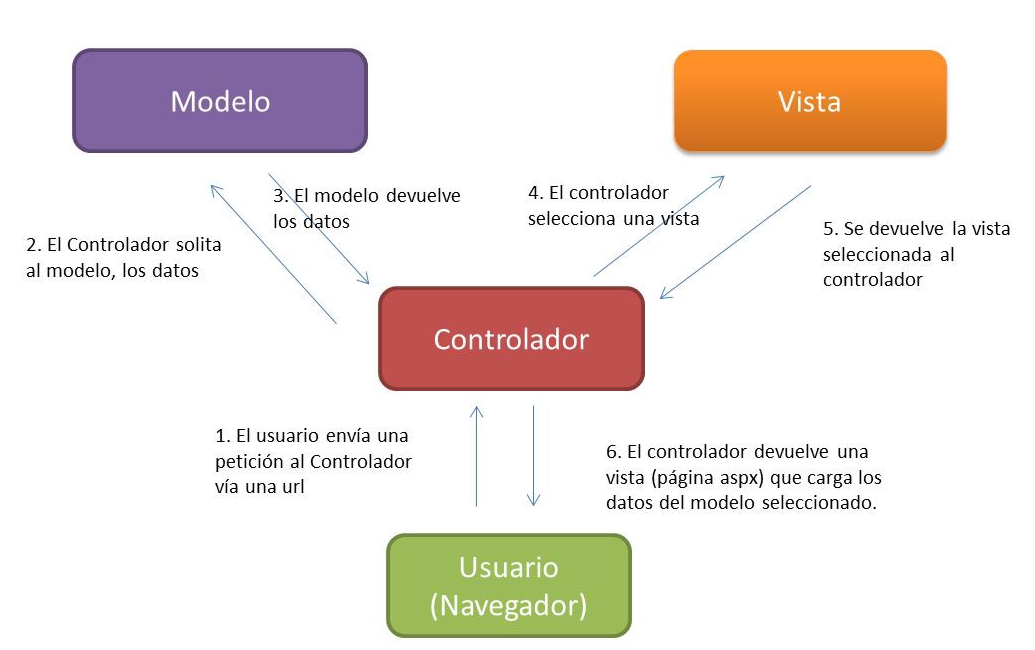
El funcionamiento es muy simple: cada participante dispone de un juego de cartas, y en la estimación de cada tarea, todos vuelven boca arriba la combinación que suma el esfuerzo estimado. Cuando se considera que éste es mayor de x horas ideales (el tamaño máximo considerado por el equipo para una historia), se levanta la carta. Las tareas que exceden el tamaño máximo deben descomponerse en sub tareas de menor tamaño. Cada equipo u organización puede utilizar un juego de cartas con las numeraciones adecuadas a la unidad de esfuerzo con la que trabajan, y el tamaño máximo de tarea o historia que se va a estimar.

## 2.15 Arquitectura del sistema

La arquitectura, referida al software, es un concepto que surge ya en los años 60 y se refiere a una planificación basada en modelos, patrones y abstracciones teóricas, a la hora de realizar una pieza de software de cierta complejidad y como paso previo a cualquier implementación. De esta forma se dispone de una guía teórica detallada que nos permite entender cómo van a encajar cada una de las piezas de nuestro producto o servicio.

Por tanto, en arquitectura llamamos patrón a cualquier solución general y reutilizable para problemas recurrentes en ingeniería del software en un contexto dado, son similares a los patrones usados en la programación, pero orientados específicamente a la estructura a un nivel superior y más genérico.

### 2.15.1 MVC (Modelo Vista Controlador)



*Figura 4: Modelo Vista Controlador*

*Fuente: Elaboración Propia*

Modelo-Vista-Controlador (MVC) es un patrón arquitectónico/de diseño que separa una aplicación en tres componentes lógicos principales Modelo, Vista y Controlador. Cada componente arquitectónico está construido para manejar aspectos de desarrollo específicos de una aplicación. Aísla la lógica empresarial y la capa de presentación entre sí. Se usaba tradicionalmente para interfaces gráficas de usuario (GUI) de escritorio. Hoy en día, MVC es uno de los marcos de desarrollo web estándar de la industria más utilizados para crear proyectos escalables y extensibles. También se utiliza para diseñar aplicaciones móviles.

MVC fue creado por Trygve Reenskaug. El objetivo principal de este patrón de diseño era resolver el problema de los usuarios que controlan un conjunto de datos grande y complejo al dividir una aplicación grande en secciones específicas que tienen su propio propósito.

## 2.16 Lenguajes

Los lenguajes de programación son instrucciones que puede realizar una máquina para controlar el comportamiento lógico y físico de la misma, especificando el campo exacto que se desea controlar. Debe ser correcto, claro, eficiente y multiplataforma para permitir al usuario interactuar de manera precisa el programa resultante.

### 2.16.1 Lenguaje de bajo nivel

Un lenguaje de programación de bajo nivel es el que proporciona poca o ninguna abstracción del microprocesador de un ordenador. Son lenguajes totalmente dependientes de la máquina, el programa que se realiza con este tipo de lenguajes no se puede utilizar en otras máquinas.

### 2.16.2 Lenguaje de alto nivel

Un lenguaje de programación de alto nivel se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de a la capacidad ejecutora de las máquinas. Se tratan de lenguajes independientes de la arquitectura del ordenador. Por lo que, un programa escrito en un lenguaje de alto nivel, se puede migrar de una máquina a otra sin ningún tipo de problema.

### 2.16.3 Python

Python es un lenguaje de programación multiplataforma y de código abierto que puede utilizarse tanto para desarrollo web, creación de software y procesamiento de datos, entre muchos otros propósitos. Esta versatilidad y facilidad para aprenderlo (es ampliamente considerado el lenguaje más sencillo de aprender) lo han convertido en el lenguaje de programación más popular del mundo.

Fue lanzado por primera vez en 1991 por Guido van Rossum y sigue siendo desarrollado hoy en día por la Python Software Foundation. Python se basa en los lenguajes C y C++ y tiene sus raíces en el sistema operativo UNIX. Python existe desde hace años, pero no fue hasta mediados de la década de 2000 cuando se afianzó en el mundo del desarrollo web

### 2.16.4 Javascript

JavaScript es un lenguaje de programación ligero que los desarrolladores web suelen utilizar para crear interacciones más dinámicas al desarrollar páginas web, aplicaciones, servidores e incluso juegos.

Los desarrolladores generalmente usan JavaScript junto con HTML y CSS. El lenguaje de secuencias de comandos funciona bien con CSS para formatear elementos HTML. Sin embargo, JavaScript aún mantiene la interacción con el usuario, algo que CSS no puede hacer por sí solo.

Las implementaciones de JavaScript dentro de la web, la aplicación móvil y el desarrollo de juegos hacen que valga la pena aprender el lenguaje de secuencias de comandos.

## 2.17 Frameworks

La palabra Framework es la combinación de dos palabras, es decir, Marco (frame) y Trabajo (work). Esto significa que ya se ha diseñado un marco y que el desarrollador debe funcionar en ese marco para cumplir con los requisitos de su proyecto. Es solo una herramienta que ayuda al desarrollador a codificar mejor y más rápido.

Un web framework o simplemente un marco web representa una colección de bibliotecas y módulos que permiten a los desarrolladores de aplicaciones web escribir aplicaciones sin preocuparse por los detalles de bajo nivel, como el protocolo, la gestión de subprocesos, etc.

### 2.17.1 Django

Django es un framework de desarrollo web, una herramienta de código abierto que es gratuita y cuenta con una gran comunidad detrás de este framework, Django es una herramienta que se puede usar para el desarrollo full-stack de aplicaciones y páginas web, asi cómo para el desarrollo de servidores. Está considerado como uno de los mejores frameworks de python y también el más demandado por los programadores que trabajan con este lenguaje en el desarrollo web

### 2.17.2 Bootstrap

Bootstrap es un framework de desarrollo front-end gratuito y de código abierto para la creación de sitios web y aplicaciones web. Diseñado para permitir el desarrollo receptivo de sitios web móviles, Bootstrap proporciona una colección de sintaxis para diseños de plantillas.

Como framework, Bootstrap incluye los conceptos básicos para el desarrollo web receptivo, por lo que los desarrolladores solo necesitan insertar el código en un sistema de cuadrícula predefinido. El marco Bootstrap se basa en lenguaje de marcado de hipertexto (HTML), hojas de estilo en cascada (CSS) y JavaScript. Los desarrolladores web que utilizan Bootstrap pueden crear sitios web mucho más rápido sin perder tiempo preocupándose por los comandos y funciones básicos.

## 2.18 Base de datos

Un sistema gestor de base de datos es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos. Los usuarios pueden acceder a la información usando herramientas específicas de consulta y de generación de informes, o bien mediante aplicaciones al efecto.

### 2.18.1 Mysql

MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto con un modelo cliente-servidor. RDBMS es un software o servicio utilizado para crear y administrar bases de datos basadas en un modelo relacional. Es uno de los gestores de bases de datos más populares en el mundo por lo que tiene una gran comunidad detrás de ella lo cual es bueno ya que siempre lo necesitaremos para solucionar cualquier tipo de problema.

## 2.19 IDE

Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment, es una aplicación informática que proporciona servicios integrales para facilitar al desarrollador o programador el desarrollo de software.

### 2.19.1 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

# III. Ingeniería del proyecto

## 3.1 Fase de exploración

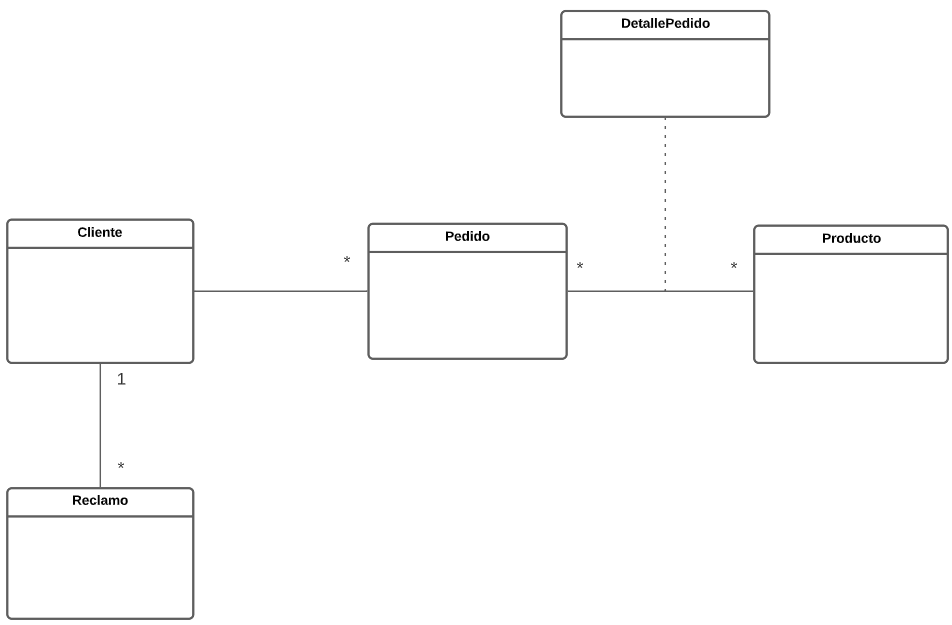
### 3.1.1 Requisitos no funcionales

|  |  |  |
| --- | --- | --- |
| Nro. | Requisitos no funcionales | Descripción |
| RNF-1 | Accesibilidad | El sistema deberá ser intuitivo y fácil de utilizar para distintos tipos de usuarios |
| RNF-2 | Disponibilidad | El sistema deberá estar disponible las 24 horas del día |
| RNF-3 | Seguridad | El sistema deberá contar con un sistema de logueo para brindar seguridad de la información |
| RNF-4 | Eficiencia en el manejo de los recursos | El sistema debe ahorrar los recursos de los equipos |
| RNF-5 | Escalabilidad | Django funciona por componentes sustituibles e intercambiables, eso significa que se puede escalar con bastante facilidad. |

*Tabla 1: Requisitos no funcionales*

*Fuente: Elaboración Propia*

### 3.1.2 Diagrama de clases conceptuales



*Figura 5: Diagrama de clases conceptuales*

*Fuente: Elaboración Propia*

### 3.1.3 Product Backlog - Historias de usuario

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Título** | |
| 001 | Gestionar Productos | |
| **Descripción** | | |
| Como gerente, necesito gestionar los productos del sistema, para realizar operaciones como agregar, editar, eliminar un producto | | |
| **Estimación (HP)** | **Prioridad** | **Dependiente de** |
| 3 | Alta | Equipo de desarrollo |
| **Criterios de Aceptación** | | |
| • Este apartado debe ser fácil de usar e intuitivo  • Este apartado del sistema permite agregar, actualizar y eliminar productos  • El sistema muestra un mensaje que la operación se realizó con éxito | | |

*Tabla 2: Historia de usuario: Gestionar Productos*

*Fuente: Elaboración Propia*

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Título** | |
| 002 | Gestionar Clientes | |
| **Descripción** | | |
| Como gerente, necesito gestionar los clientes, para realizar operaciones como agregar, editar, eliminar clientes | | |
| **Estimación (HP)** | **Prioridad** | **Dependiente de** |
| 3 | Alta | Equipo de desarrollo |
| **Criterios de Aceptación** | | |
| • Este apartado debe ser fácil de usar e intuitivo  • Este apartado del sistema permite agregar, actualizar y eliminar clientes  • El sistema muestra un mensaje que la operación se realizó con éxito | | |

*Tabla 3: Historia de usuario: Gestionar Clientes*

*Fuente: Elaboración Propia*

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Título** | |
| 003 | Registrar Quejas | |
| **Descripción** | | |
| Como gerente, necesito poder gestionar las quejas, para realizar la operación de registrar queja | | |
| **Estimación (HP)** | **Prioridad** | **Dependiente de** |
| 3 | Alta | Equipo de desarrollo |
| **Criterios de Aceptación** | | |
| • Este apartado debe ser fácil de usar e intuitivo  • Este apartado del sistema permite registrar quejas de los clientes  • El sistema muestra un mensaje que la operación se realizó con éxito | | |

*Tabla 4: Historia de usuario: Registrar Quejas*

*Fuente: Elaboración Propia*

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Título** | |
| 004 | Realizar Pedido | |
| **Descripción** | | |
| Como gerente, necesito poder registrar un pedido seleccionando al cliente, ingresando la temática y su imagen, seleccionando los productos y ingresando la fecha de pedido y de entrega | | |
| **Estimación (HP)** | **Prioridad** | **Dependiente de** |
| 7 | Alta | Equipo de desarrollo |
| **Criterios de Aceptación** | | |
| • Este apartado debe ser fácil de usar e intuitivo  • Este apartado del sistema permite realizar un pedido  • El sistema muestra un mensaje de que la operación se realizó con éxito | | |

*Tabla 5: Historia de usuario: Realizar Pedido*

*Fuente: Elaboración Propia*

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Título** | |
| 005 | Ver listado de mis pedidos | |
| **Descripción** | | |
| Como gerente, necesito poder ver la lista de pedidos que he realizado para poder llevar un control de los mismos. | | |
| **Estimación (HP)** | **Prioridad** | **Dependiente de** |
| 5 | Alta | Equipo de desarrollo |
| **Criterios de Aceptación** | | |
| • Este apartado debe ser fácil de usar e intuitivo  • Este apartado del sistema permite ver los pedidos realizados anteriormente.  • Cada pedido deberá tener la información correcta del pedido.  • Cada pedido deberá tener la opción "ver pedido" | | |

*Tabla 6: Historia de usuario: Ver listado de pedidos*

*Fuente: Elaboración Propia*

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Título** | |
| 006 | Ver pedido específico | |
| **Descripción** | | |
| Como gerente, necesito poder ver los detalles de un pedido específico para así poder mandarlo a la preparación. | | |
| **Estimación (HP)** | **Prioridad** | **Dependiente de** |
| 4 | Alta | Equipo de desarrollo |
| **Criterios de Aceptación** | | |
| • Este apartado debe ser fácil de usar e intuitivo  • Este apartado del sistema permite ver los pedidos realizados anteriormente.  • Cada pedido deberá tener la información correcta del pedido.  • Cada pedido deberá tener la opción "ver pedido" | | |

*Tabla 7: Historia de usuario: Ver pedido específico*

*Fuente: Elaboración Propia*

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Título** | |
| 007 | Completar pedido | |
| **Descripción** | | |
| Como gerente, necesito poder completar un pedido, presionando sobre un botón de entregar | | |
| **Estimación (HP)** | **Prioridad** | **Dependiente de** |
| 5 | Alta | Equipo de desarrollo |
| **Criterios de Aceptación** | | |
| • Este apartado debe ser fácil de usar e intuitivo  • Este apartado del sistema permite completar el pedido.  • Este apartado del sistema deberá tener un botón completar pedido | | |

*Tabla 8: Historia de usuario: Completar pedido*

*Fuente: Elaboración Propia*

|  |  |  |
| --- | --- | --- |
| **Nro. de Historia** | **Título** | |
| 008 | Ver pedidos completados | |
| **Descripción** | | |
| Como gerente, necesito poder ver un listado de pedidos que fueron entregados, para poder llevar un mejor control. | | |
| **Estimación (HP)** | **Prioridad** | **Dependiente de** |
| 5 | Alta | Equipo de desarrollo |
| **Criterios de Aceptación** | | |
| • Este apartado debe ser fácil de usar e intuitivo  • Este apartado del sistema deberá permitir ver un listado de los pedidos que fueron completados anteriormente. | | |

*Tabla 8: Historia de usuario: Ver pedidos completados*

*Fuente: Elaboración Propia*

## 3.2 Fase de planificación de la entrega (Release Plan)

### 3.2.1 Priorización de las historias de usuario

|  |  |  |
| --- | --- | --- |
| ID | Historias de usuario | Prioridad |
|
| 001 | Gestionar Productos | ALTA |
|
| 002 | Gestionar Clientes | ALTA |
|
| 003 | Registrar Quejas | ALTA |
|
| 004 | Realizar Pedido | ALTA |
|
| 005 | Ver listado de pedidos | ALTA |
|
| 006 | Ver pedido específico | ALTA |
|
| 007 | Completar pedido | ALTA |
|
| 008 | Ver pedidos completados | ALTA |
|

*Tabla 9: Priorización de las historias de usuario*

*Fuente: Elaboración Propia*

### 3.2.2 Estimación de esfuerzo por puntos

|  |  |  |
| --- | --- | --- |
| ID | Historias de usuario | Puntos de historia |
|
| 001 | Gestionar Productos | 4 |
|
| 002 | Gestionar Clientes | 4 |
|
| 003 | Registrar Quejas | 4 |
|
| 004 | Realizar Pedido | 7 |
|
| 005 | Ver listado de pedidos | 5 |
|
| 006 | Ver pedido específico | 4 |
|
| 007 | Completar pedido | 5 |
|
| 008 | Ver pedidos completados | 5 |
|

*Tabla 10: Estimación de esfuerzo por puntos*

*Fuente: Elaboración Propia*

## 3.3 Fase de Iteraciones: (para cada sprint)

### 3.3.1 Sprint 1: Gestionar Productos - Gestionar Clientes - Registrar Quejas

#### 3.3.1.1 Sprint Backlog

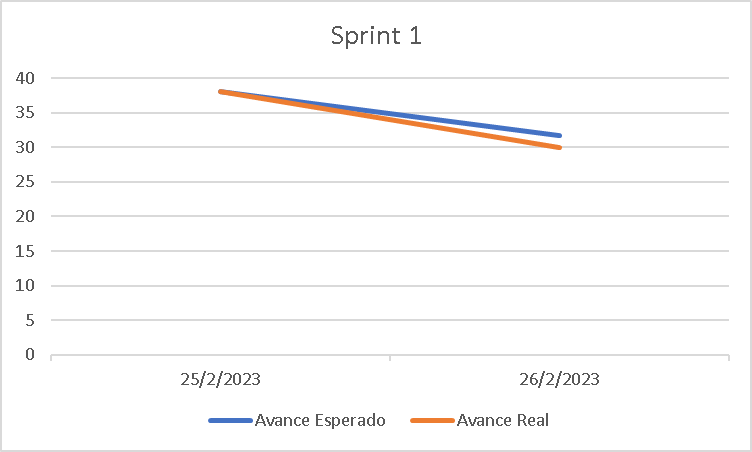
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Historia** | **Etapas** | **Estado** | **Estimación (Horas)** | **Inicio** | **Fin** |
| **Sprint 1** | Gestionar Producto | **Planificación** | Completo | 1 | 25/02/23 | 25/02/23 |
| **Ejecución** | Completo | 1 | 25/02/23 | 25/02/23 |
| **Demostración** | Completo | 1 | 25/02/23 | 25/02/23 |
| **Implementación** | Completo | 1 | 25/02/23 | 25/02/23 |
| Gestionar Cliente | **Planificación** | Completo | 1 | 25/02/23 | 25/02/23 |
| **Ejecución** | Completo | 1 | 25/02/23 | 25/02/23 |
| **Demostración** | Completo | 1 | 25/02/23 | 25/02/23 |
| **Implementación** | Completo | 1 | 25/02/23 | 25/02/23 |
| Registrar Quejas | **Planificación** | Completo | 1 | 26/02/23 | 26/02/23 |
| **Ejecución** | Completo | 1 | 26/02/23 | 26/02/23 |
| **Demostración** | Completo | 1 | 26/02/23 | 26/02/23 |
| **Implementación** | Completo | 1 | 26/02/23 | 26/02/23 |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Historias de usuario | Puntos de historia | Prioridad |
|
| 001 | Gestionar Productos | 4 | ALTA |
|
| 002 | Gestionar Clientes | 4 | ALTA |
|
| 003 | Registrar Quejas | 4 | ALTA |
|

*Tabla 11: Sprint backlog - sprint 2*

*Fuente: Elaboración Propia*

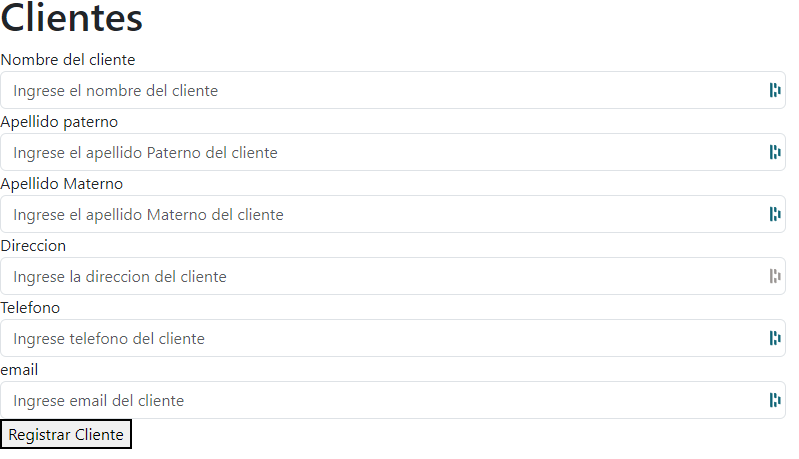
#### 3.3.1.2 Burn Down Chart



#### 3.3.1.3 Pruebas de aceptación

|  |  |  |
| --- | --- | --- |
| **Historia** | **Criterio de aceptación** | **Observaciones** |
| Gestionar Producto | El sistema deberá mostrar la interfaz para el registro del Producto | Se observa que el sistema muestra la interfaz para el registro de Producto |
| El sistema deberá validar si el Producto existe o no en la base de datos. | Se observa que sistema valida los campos solicitados |
| El sistema deberá tener la opción de guardar Producto. | Se observa que el sistema en la ventana específica tiene el botón guardar |
| El sistema debe mostrar un mensaje de confirmación (Guardado correctamente, operación no realizada). | Se observa que el sistema muestra el mensaje solicitado al realizar la operación |
| Gestionar Clientes | El sistema deberá mostrar la interfaz para el registro del cliente | Se observa que el sistema muestra la interfaz para el registro de cliente |
| El sistema deberá validar si el cliente existe o no en la base de datos. | Se observa que sistema valida los campos solicitados |
| El sistema deberá tener la opción de guardar cliente. | Se observa que el sistema en la ventana específica tiene el botón guardar |
| El sistema debe mostrar un mensaje de confirmación (Guardado correctamente, operación no realizada). | Se observa que el sistema muestra el mensaje solicitado al realizar la operación |
| Registrar Quejas | El sistema deberá mostrar una interfaz amigable y fácil de usar con los espacios requeridos. | El sistema muestra una interfaz fácil de usar ante los usuarios |
| El sistema deberá validar si los campos rellenados existen o no en la base de datos. | Se observa que sistema valida los campos solicitados |
| El sistema deberá tener la opción de guardar queja o reclamo | Se observa que el sistema en la ventana específica tiene el botón guardar |
| El sistema debe mostrar un mensaje de operación realizado/no realizada (Guardado correctamente, operación no realizada). | Se observa que el sistema muestra el mensaje solicitado al realizar la operación correctamente. |

#### 3.3.1.4 Incremento



*Figura 6: Formulario Clientes*

*Fuente: Elaboración Propia*



*Figura 7: Registros de clientes*

*Fuente: Elaboración Propia*



*Figura 8: Formulario de productos*

*Fuente: Elaboración Propia*



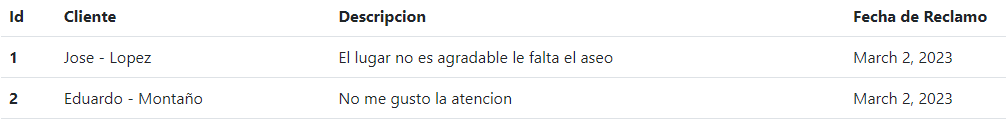
*Figura 9: Registros de productos*

*Fuente: Elaboración Propia*



*Figura 10: Formulario de reclamo*

*Fuente: Elaboración Propia*



*Figura 11: Registros de reclamos*

*Fuente: Elaboración Propia*

### 3.3.2 Sprint 2: Realizar Pedido - Ver Listado de pedidos - Ver pedido

#### 3.3.2.1 Sprint Backlog

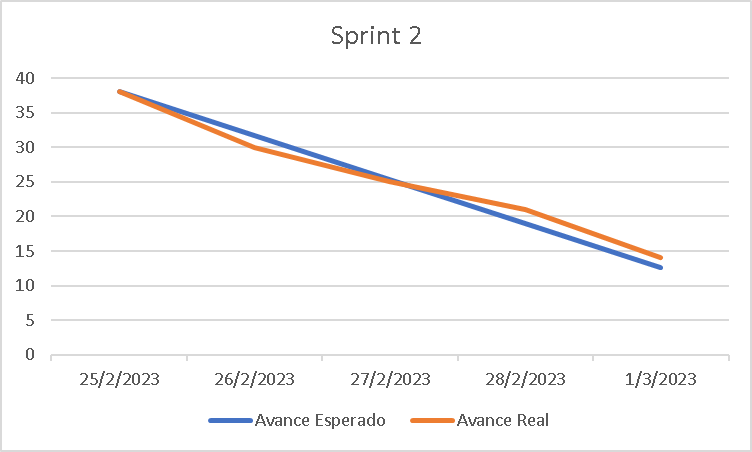
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Historia** | **Etapas** | **Estado** | **Estimación** | **Inicio** | **Fin** |
| **Sprint 2** | Realizar Pedido | **Planificación** | Completo | 1 | 27/02/23 | 27/02/23 |
| **Ejecución** | Completo | 4 | 27/02/23 | 27/02/23 |
| **Demostración** | Completo | 1 | 27/02/23 | 27/02/23 |
| **Implementación** | Completo | 1 | 27/02/23 | 27/02/23 |
| Ver listado de Pedidos | **Planificación** | Completo | 1 | 28/02/23 | 28/02/23 |
| **Ejecución** | Completo | 2 | 28/02/23 | 28/02/23 |
| **Demostración** | Completo | 1 | 28/02/23 | 28/02/23 |
| **Implementación** | Completo | 1 | 28/02/23 | 28/02/23 |
| Ver Pedido Específico | **Planificación** | Completo | 1 | 01/03/23 | 01/03/23 |
| **Ejecución** | Completo | 1 | 01/03/23 | 01/03/23 |
| **Demostración** | Completo | 1 | 01/03/23 | 01/03/23 |
| **Implementación** | Completo | 1 | 01/03/23 | 01/03/23 |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Historias de usuario | Puntos de historia | Prioridad |
|
| 004 | Realizar Pedido | 7 | ALTA |
|
| 005 | Ver listado de pedidos | 5 | ALTA |
|
| 006 | Ver pedido específico | 4 | ALTA |
|

*Tabla 12: sprint backlog - sprint 2*

*Fuente: Elaboración Propia*

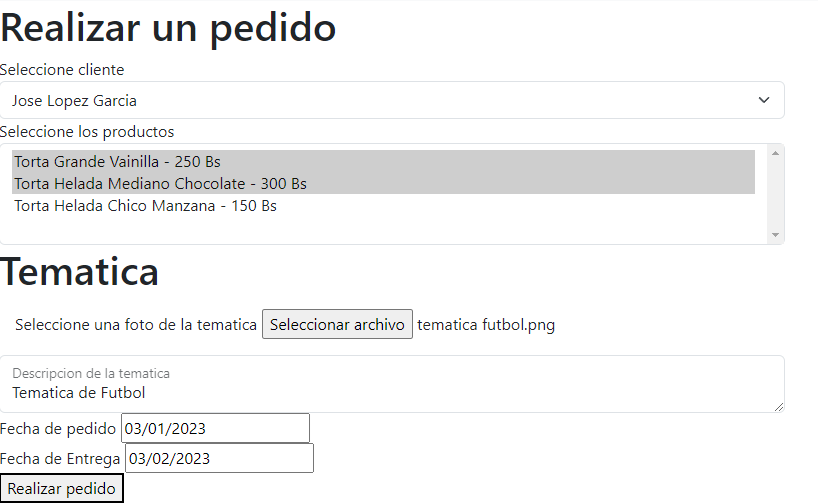
#### 3.3.2.2 Burn Down Chart



#### 3.3.2.3 Pruebas de aceptación

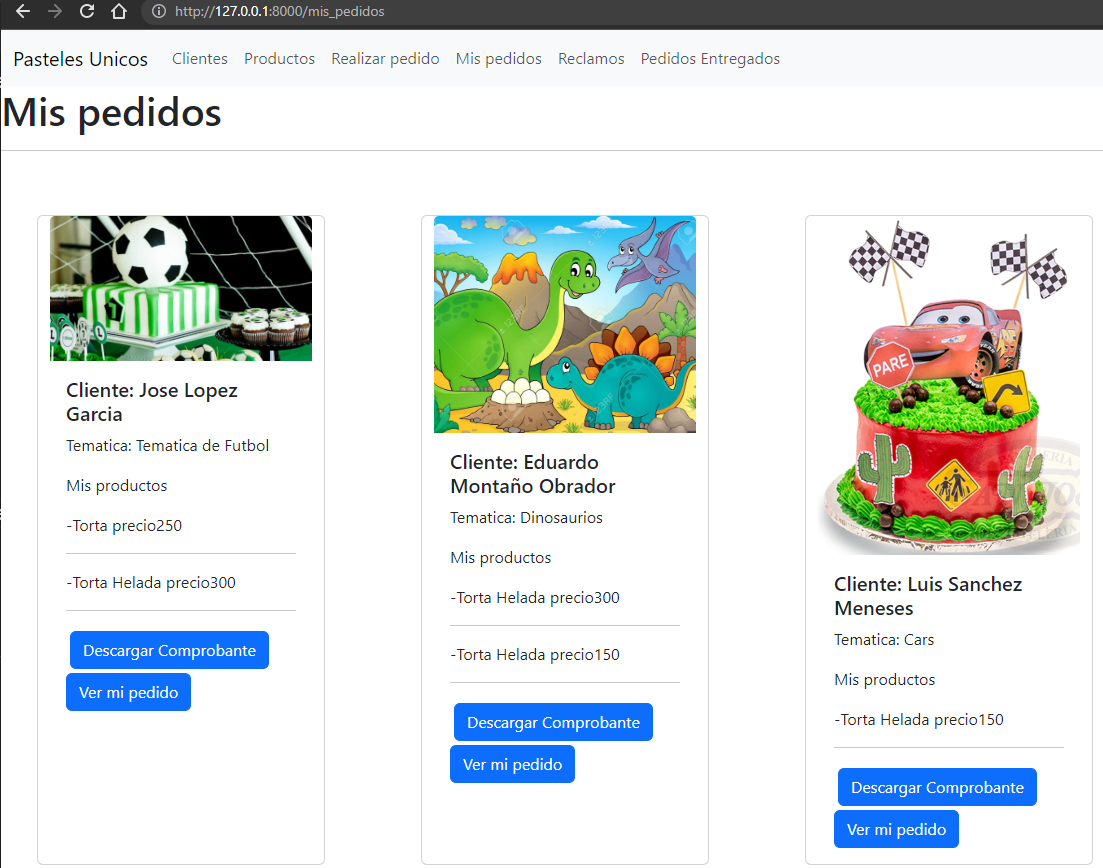
|  |  |  |
| --- | --- | --- |
| **Historia** | **Criterio de aceptación** | **Observaciones** |
| Realizar Pedido | El sistema deberá mostrar una interfaz amigable y fácil de usar con los espacios requeridos. | El sistema muestra una interfaz fácil de usar ante los usuarios |
| El sistema deberá validar si los campos rellenados existen o no en la base de datos. | Se observa que sistema valida los campos solicitados |
| El sistema debe permitir cargar archivos de imagen | El sistema permite cargar archivo de imagen |
| El sistema deberá tener la opción de guardar datos | Se observa que el sistema en la ventana específica tiene el botón guardar |
| El sistema debe mostrar un mensaje de operación realizado/no realizada (Guardado correctamente, operación no realizada). | Se observa que el sistema muestra el mensaje solicitado al realizar la operación correctamente. |
| Ver listado de pedidos | El sistema deberá mostrar una interfaz amigable y fácil de usar con las opciones necesarias | El sistema muestra una interfaz fácil de usar ante los usuarios. |
| El sistema muestra información requerida de sus clientes. | Se observa que el sistema muestra la información necesaria. |
| El sistema deberá tener la opción ver perdidos y descargar comprobante | Se observa que el sistema tiene dichas opciones |
| Ver Pedido Específico | El sistema deberá mostrar una interfaz amigable y fácil de usar. | El sistema muestra una interfaz fácil de usar ante los usuarios |
| El sistema deberá mostrar información específica de cada pedido. | Se observa que sistema mucha la información requerida. |

#### 3.3.2.4 Incremento



*Figura 12: Formulario realizar pedido*

*Fuente: Elaboración Propia*



*Figura 13 Listado de pedidos*

*Fuente: Elaboración Propia*



*Figura 14 Ver pedido específico*

*Fuente: Elaboración Propia*

### 3.3.3 Sprint 3: Completar Pedido – Ver pedidos completados

#### 3.3.3.1 Sprint Backlog

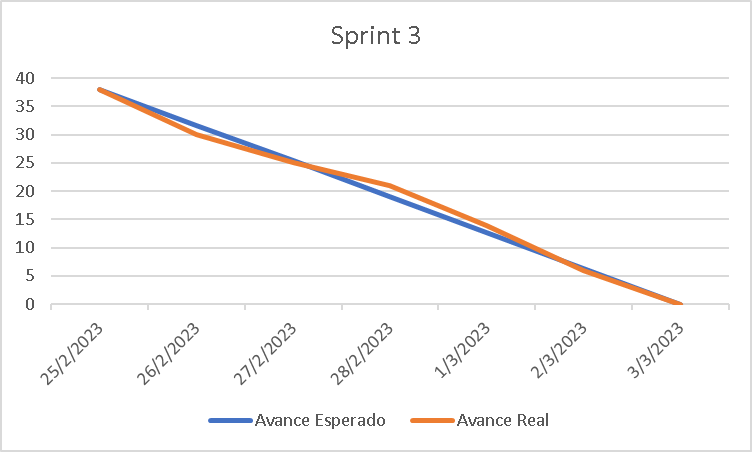
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Historia** | **Etapas** | **Estado** | **Estimación** | **Inicio** | **Fin** |
| **Sprint 3** | Completar Pedido | **Planificación** | Completo | 1 | 02/03/23 | 02/03/23 |
| **Ejecución** | Completo | 2 | 02/03/23 | 02/03/23 |
| **Demostración** | Completo | 1 | 02/03/23 | 02/03/23 |
| **Implementación** | Completo | 1 | 02/03/23 | 02/03/23 |
| Ver Pedido Completados | **Planificación** | Completo | 1 | 03/03/23 | 03/03/23 |
| **Ejecución** | Completo | 2 | 03/03/23 | 03/03/23 |
| **Demostración** | Completo | 1 | 03/03/23 | 03/03/23 |
| **Implementación** | Completo | 1 | 03/03/23 | 03/03/23 |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Historias de usuario | Puntos de historia | Prioridad |
|
| 007 | Completar pedido | 5 | ALTA |
|
| 008 | Ver pedidos completados | 5 | ALTA |
|

*Tabla 13: sprint backlog - sprint 3*

*Fuente: Elaboración Propia*

#### 3.3.3.2 Burn Down Chart



#### 3.3.3.3 Prueba de aceptacion

|  |  |  |
| --- | --- | --- |
| **Historia** | **Criterio de aceptación** | **Observaciones** |
| Completar pedido | El sistema deberá mostrar una interfaz amigable y fácil de usar con las opciones necesarias | El sistema muestra una interfaz fácil de usar ante los usuarios. |
| El sistema muestra información requerida de sus clientes. | Se observa que el sistema muestra la información necesaria. |
| El sistema deberá tener la opción de entregar pedido. | Se observa que el sistema tiene dichas opciones. |
| Ver Pedidos Completados | El sistema deberá mostrar una interfaz amigable y fácil de usar con las opciones necesarias | El sistema muestra una interfaz fácil de usar ante los usuarios. |
| El sistema muestra información requerida de sus clientes. | Se observa que sistema muestra la información necesaria. |
| El sistema deberá tener la opción ver historial de pedidos completados | Se observa que el sistema tiene dicha opción. |

#### 3.3.3.4 Incremento



*Figura 15 Completar Pedido*

*Fuente: Elaboración Propia*

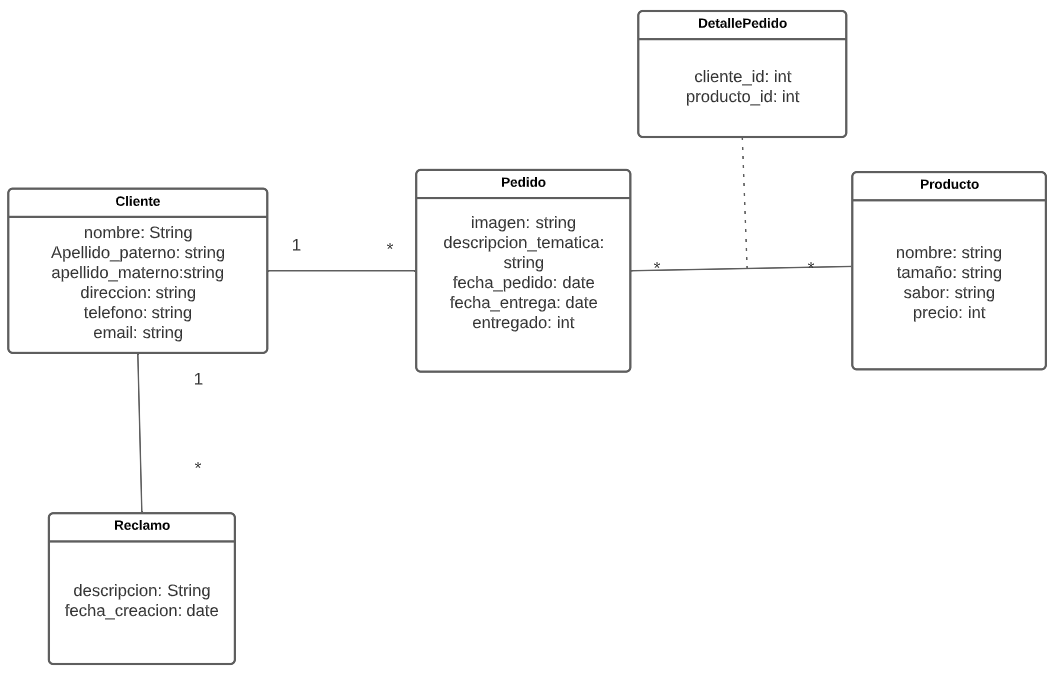


*Figura 16 Ver pedidos completados*

*Fuente: Elaboración Propia*

## 3.4 Fase de producción:

### 3.4.1 Diagrama de clases de diseño



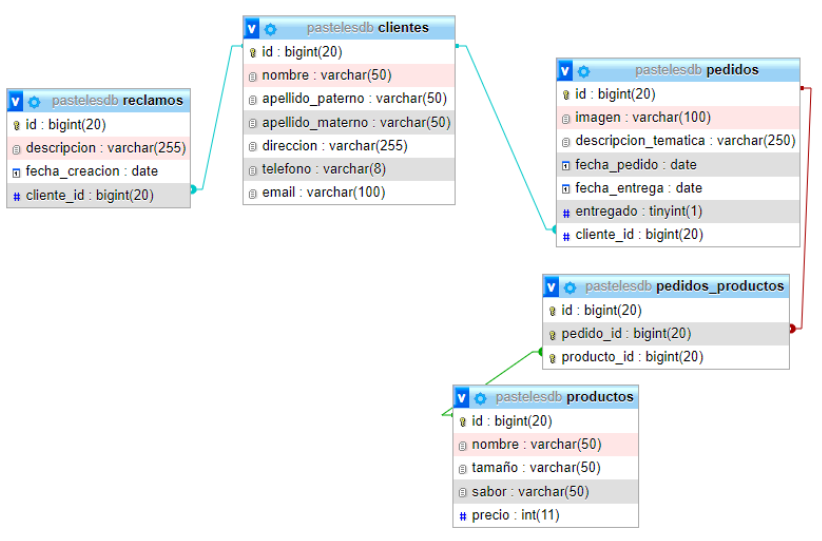
*Figura 17 Diagrama de clases de diseño*

*Fuente: Elaboración Propia*

### 3.4.2 Normalización de la base de datos

Se normaliza la base de datos, hasta la tercera forma normal, minimizando la redundancia de datos y disminuyendo los problemas de actualización de datos en las tablas, pasando por atomizar, dependencia de llave primaria y la transitividad respectivamente.

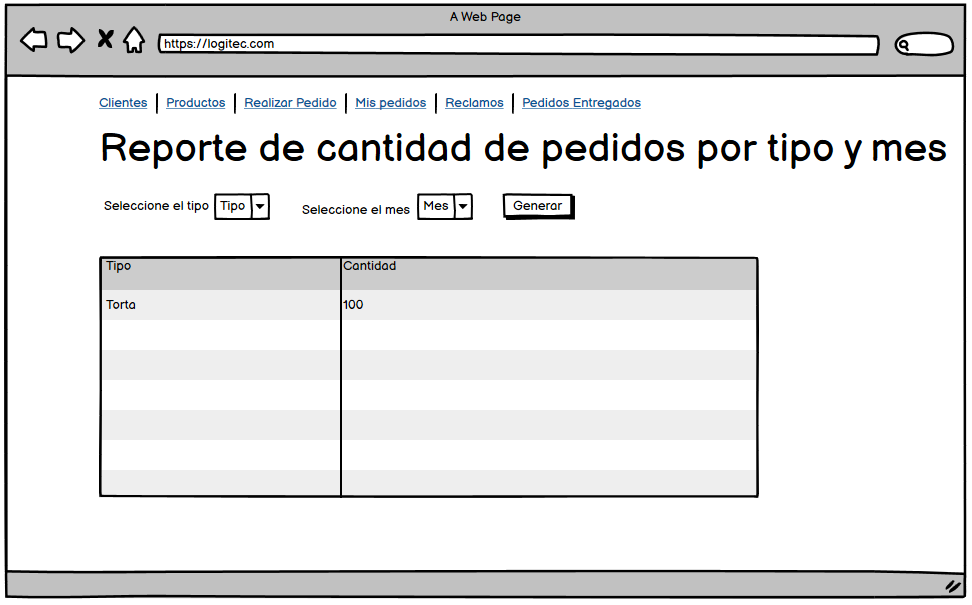
### 3.4.3 Modelo de datos relacional (modelo de base de datos lógico/físico)



*Figura 18: Modelo de datos relacional*

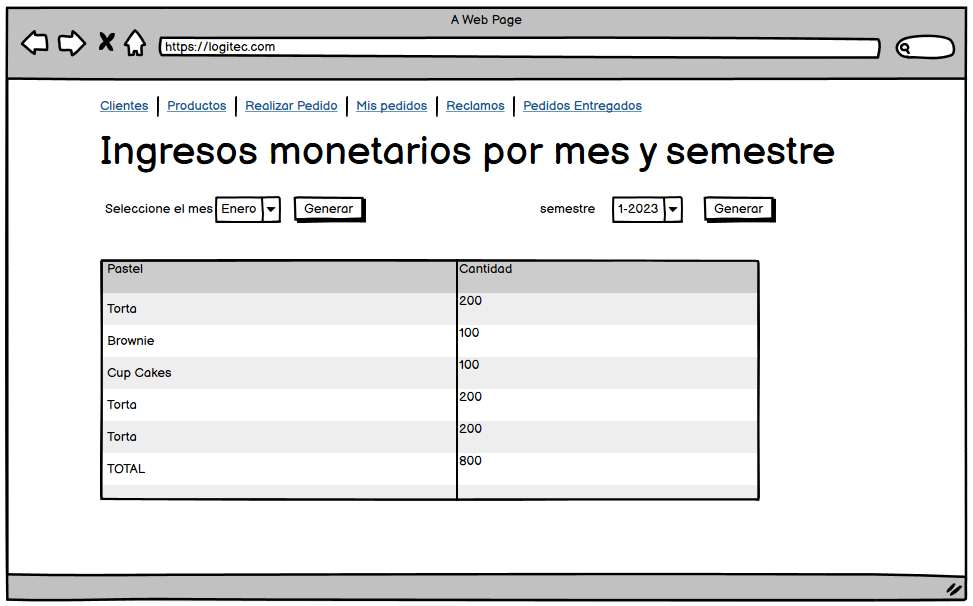
*Fuente: Elaboración Propia*

### 3.4.4 Diseño de reportes



*Figura 18: Reporte de cantidad de pedidos por tipo y mes*

*Fuente: Elaboración Propia*



*Figura 19: Reporte de ingresos monetarios por mes y semestre*

*Fuente: Elaboración Propia*

### 3.4.5 Triggers

----triggers tabla productos

----after insert

DROP TRIGGER IF EXISTS TgInsertarProductoLog;

DELIMITER $$

CREATE TRIGGER TgInsertarProductoLog AFTER INSERT ON productos

FOR EACH ROW

BEGIN

INSERT INTO Logs (descripcion, usuario, fecha)

VALUES (

"Inserto producto",

CURRENT\_USER,

NOW());

END;

$$

DELIMITER ;

--- after update

DROP TRIGGER IF EXISTS TgUpdateProductoLog;

DELIMITER $$

CREATE TRIGGER TgUpdateProductoLog AFTER UPDATE ON productos

FOR EACH ROW

BEGIN

INSERT INTO Logs (descripcion, usuario, fecha)

VALUES (

"Se actualizo el producto",

CURRENT\_USER,

NOW());

END;

$$

DELIMITER ;

--- after delete

DROP TRIGGER IF EXISTS TgDeleteProductoLog;

DELIMITER $$

CREATE TRIGGER TgDeleteProductoLog AFTER DELETE ON productos

FOR EACH ROW

BEGIN

INSERT INTO Logs (descripcion, usuario, fecha)

VALUES (

"Se elimino un producto",

CURRENT\_USER,

NOW());

END;

$$

DELIMITER ;

### 3.4.6 Consultas más complejas

select clientes.nombre as 'Nombre cliente', pedidos.descripcion\_tematica as 'Tematica del pedido', pedidos.fecha\_pedido as 'Fecha del pedido', pedidos.fecha\_entrega as 'Fecha de entrega', pedidos\_productos.pedido\_id, productos.nombre 'Nombre del producto', productos.precio as 'Precio del producto'

from clientes, pedidos, pedidos\_productos, productos

where clientes.id = pedidos.cliente\_id and pedidos.id = pedidos\_productos.pedido\_id and productos.id = pedidos\_productos.producto\_id;

### 

### 

### 3.4.7 Procedimientos almacenados

------ stored procedures

--- productos

drop procedure insertar\_producto;

DELIMITER //

create procedure insertar\_producto(in \_nombre varchar(50), in \_tamaño varchar(50), in \_sabor varchar(50), in \_precio int(11))

begin

INSERT INTO productos (nombre, tamaño, sabor, precio)

VALUES (\_nombre, \_tamaño, \_sabor, \_precio);

end//

call insertar\_producto('Brownie', 'Grande', 'Vainilla', 100);

drop procedure eliminar\_producto;

DELIMITER //

create procedure eliminar\_producto(in \_id int)

begin

DELETE FROM productos where id = \_id;

end//

call eliminar\_producto(4);

## 3.5 Fase de mantenimiento

### 3.5.1 Plan de backup de base de datos

Se deberá realizar una copia de seguridad de la base de datos, es de suma importancia que haya una copia dentro de la institución es decir de manera local y otra en un lugar externo ya sea en la nube con la posibilidad de que ésta sea restaurada en caso de siniestro o catástrofe material.

Primero se creará una copia de seguridad completa el fin de cada mes. Las modificaciones diarias deben ser respaldadas por otras copias de seguridad en esta se usará el tipo incremental de manera automática mediante uso de herramientas de ejecución automática se realizará de manera diario a las 11 p.m. en días laborales debido a que la empresa tendrá un horario de atención constante y alta demanda hasta las 8 p.m.

## 3.6 Fase de muerte del proyecto

### 3.6.1 Rendimiento del sistema

El rendimiento del sistema será óptimo ya que el servidor en el que se planea implementar es el servicio de amazon web services (AWS) que ofrece un funcionamiento de más del 95% del día.

### 3.6.2 Confiabilidad del sistema

La confiabilidad del sistema es buena ya que se trabajará de manera segura con servidores certificados, también el sistema no permitirá el ingreso de personal no autorizado de este mismo.

# IV. CONCLUSIONES

* Se identificaron las historias de usuario requeridas en el sistema.
* Se diseñó la base de datos.
* Se diseñaron los módulos que serán desarrollados.
* Se implementó la base de datos.
* Se desarrollaron los controladores para consumir los datos de DB.
* Se desarrolló la interfaz del sistema usando html, css y javascript.

# V. RECOMENDACIONES

* XP es recomendado para proyectos medianos.
* Usar XP en equipos pequeños de desarrollo.
* Integrar al cliente desde el inicio hasta el final.
* Los desarrolladores deben compartir código.
* Es indispensable que el cliente apruebe la metodología.

# VI. BIBLIOGRAFÍA

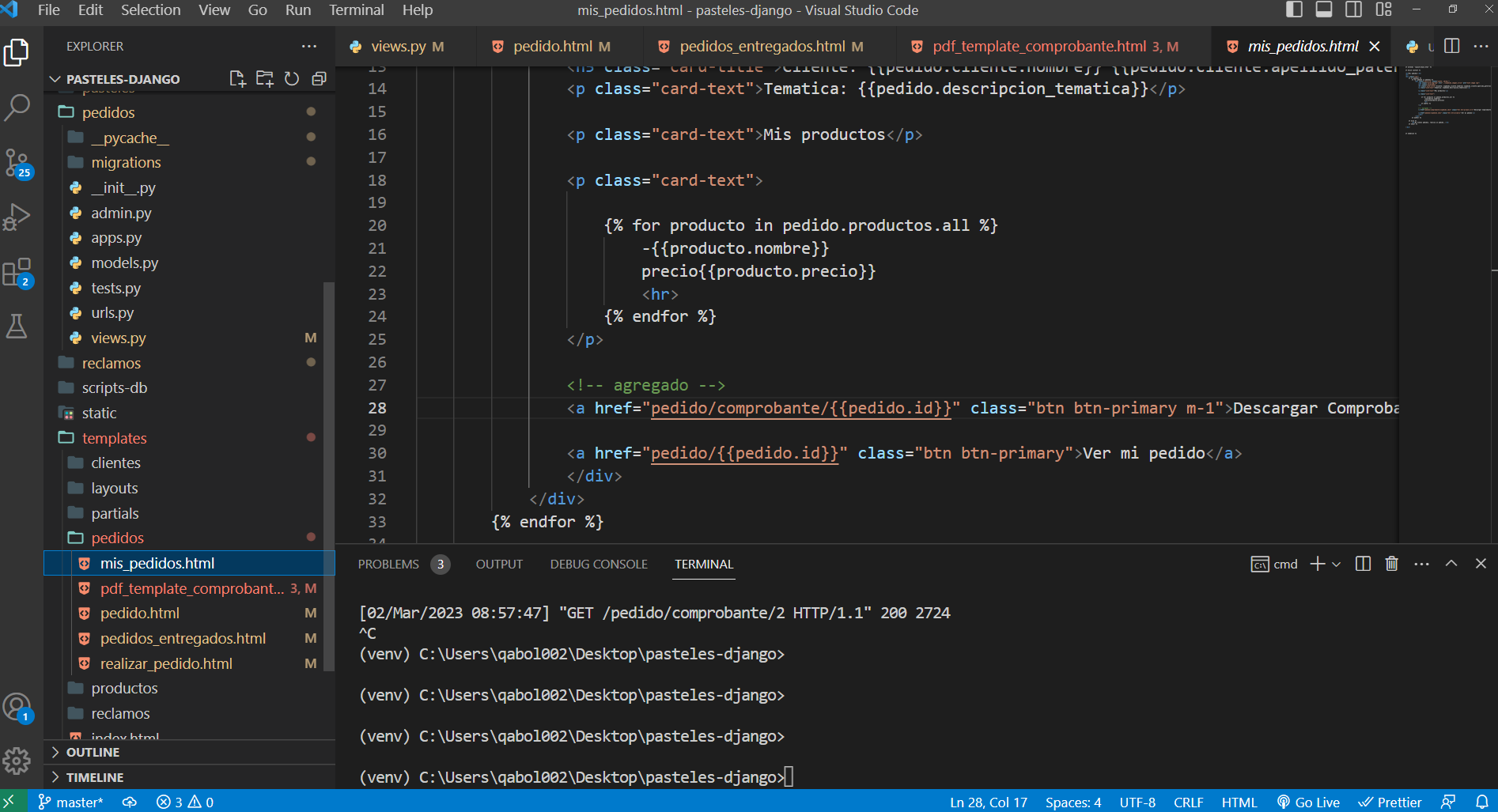
*Kendall, Kenneth E., and Julie E. Kendall. Análisis y diseño de sistemas. Translated by Antonio Núñez Ramos, Pearson Educación, 2005. Accessed 17 February 2023.*

*Pressman, Roger S. Ingeniería del software.*

*Rumbaugh, James, et al. El lenguaje unificado de modelado. Manual de referencia. Addison-Wesley, 2004.*

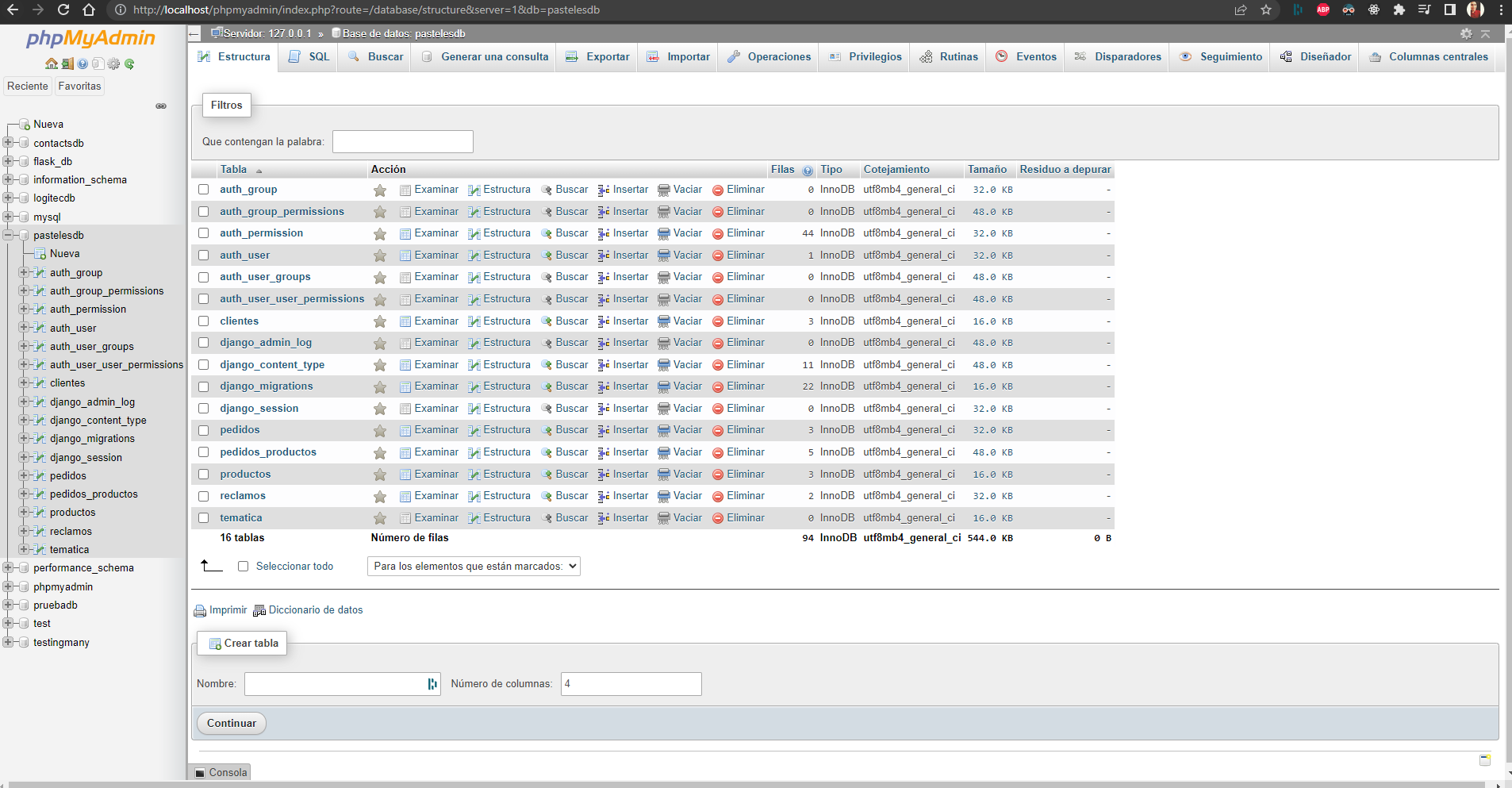
*Sudarshan, S., et al. Fundamentos de bases de datos. Edited by Luis Grau Fernández, McGraw Hill, 2002. Accessed 17 February 2023.*

# VII. ANEXOS



*Figura 20: IDE Visual studio code*

*Fuente: Elaboración Propia*



*Figura 21: phpMyAdmin base de datos pastelesdb*

*Fuente: Elaboración Propia*