**Sistema informático de reservas para la empresa Urban Style**



Franz Bascope Jordán.

Octubre 2019.

Universidad NUR.

Carrera de Ingeniería de sistemas.

Taller de grado

Contenido

[Capítulo 1 Marco Introductorio 1](#_Toc27104387)

[Introducción 1](#_Toc27104388)

[Antecedentes 1](#_Toc27104389)

[Problema 1](#_Toc27104390)

[Planteamiento del problema 2](#_Toc27104391)

[Problemas específicos 2](#_Toc27104392)

[Formulación del problema 2](#_Toc27104393)

[Objetivos 3](#_Toc27104394)

[Objetivos General 3](#_Toc27104395)

[Objetivos específicos 3](#_Toc27104396)

[Delimitación 3](#_Toc27104397)

[Delimitación espacial 3](#_Toc27104398)

[Delimitación temporal 3](#_Toc27104399)

[Delimitación sustantiva. 4](#_Toc27104400)

[Justificación 4](#_Toc27104401)

[Justificación teórica 4](#_Toc27104402)

[Justificación práctica 4](#_Toc27104403)

[Justificación social 4](#_Toc27104404)

[Justificación individual 4](#_Toc27104405)

[Estrategia metodológica. 5](#_Toc27104406)

[Investigación Aplicada 5](#_Toc27104407)

[Demostración (Factibilidad del Proyecto) 5](#_Toc27104408)

[Capítulo 2 Marco teórico 7](#_Toc27104409)

[Marco contextual 7](#_Toc27104410)

[Marco conceptual 7](#_Toc27104411)

[Metodologías de desarrollo de software 7](#_Toc27104412)

[Bases de datos relacionales (SQL) 13](#_Toc27104413)

[Bases de datos no relacionales (NO SQL) 14](#_Toc27104414)

[Capítulo 3 Marco aplicativo 27](#_Toc27104415)

[Delineación de la investigación 27](#_Toc27104416)

[Entrevistas 27](#_Toc27104417)

[Objetivo 27](#_Toc27104418)

[Alcance 27](#_Toc27104419)

[Tipos de entrevista 27](#_Toc27104420)

[Personas o cargos a entrevistar 27](#_Toc27104421)

[Formato de entrevista 28](#_Toc27104422)

[Objetivo 28](#_Toc27104423)

[Tipos de observación 28](#_Toc27104424)

[Procesos a observar 28](#_Toc27104425)

[Instrumentos para la levantar la información 30](#_Toc27104426)

[Product Backlog 33](#_Toc27104427)

[Equipo investigador 33](#_Toc27104428)

[Planificación 33](#_Toc27104429)

[Evaluación 34](#_Toc27104430)

[Construcción de la base de datos 35](#_Toc27104431)

[Procesamiento de la información 35](#_Toc27104432)

[Objetivo 36](#_Toc27104433)

[Requerimientos 36](#_Toc27104434)

[Índice tentativo o esquema de contenido 38](#_Toc27104435)

[Planificación 39](#_Toc27104436)

[Esquema de contenido 39](#_Toc27104437)

[Costos 39](#_Toc27104438)

[Cronograma 39](#_Toc27104439)

[Glosario 40](#_Toc27104440)

[Fichas bibliográficas 44](#_Toc27104441)

# Capítulo 1 Marco Introductorio

### Introducción

En los últimos años desde la llegada de los smartphones se han creado cientos de aplicaciones que nos ayudan en nuestro día a día, gracias a que ahora la mayoría de los dispositivos móviles cuentan con GPS, conexión a internet y un sistema operativo inteligente. Podemos observar como ejemplo la aplicación Uber, la cual ha revolucionado la manera de pedir un taxi por su facilidad y sencillez.

El objetivo principal del presente trabajo es desarrollar una aplicación que permite poder realizar reservas a cualquier salón de belleza o barbería de una manera rápida y sencilla y a su vez gestionar las reservas en los salones, sus clientes y su flujo de caja.

Para ello se decidió desarrollar una aplicación móvil con la tecnología React Native la cual fue creada por Facebook y es utilizado por grandes empresas como ser Airbnb, Instagram, Skype la cual nos permitirá desarrollar aplicaciones para las principales tiendas de aplicaciones (Play Store, App Store) para que así todas las personas puedan utilizarlo. React Native desde su salida en el año 2015 ha tenido una muy buena acogida en el mercado convirtiéndose en el principal framework para desarrollar aplicaciones hibridas, una de las razones de su gran acogida es que es respaldada por Facebook y tiene una gran comunidad y mucha documentación.

### Antecedentes

En Bolivia, en los últimos 5 años el sector económico de servicios de belleza ha aumentado un 80%. En el año 2010 existían tan solo 100 empresas registradas bajo la función económica de servicios de belleza. A pesar de la modernización de muchos mercados debido a la tecnología, como ser pedir un taxi o pedir comida, en el sector de los salones de belleza este avance no se ha hecho evidente esta modernización, no existe una alternativa para que las personas puedan fácilmente ver los salones de belleza a su alrededor, sus servicios y realizar una reserva. Por lo cual las personas les resulta complicado realizar una reserva, esperan horas y no pueden ver sus alternativas. A su vez, los salones de belleza no cuentan con herramientas eficaces para poder gestionar sus clientes, reservas y flujo de caja.

### Problema

El mundo ha evolucionado, las personas están ahora acostumbradas a realizar sus tareas o acciones desde internet, a su vez los sectores económicos se han adaptado para poder ofrecer esta posibilidad a las personas, un claro ejemplo es la banca por internet, que permite realizar muchas operaciones sin ir al banco. Sin embargo, los salones de belleza no cuentan con una herramienta para que las personas puedan realizar reservas y ver sus alternativas.

### Planteamiento del problema

La mayoría de las personas asiste a salones de belleza o barberías regularmente, ya sea para un simple corte de cabello en el caso de las mujeres para realizarse la manicura o pedicura, pero muchas veces se encuentran con el problema de largas esperas y además que su estilista favorita no está trabajando ese día, lo cual genera una gran insatisfacción en las personas. Además, muchas veces no pueden ver sus alternativas ya que no tienen un listado de los salones de belleza que ofrecen los servicios requeridos, lo cual podría darles una mejor experiencia.

### Problemas específicos

* Las personas no tienen una herramienta de fácil uso para poder realizar reservas en salones de bellezas, escoger su estilista y su servicio.
* Las personas no tienen un listado de salones de belleza que se adecúe a sus necesidades, para así poder escoger y ver mejores alternativas.
* La mayoría de los salones de belleza no cuentan con una base de datos de clientes, para así poder generar información relevante para la toma de decisiones.
* Los salones de belleza no cuentan con una manera eficiente para poder registrar sus flujos de caja.

### Formulación del problema

¿Por qué realizar largas esperas en los salones de belleza, cuando desde tu celular podrías realizar una simple reserva con tu estilista favorito?

### Objetivos

### Objetivos General

Desarrollar una aplicación móvil para dispositivos Android y IOS para poder ubicar salones de belleza cercanos y realizar reservas. A su vez, desarrollar un sistema web para los salones de belleza para poder gestionar estas reservas, registrar a sus clientes y registrar su flujo de caja.

### Objetivos específicos

* Determinar modelo de dominio y de negocio a través de la recopilación de todos los procesos e información de la barbería Urban Style como empresa piloto para las pruebas.
* Analizar y seleccionar la plataforma en la cual se desarrollará la aplicación móvil y el sistema web para los salones de belleza.
* Diseñar la base de datos en la que estará soportada toda la información de los clientes, reservas y los flujos de caja.
* Desarrollar un plan de trabajo utilizando como referencia la metodología de desarrollo Scrum.
* Desarrollar módulos del prototipo para la aplicación móvil y el sistema web de los salones de belleza en las plataformas seleccionadas.
* Desarrollar interfaz de la aplicación móvil.
* Validar la usabilidad y rendimiento de la aplicación y de la página web.

### Delimitación

### Delimitación espacial

El trabajo se realizará en la ciudad de Santa Cruz, Bolivia, tomando como empresa piloto para las pruebas a la barbería Urban Style con NIT 7804610013 ubicada en la misma localidad.

### Delimitación temporal

El trabajo iniciara el 10/01/2020 concluyendo el 31/03/2020, teniendo así una duración de casi 3 meses/

### Delimitación sustantiva.

Las tecnologías, arquitecturas, patrones de diseño y metodologías de desarrollo de software a analizar y utilizar serán las siguientes

* Arquitecturas: cliente servidor.
* Patrones de diseño: MVC, FLUX
* Lenguajes de programación: Ruby, Python, JavaScript
* Frameworks de desarrollo backend: Ruby on Rails
* Base de datos: NoSQL, SQL
* Frameworks de desarrollo móvil: React native, Flutter, Ionic.
* Metodologías de desarrollo de software: RUP, Scrum

### Justificación

### Justificación teórica

La justificación teórica de este trabajo es contrastar las herramientas actuales que se tienen para realizar desarrollo móvil y desarrollo web, junto también con las metodologías de desarrollo de software actuales.

### Justificación práctica

Este trabajo se realiza por la necesidad de mejorar la experiencia de las personas al asistir a un salón de belleza o barbería, la cual viene a ser casi una necesidad básica, con el uso de una aplicación móvil.

### Justificación social

La justificación social de este trabajo es mejorar los medios de accesos a salones de belleza generando así un mejor ambiente para investir en negocios de este sector y además generar las bases para desarrollar un software completo que se pueda encargar de otras necesidades fundamentales que no se han tomado en cuenta, como por ejemplo rentar bicicletas o alquilar computadoras.

### Justificación individual

La justificación individual es la de generar una herramienta que facilitará las vidas de las personas y será usa masivamente.

### Estrategia metodológica.

Para llevar a cabo esta investigación, se ha tomado como referencia la investigación científica, en la cual se tendrá presente cuatro elementos principales en el proceso de desarrollo de la misma, los cuales son mencionados a continuación:

* **Sujeto** El investigador del presente proyecto
* **Objeto** Desarrollo de una aplicación “Realzar reservas en salones de belleza”.
* **Medio** El medio a utilizar serán las siguientes herramientas.
  + Ruby on Rails
  + React JS
  + React Native
  + Scrum

* **Fin**: Demostrar como una aplicación para realizar reservas mejoraría la experiencia de los clientes al ir a un salón de belleza, y como lo reduce a solo pocos pasos.

Dentro de este tipo de investigación encontramos la investigación aplicada que es la que se ha utilizado en el proceso de desarrollo de presente trabajo.

### Investigación Aplicada

Se ha tomado este tipo de investigación, ya que se desarrolla una propuesta viable aprovechando la aportación de la teoría encontrada con respecto al tema investigado llevándolo a la práctica, en la cual se obtiene como resultado una aplicación para realizar reservas en salones de bellezas, a los diferentes usuarios los mismos que pueden tener acceso a la información de manera rápida y eficaz.

### Demostración (Factibilidad del Proyecto)

EL proyecto se base en las reservas a los salones de belleza, el mismo se declara como un proyecto factible, que se podría aplicar también en otras áreas como reservas de citas médicas, psicológicas según cual sea el fin de lo que se quiere lograr.

En el proceso desarrollo de la aplicación utilizaremos diferentes servicios, como GPS, mapas o notificaciones. Utilizaremos información recopilada de las fuentes de información que en este caso han sido de la empresa Google, además de utilizar otras herramientas de diferentes empresas.

# Capítulo 2 Marco teórico

### Marco contextual

El presente trabajo se realizará en la ciudad de Santa Cruz de la Sierra ubicada en la provincia de Andrés Ibáñez en el departamento de Santa Cruz, Bolivia. Esta ciudad ha contado con un crecimiento de sector económico de salones de belleza, sobre todo en el área de barberías. En el año 2015 la ciudad de Santa Cruz presentaba un total de 100 empresas registradas bajo el concepto de “Servicios de belleza”, en el año 2019 se ha evidenciado un crecimiento del 200 % de este sector contando ahora con 300 empresas registrabas bajo el mismo concepto, por lo que se consideró un lugar ideal para realizar este trabajo.

La empresa Urban Style con NIT 7804610013 ubicada en la ciudad de Santa Cruz de la Sierra, será tomada como empresa para prueba piloto y recopilación de datos para esta investigación. Esta empresa se encuentra vigente en el mercado desde el 8 de septiembre de 2017, contando con más de 2 años de experiencia en el rubro, además de ser considerada como una de las barberías más prestigiosas de la ciudad al ser elegida como barbería modelo para la empresa Unilever Andina S.A para la promoción de los productos AXE. La empresa Urban Style es una empresa unipersonal, que cuenta con 2 sucursales. Su casa matriz se encuentra sobre el tercer anillo externo, calle las Liras de la ciudad de Santa Cruz de la Sierra. Su sucursal se encuentra ubicado en el centro de la ciudad sobre la calle Velasco con número 56 entre las calles Ingavi y Suarez de Figueroa.

La empresa Urban Style cuenta con un total de siete empleados de planta, tres operarios estilistas profesionales, dos supervisores de sucursal, un gerente general, junto con un asesor de marketing serializado, los cuales serán nuestra principal fuente para recolección de información sobre las necesidades de los salones de belleza para las reservas, información relevante de los clientes y funcionamiento de su flujo de caja.

## Marco conceptual

### Metodologías de desarrollo de software

#### 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.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita (METODOSS, 2019)obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

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.

(Proyectos ágiles, 2019)

#### RUP (Proceso unificado racional)

La metodología RUP, abreviatura de Rational Unified Process (o Proceso Unificado Racional), es un proceso propietario de la ingeniería de software creado por Rational Software, adquirida por IBM, ganando un nuevo nombre Irup que ahora es una abreviatura Rational Unified Process y lo que es una marca en el área de software, proporcionando técnicas que deben seguir los miembros del equipo de desarrollo de software con el fin de aumentar su productividad en el proceso de desarrollo.

La metodología RUP utiliza el enfoque de la orientación a objetos en su diseño y está diseñado y documentado el uso de la notación UML (Unified Modeling Language) para ilustrar los procesos en acción. Utiliza técnicas y prácticas probadas comercialmente.

Es un proceso considerado pesado y preferentemente aplicable a grandes equipos de desarrollo y grandes proyectos, pero el hecho de que es ampliamente personalizable que permite adaptarse a proyectos de cualquier escala.

Para la gestión del proyecto, la metodología RUP proporciona una solución disciplinada como las tareas y responsabilidades señaladas dentro de una organización de desarrollo de software.

RUP es, en sí, un producto de software. Es modular y automatizado, y toda su metodología se apoya en varias herramientas de desarrollo integradas y vendidos por IBM a través de sus «Suites racional.»

(METODOSS, 2019)

El RUP o proceso unificado suele usarse más para proyectos muy grandes en los cuales se necesita generar mucha documentación sobre el producto a terminar. Sin embargo, la metodología scrum es una de las metodologías más utilizadas actualmente por su alta flexibilidad y porque proporciona rapidez para desarrollar software y llevar un control sobre los estándares de calidad. A continuación, mostraremos un gráfico donde compararemos las metodologías SCRUM y RUP para así poder detectar las ventajas y desventajas de cada una.

|  |  |  |
| --- | --- | --- |
|  | RUP | SCRUM |
| Enfoque | Iterativo | Iterativo |
| Ciclo | Ciclo formal se define a través de 4 fases, pero algunos flujos de trabajo pueden ser concurrentes. | Cada sprint (iteración) es un ciclo completo. |
| Planificación | Plan de proyecto formal, asociada a múltiples iteraciones, se utiliza. El plan es impulsado fecha final y también cuenta con hitos intermedios. | No de extremo a extremo del plan del proyecto. Cada plan de la siguiente iteración se determina al final de la iteración actual (no la fecha final de tracción). Dueño del Producto (usuario de negocios clave) determina el momento en que el proyecto se lleva a cabo. |
| Alcance | Ámbito de aplicación está predefinido antes del inicio del proyecto y se documenta en el documento de Alcance. Ámbito de aplicación pueden ser revisados ​​durante el proyecto, los requisitos se están aclarando, pero estas modificaciones están sujetas a un procedimiento estrictamente controlado. | En vez de alcance, SCRUM utiliza una Cartera de Proyectos, que se reevaluado al final de cada iteración (sprint). |
| Los artefactos | Visión / Ámbito de aplicación del documento, el paquete formal de requisitos funcionales, documento de arquitectura del sistema, plan de desarrollo, plan de pruebas, scripts de prueba, etc. | El único artefacto formal es el software operativo. |
| Tipo de proyecto / producto | Recomendado para grandes, a largo plazo, a nivel de empresa con proyectos a medio y alta complejidad. | Recomendado para las mejoras rápidas y organizaciones que no dependen de una fecha límite. |

#### Conclusiones

La teoría a utilizar en el proyecto será la metodología de desarrollo Scrum por las siguientes razones:

* Debido a que el proyecto a realizar tendrá requisitos cambiando constantemente
* Nos permite entregas de productos finalizados en cada etapa.
* Nos permite realizar una retroalimentación en cada etapa para así evitar cometer un error durante todo el proyecto.

El cliente puede ver el avance del proyecto más fácilmente

Las bases de datos han sido fundamentales en el desarrollo de sistemas para las organizaciones. Conocer y manipular bases de datos resulta útil para un sinfín de usos, desde procesos de negocio, como crear un startup o un sistema de información eficiente y productivo, hasta temas mucho más especializados como Big Data.

El propósito de las bases de datos surge con la necesidad de registrar y almacenar datos. Por muchos años la mejor forma de hacer esto consistía en un archivo de documentos en papel, pero pronto esto se volvió ineficiente, pues lo siguiente después de guardar datos es poderlos consultar fácilmente.

Hacer una búsqueda puede ser un proceso relativo, pues si tienes muchos tipos de datos podrías querer buscar a partir de cualquiera de ellos. Esto es lo que fundamenta la teoría de bases de datos. Imagina tener un archivo gigante de historias clínicas en un hospital, y que no haya una forma fácil de encontrar la de un paciente específico. Si no existe un proceso de indexación de la información podríamos pensar en buscar las historias de muchas maneras y tardar mucho en encontrar.

Actualmente existen diferentes tipos de bases de datos que permiten hacer consultas y escalar rápidamente.

(https://platzi.com/blog/bases-de-datos-que-son-que-tipos-existen/, 2019)

#### ¿Qué es un dato?

Un dato nos permite describir un objeto. Dicho objeto podemos llamarlo entidad, por ejemplo, una casa en la que viven personas. La casa es la entidad y la cantidad de personas que viven en la casa son un dato, que en este caso es numérico.

Hay diferentes tipos de datos que se pueden tener en una base de datos: caracteres, numéricos, imágenes, fechas, monedas, texto, bit, decimales y varchar. Hay bases de datos más o menos adecuadas según el tipo de dato, y hay un proceso para convertir un dato en información, pues un solo dato por sí mismo no representa nada si no lo vemos en contraste con otros. Así se identifican comportamientos.

Esto es solo un poco de lo que puedes aprender en el Curso de Fundamentos de Bases de Datos. Las empresas y las personas necesitan de las bases de datos en su vida diaria. Las utilizamos más de lo que nos imaginamos, cada vez que usamos el teléfono, hacemos una compra o cuando hacemos una transacción bancaria. Si estás empezando un proyecto nuevo o vas a aprender a programar vale la pena tener muy claro el funcionamiento de las bases de datos.

#### Tipos de base de datos

Hay bases de datos relacionales, como MySQL, SQL Server y Oracle. Como su nombre lo indica utilizan el modelo relacional y siempre es mejor usarlas cuando los datos son consistentes y ya tienes algo planificado.

También existen las no relacionales, como MongoDB y Redis, conocidas como NO-SQL (NoSQL). Estas son más flexibles en cuanto a consistencia de datos y se han convertido en una opción que intenta solucionar algunas limitaciones que tiene el modelo relacional.

Además, hay otras BBDD no tan tradicionales, como las basadas en grafos o aquellas que tienen información cartográfica, que pueden servir, por ejemplo, si estás creando un e-commerce para encontrar relaciones entre los productos y las preferencias de los usuarios.

Tener un buen diseño de base de datos desde el comienzo te puede ayudar a ahorrar tiempo. Las bases de datos relacionales y no relacionales se organizan de formas diferentes y trabajan con tipos de datos distintos, así que es importante entender cómo se diseña cada una. En el caso de bases de datos relacionales se trabaja con el estándar SQL, que se usa para actualizar o recuperar datos.

Un ejemplo claro para entender las bases de datos es el funcionamiento de una aerolínea. Todas las aerolíneas tienen aplicaciones para hacer reservas y trabajan con diferentes rutas. Todas las personas que van a viajar tienen que proveer datos para hacer una reservación de una ruta específica, por ejemplo, la fecha en que van a viajar. Para esto se hace necesario un repositorio en el que se pueda almacenar esta información y se pueda cruzar, además, con una cantidad de sillas por avión. Una base de datos permite no redundar en los datos.

(https://platzi.com/blog/bases-de-datos-que-son-que-tipos-existen/, 2019)

## Bases de datos relacionales (SQL)

Un base de datos relacional es una recopilación de elementos de datos con relaciones predefinidas entre ellos. Estos elementos se organizan como un conjunto de tablas con columnas y filas. Las tablas se utilizan para guardar información sobre los objetos que se van a representar en la base de datos. Cada columna de una tabla guarda un determinado tipo de datos y un campo almacena el valor real de un atributo. Las filas de la tabla representan una recopilación de valores relacionados de un objeto o entidad. Cada fila de una tabla podría marcarse con un identificador único denominado clave principal, mientras que filas de varias tablas pueden relacionarse con claves extranjeras. Se puede obtener acceso a estos datos de muchas formas distintas sin reorganizar las propias tablas de la base de datos.

(Amazon Web Services, 2019)

## Bases de datos no relacionales (NO SQL)

Un dato nos permite describir un objeto. Dicho objeto podemos llamarlo entidad, por ejemplo, una casa en la que viven personas. La casa es la entidad y la cantidad de personas que viven en la casa son un dato, que en este caso es numérico.

Hay diferentes tipos de datos que se pueden tener en una base de datos: caracteres, numéricos, imágenes, fechas, monedas, texto, bit, decimales y varchar. Hay bases de datos más o menos adecuadas según el tipo de dato, y hay un proceso para convertir un dato en información, pues un solo dato por sí mismo no representa nada si no lo vemos en contraste con otros. Así se identifican comportamientos.

Esto es solo un poco de lo que puedes aprender en el Curso de Fundamentos de Bases de Datos. Las empresas y las personas necesitan de las bases de datos en su vida diaria. Las utilizamos más de lo que nos imaginamos, cada vez que usamos el teléfono, hacemos una compra o cuando hacemos una transacción bancaria. Si estás empezando un proyecto nuevo o vas a aprender a programar vale la pena tener muy claro el funcionamiento de las bases de datos.

Las bases de datos NoSQL utilizan una variedad de modelos de datos para acceder y administrar datos, como documentos, gráficos, clave-valor, en-memoria y búsqueda. Estos tipos de bases de datos están optimizados específicamente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles, lo que se logra mediante la flexibilización de algunas de las restricciones de coherencia de datos en otras bases de datos.

(Amazon Web Services, 2019)

#### Conclusiones base de datos

Para el proyecto se utilizará la base de datos SQL por las siguientes razones:

* Comunidad más amplia, llevan más tiempo en el mercado
* Facilidad para conseguir recursos
* Estructura solida
* Seguridad en operaciones transaccionales

Debido a que las bases de datos SQL llevan más tiempo en el mercado, por lo que cuentan con una comunidad de soporte más grande y es más fácil de conseguir recursos. Además, proveen una estructura sólida a pesar de restringida de los datos lo cual garantiza la seguridad en las operaciones transaccionales. A pesar de que las bases de datos SQL, cuentan con problemas a la hora de encontrar escalabilidad es la alternativa correcta para este proyecto, para poder guardar la información de una manera correcta y relacionada evitando la duplicidad y garantizar la seguridad de los datos.

#### Arquitectura cliente servidor

La arquitectura cliente servidor tiene dos partes claramente diferenciadas, por un lado, la parte del servidor y por otro la parte de cliente o grupo de clientes donde lo habitual es que un servidor sea una máquina bastante potente con un hardware y software específico que actúa de depósito de datos y funcione como un sistema gestor de base de datos o aplicaciones.

En esta arquitectura el cliente suele ser estaciones de trabajo que solicitan varios servicios al servidor, mientras que un servidor es una máquina que actúa como depósito de datos y funciona como un sistema gestor de base de datos, este se encarga de dar la respuesta demandada por el cliente.

Esta arquitectura se aplica en diferentes modelos informáticos alrededor del mundo donde su propósito es mantener unas comunicaciones de información entre diferentes entidades de una red mediante el uso de protocolos establecidos y el apropiado almacenaje de la misma.

(Infranetworking, 2019)

#### Framework

En el desarrollo de Software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, librerías y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

#### Framework de backend

Siguiendo la arquitectura cliente servidor la cual es la más aceptada y utilizada en el desarrollo de software, necesitamos escoger un framework para desarrollar nuestro programa de servidor que administrará los datos y permitirá interactuar a los clientes con ellos, para ello utilizaremos una tabla comparativa con los frameworks de backend más utilizados, recomendados y avalados por la comunidad de desarrollo de software.

|  |  |  |
| --- | --- | --- |
|  | Django | Ruby on Ralis |
| Lenguaje de programación | Python | Ruby |
| Patrón de diseño | MVC | MVC |
| Popularidad | Alta | Media |
| Flexibilidad | Media | Alta |
| Velocidad | Toma más tiempo desarrollar un proyecto base y empezar el desarrollo. | Mas rápido de desarrollar en nuevos proyectos |
| Fecha de lanzamiento | 2012 | 2004 |
| Estrellas en GitHub | 4.67 k | 2.37 k |
| Filosofía | DRY (no te repitas a ti mismo) | Convención sobre configuración |
| Librerías disponibles | Media | Alta |
| Comunidad | Alta | Media |

Ambos frameworks de desarrollo backend son muy utilizados y cuentan con una gran comunidad. Además, cada uno cuenta con su propio lenguaje de programación en el caso de Django Python y en el caso de Rails el lenguaje Ruby, ambos lenguajes orientado a objetos y lenguajes no compilados. Ambos frameworks manejan diferentes tipos de base de datos SQL. Sin embargo, debido a que Ruby on Rails cuenta con más tiempo en el mercado y es más rápido al comenzar el desarrollo de un proyecto, además por su gran cantidad de librerías para desarrollar más rápido.

#### Framework de frontend

Siguiendo la arquitectura cliente servidor, necesitamos una aplicación de cliente que nos permita interactuar con nuestro servidor, en este caso lo haremos a través de una aplicación web porque esto nos supone las siguientes ventajas.

* No necesita instalación
* Se puede acceder desde cualquier lugar del mundo
* Se puede adaptar a los diferentes dispositivos celulares, tabletas y laptops o computadoras.

Para realizar la arquitectura de los clientes se tienen muchas opciones, entre las más recomendadas y utilizadas tenemos los sistemas web multi páginas y los sistemas web de una sola página.

Multi paginas

* Las ventajas de este tipo de desarrollo de páginas web son las siguientes

Permite desarrollar más rápido

* Permite trabajar directamente desde el framework de backend, lo cual supone ventajas en validaciones y verificar errores.

Algunas de sus desventajas son las siguientes

* Lentos al traer los datos, cargan todos los datos cada vez que traen datos
* Mala experiencia de usuario debido a su lentitud

Sobre los sitios web de una sola página o “Single page apps” tenemos las siguientes ventajas.

* Rápido, solo trae los datos que necesita, nunca recarga la página completa lo cual suele ser muy perjudicial.
* Mejor experiencia de usuario, permite dar más funcionalidades y hacer las cosas más rápidas.

Algunas de sus desventajas pueden ser las siguientes

* Se necesita desarrollar un api para poder conectar a los datos con el backend
* Se necesita más trabajo para realizar funcionalidades como autenticación y búsquedas.

Teniendo en cuenta que la experiencia de usuario es uno de los factores más importantes, para el desarrollo de este proyecto se utilizara una página web del tipo una sola página o “Single page app”.

Existe una variedad de frameworks para desarrollar páginas web de una sola página, analizaremos las más utilizadas, para poder tener un análisis objetivo y poder aplicarla en nuestro proyecto.

#### React

React es una biblioteca escrita en JavaScript, desarrollada en Facebook para facilitar la creación de componentes interactivos, reutilizables, para interfaces de usuario. Se utiliza en Facebook para la producción de componentes, e Instagram está escrito enteramente en React. Uno de sus puntos más destacados, es que no sólo se utiliza en el lado del cliente, sino que también se puede representar en el servidor, y trabajar juntos.

React.js está construido en torno a hacer funciones, que toman las actualizaciones de estado de la página y que se traduzcan en una representación virtual de la página resultante. Siempre que React es informado de un cambio de estado, vuelve a ejecutar esas funciones para determinar una nueva representación virtual de la página, a continuación, se traduce automáticamente ese resultado en los cambios del DOM necesarios para reflejar la nueva presentación de la página.

(Caballero, 2019)

#### Angular

AngularJS es un framework para desarrollo web construido por Google e inicialmente liberado en 2010. Angular (a secas) también es un framework para desarrollo web, y también construido por Google (y de hecho por varios de las mismas personas que hicieron AngularJS). Lo que hoy se conoce como “Angular” fue inicialmente llamado “Angular 2” ya que era visto como la siguiente versión de Angular 1.x (lo que hoy es llamado AngularJS). Sin embargo, dado que no son compatibles, y que Angular 2 tiene un alcance más amplio, se decidió mantener la rama 1.x de Angular bajo el nombre AngularJS mientras que la rama nueva se quedó simplemente como Angular.

La versión inicial de esta nueva encarnación de Angular fue liberada en septiembre del 2016 y cada 6 meses se libera una versión mayor nueva bajo un esquema de versionado semántico (ej. 1.2.3-beta.1 se refiere a 1=versión mayor posiblemente con breaking changes, 2= cambios menores, 3 fixes, meta.1 pre-release). Al escribir este artículo, la versión estable más reciente es la 5.2.3 y la próxima versión ya se encuentra en 6.0.0-beta2.

La programación en Angular se hace usando TypeScript, un lenguaje que es un superconjunto de JavaScript que agrega capacidades de tipado estático. Esto nos da la ventaja de poder tipar cosas como variables, funciones, devoluciones, además de poder crear Interfaces. TypeScript también nos da la capacidad de usar enumerators, modules, namespaces, decorators y generics. Y, por último, pero no por ello menos importante está sistema de import, que vamos a utilizar a diario para atomizar y modularizar todo nuestro código.

(Cano, 2019)

#### Tabla comparativa

|  |  |  |
| --- | --- | --- |
|  | Angular | React |
| Es un | Framework | Liberaría |
| Flexibilidad | Poco flexible | Muy flexible |
| Avalado | Avalado por Google | Avalado por Facebook |
| Framework para móvil | Framework para aplicaciones móviles Ionic | Framework para aplicaciones móviles React Native |
| Experiencia de usuario | Media | Alta |
| Comunidad | Alta | Alta |

#### Conclusiones framework front-end

La tecnología a utilizar para el front-end será la librería React JS desarrollada por Facebook, esta librería se adecua mas al proyecto por las siguientes razones:

* Configurabilidad, es una librería solo para vistas lo que nos permite escoger toda la demás parte de nuestro proyecto como ser, capa de acceso a datos, peticiones HTTP.
* Popularidad, actualmente React es una de las tecnologías mas buscadas en Google y Stackoverflow, lo cual no da una buena señal sobre su comunidad y la ayuda a la que podamos obtener.
* Utilizaremos React porque su framework para desarrollo de aplicaciones (React native) nos permite desarrollar aplicaciones totalmente nativas que funcionan en ambas plataformas IOS y Android utilizando solo JavaScript, a diferencia de Angular que su framework para aplicaciones Ionic, no ofrece una experiencia nativa en las aplicaciones, sino es una pagina web corriendo en un container para aplicaciones.

Cualquiera de estos framework nos ofrece una gran cantidad de herramientas y desarrollar páginas web rápidas y fáciles de utilizar. Cualquiera es una buena opción ya que ambas son avaladas por grandes empresas, en su caso React es avalado por Facebook y Angular es avalado por Google. Sin embargo, ya que el framework de React que es React Native ofrece una mejor experiencia de usuario para las aplicaciones, utilizaremos esta tecnología que nos da Facebook para así ofrecer la mejor experiencia a nuestros usuarios.

Ahora que ya tenemos nuestro framework para realizar el backend, junto con nuestro framework para poder realizar las páginas web , ahora nos falta un tema más que sería nuestro framework para desarrollar la aplicación móvil para realizar la aplicación para que los clientes pueden hacer sus reservas las cuales serán atendidas en la página web, para ello tenemos dos opciones, aplicaciones nativas y aplicaciones hibridas, necesitamos hacer una análisis de estas dos opciones para ver sus ventajas y desventajas y decidir cuál es el mejor para el proyecto.

Ambos frameworks utilizados para desarrollar páginas web de una sola página, cualquiera es una buena elección debido a que cada una está resguardada por grandes empresas en el case de “React” está respaldada por Facebook y en el caso de Angular está respaldada por Google.

#### Aplicaciones nativas

La aplicación nativa está desarrollada y optimizada específicamente para el sistema operativo determinado y la plataforma de desarrollo del fabricante (Android, iOS, etc.).

Este tipo de aplicaciones se adapta al 100% con las funcionalidades y características del dispositivo obteniendo así una mejor experiencia de uso. Sin embargo, el desarrollo de una aplicación nativo comporta un mayor coste, puesto que si se desea realizar una aplicación multiplataforma se ha de realizar una nueva versión para cada sistema operativo, multiplicando así los costes de desarrollo.

#### Aplicaciones hibridas

La aplicación web es la opción más sencilla y económica de crear aplicaciones, puesto que al desarrollar una única aplicación se reducen al máximo los costes de desarrollo. Asimismo, en este tipo de aplicaciones, puede utilizarse el “responsive web design”, creando así una única aplicación adaptada para todo tipo de dispositivos. Por el contrario, la aplicación web ofrece una peor experiencia de uso, puesto que ignora las características del dispositivo y una menor seguridad ya que depende de la seguridad que ofrezca el propio navegador.

|  |  |  |
| --- | --- | --- |
| **Tipos de aplicaciones** | **Nativas** | **Hibridas** |
| Lenguaje | Java, Kotlin, Swift y Objective C | JavaScript, C#, Typescript |
| Costo de Desarrollo | Alto | Bajo |
| Interfaz de usuario | Alta | Alta |
| Rendimiento | Alto | Medio |
| Multiplataforma | No | Si |

Al realizar este análisis podemos darnos cuenta de que la principal diferencia de las aplicaciones hibridas con las nativas es el costo de desarrollo. Debido a que las aplicaciones hibridas funcionaran para ambas tiendas Playstore y AppStore y solo tenemos que programarlo una vez, supone una gran ventaja. Sin embargo, al ser hibrido su rendimiento nunca llegara a ser tan alto como el de una aplicación nativa, pero los celulares cada día son más rápidos y esta diferencia se esta volviendo mínima, por lo que ahora es considerada la mejor opción por muchas empresas, y seria al indicada para el proyecto para así poder avanzar más rápido y poder dar partes de las aplicaciones terminadas más rápidamente para someterlas a pruebas.

# Capítulo 3 Marco aplicativo

### Delineación de la investigación

Los instrumentos para realizar la investigación serán marcados por la metodología Scrum mencionada en el anterior capitulo, esta metodología nos ofrece diferentes roles al momento de desarrollar software, en este caso el encargado de obtener los requisitos, información del cliente es el Product Owner.

Siguiendo la metodología Scrum, el Product Owner es la persona que más sabe el proyecto y su función es trasmitir los deseos del cliente al equipo y priorizar tareas. Para ello se utiliza las entrevistas y observación como métodos para obtener información.

#### Entrevistas

## Objetivo

* Conocer el modelo de dominio y modelo de negocio de la empresa
* Obtener las historias de usuario

## Alcance

El alcance del proyecto se mantendrá según lo especificado para cada módulo. Sin embargo, de acuerdo a la planificación de cada módulo se podrán modificar funcionalidades internas que se adecuen mejor al modelo de negocio de la empresa, pero sin afectar el alcance de cada modulo , para así evitar salir del alcance establecido.

#### Planificación de las entrevistas

Se realizarán tres tipos de entrevistas.

* Recopilar información base sobre la empresa, determinar el modelo de dominio y de negocio, optimizar las preguntas o tiempo de la entrevista.
* Realizar una investigación profunda sobre el modulo que se quiera realizar antes de su desarrollo
* Entrevista sobre la implementación de cada modulo realizando una evaluación de calidad y satisfacción

#### Tipos de entrevista

Las entrevistas serán semi estructuradas en ambientes relajados (cafetería, caminata), para así permitir el flujo libre de las verdaderas ideas del cliente sobre su producto final, pero con un marco delimitado sobre el contenido a tomar en cada entrevista y los objetivos de cada entrevista. Las entrevistas se realizarán a las personas que ocupan los siguientes cargos en la empresa Urban Style.

## Características de la entrevista

* Entrevistas semi estructuradas, abiertas a respuestas de desarrollo y a preguntas que puedan salir de las mismas.
* Entrevistas con un solo tema.
* Entrevistas solo contaran con un máximo de 10 preguntas
* Duración de 20 a 45 minutos como máximo
* Las entrevistas se realizarán personalmente
* Las entrevistas se realizarán en un ambiente como y relajado, como ser una cafetería para así lograr un ambiente cómodo para el entrevistado que nos brindara su tiempo
* Las entrevistas tendrán una calificación sobre la información que se obtuvo del tema.

## Personas o cargos a entrevistar

* Gerente general
* Administrador de sucursal
* Personal Operativo (Barberos, Estilistas)
* Asesor de marketing

## Formato de entrevista

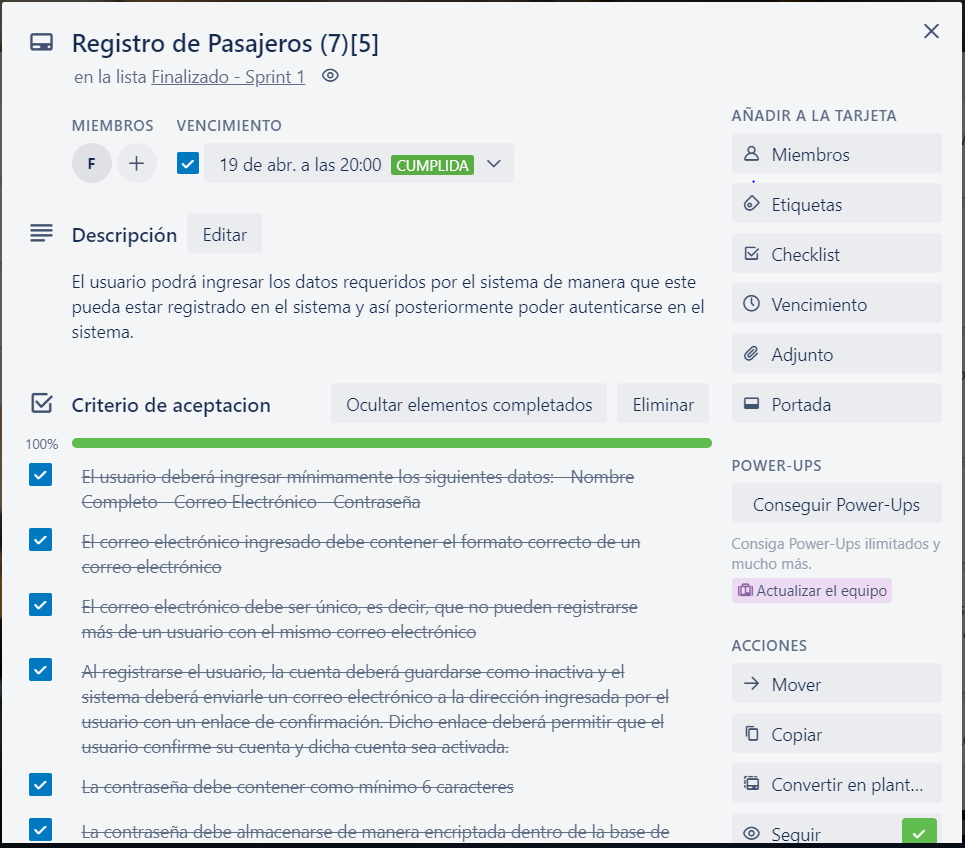
Las entrevistas serán semi estructuradas en ambientes relajados (cafetería, caminata), para así permitir el flujo libre de las verdaderas ideas del cliente sobre su producto final, pero con un marco delimitado sobre el contenido a tomar en cada entrevista y los objetivos de cada entrevista. Las entrevistas se realizarán a las personas que ocupan los siguientes cargos en la empresa Urban Style. El objetivo de la entrevista es obtener las historias de usuario sobre cada módulo o funcionalidad que se requiera para completar el alcance del proyecto.

Se realizarán múltiples entrevistas inicialmente para poder concretar los modelos de negocio y de dominio de esta empresa. Luego, se realizarán entrevistas cada finalización de sprint para realizar la retroalimentación sobre el trabajo realizado y revisar las historias de usuario para el siguiente sprint.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tema | | | Módulo de reservas | |
| Entrevistado | | | Gerente general Lic. Franz Bascope Jordán | |
| Fecha y hora | | | 03/12/2019 14:30 | |
| Lugar | | | Cafetería Starbucks, Ventura Mall | |
| Calificación (Máximo valor 100 puntos) | | |  | |
| Preguntas | Calificación de respuesta | Notas, Observaciones | |
| Existe alguna modalidad de reservas para los clientes | X |  | |
| Como funcionan las reservas | X |  | |
| Como se evita el choque de reservas | X |  | |
| Que pasa cuando un cliente esta esperando y llega un cliente con reserva a la misma hora | X |  | |
| Se registran las reservas de manera duradera | X |  | |
| Se registra la información del cliente para agilizar un futuro proceso | X |  | |
| Como calificaría su manejo de reservas actualmente | X |  | |
| Por qué canales recibe reservas | X |  | |
| Que aspectos sobre el manejo de reservas le gustaría mejorar | X |  | |

### Instrumentos para la levantar la información

La información obtenida de las entrevistas y de las visitas para observación serán transformadas en “historias de usuario” las cuales vendrán a ser nuestra principal guía para la planificación de los sprints. Las historias de usuario representan funcionalidades pequeñas que se obtuvieron de la información de la entrevista y que al juntarlas y priorizarlas nos generas nuestros planes de trabajo.



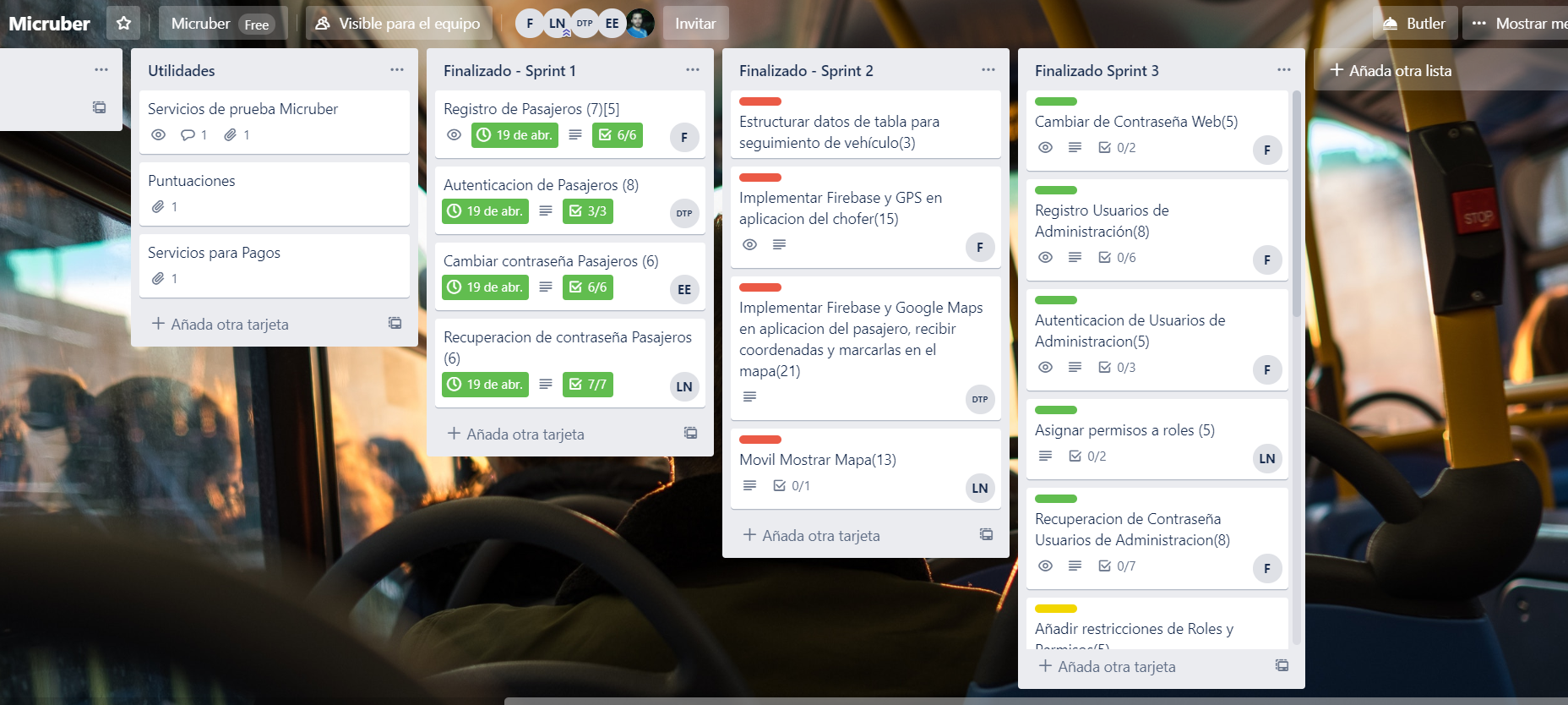
En este ejemplo de historia de usuario podemos identificar características esenciales para su efectividad,

* **Título:** nos da una breve descripción de lo que trata la historia
* **Descripción:** nos da una descripción a fondo y detallada sobre la historia de usuario.
* **Criterios de aceptación:** es un checklist que define los estándares de calidad para esta historia de usuario y los requerimientos para que se pueda aceptar como completada.
* Fecha de vencimiento: nos da una fecha tentativa sobre cuando la historia debería estar terminada.
* **Miembros:** nos permite especificar cuales integrantes del equipo son responsables de realizar y cumplir con los requerimientos de esta historia de usuario.
* **Adjunto:** En caso de que se necesita adjuntar alguna imagen, documento, o cualquier contenido que nos dé más información sobre la historia.
* **Puntaje:** El puntaje es la dificultad que tiene esta historia de usuario en medida al tiempo, esfuerzo y recursos. Para obtener este puntaje todos los miembros del equipo deben estar presentes y dar números de acuerdo a una escala previamente establecida sobre la dificultad de esta historia. Una vez asignado los números se hace un promedio y ese promedio vendría a ser el puntaje o dificultad de la historia de usuario
* **Prioridad:** la prioridad es una parte muy importante, porque nos permite colocar nuestras historias en los sprints correctos y así poder cumplir con el objetivo

Cada historia de usuario de acuerdo a su prioridad será asignada a un sprint en el cual se trabajará sobre la misma y al finalizar se realizará una retroalimentación sobre los resultados obtenidos.

El conjunto de las historias de usuario del proyecto nos dará nuestro Backlog, en el cual guardaremos toda la información organizada sobre los requisitos obtenido de los clientes, para gestionar esta información utilizaremos la herramienta web Trello, la cual nos facilita realizar la organización y el control sobre nuestro Backlog.

## Product Backlog



### Equipo investigador

El trabajo de campo se llevará a cabo solo por un investigador el cual se encargara de levantar los requisitos y plasmar la información en las historias de usuario, el investigador debe tener conocimiento sobre el desarrollo de software para poder realizar preguntas concretas para definir funcionalidades.

### Planificación

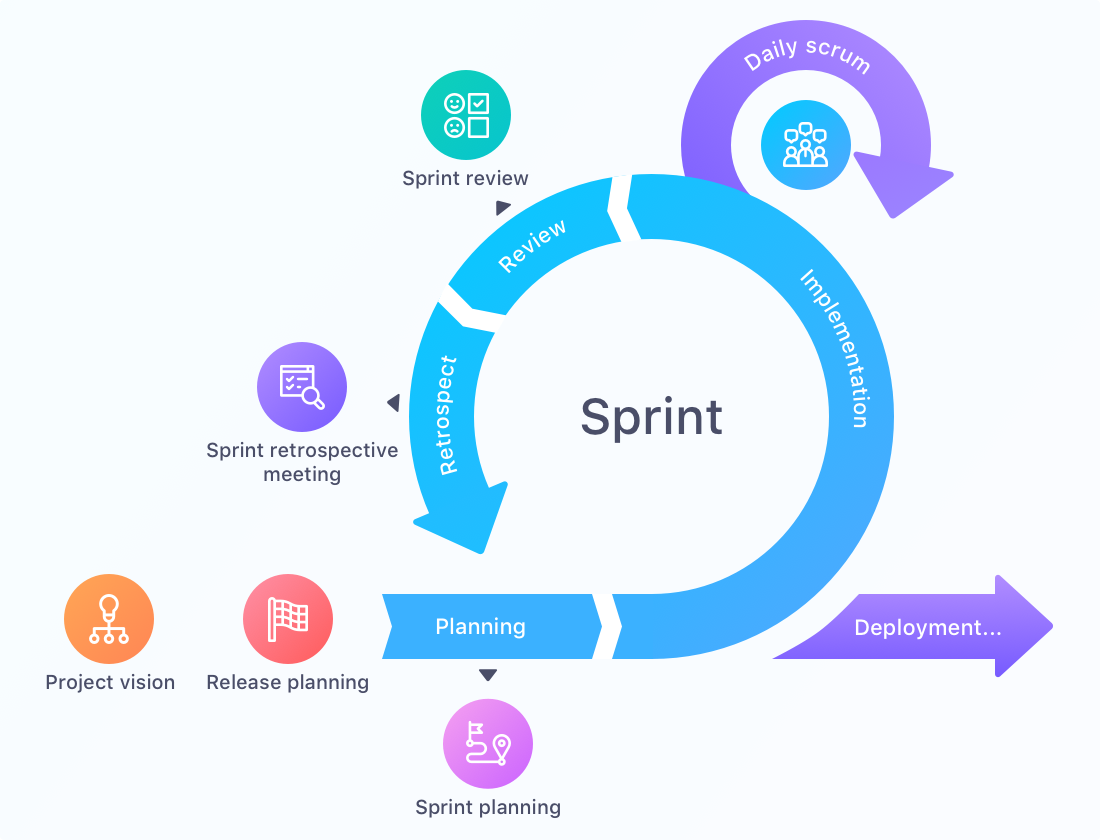
El trabajo de campo se realizará en distintas etapas, siendo la primera una etapa para reconocimiento a profundidad de la empresa y sus características para luego poder realizar entrevistas específicas sobre cada módulo a desarrollar.

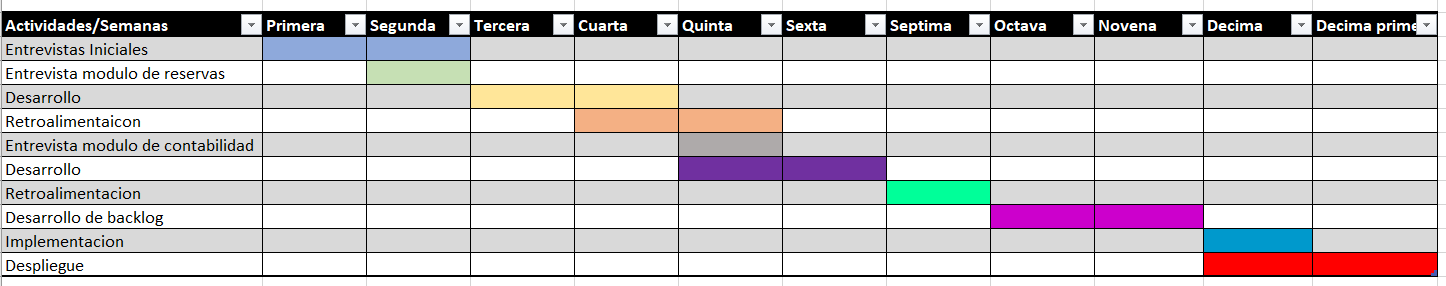
En cada etapa de desarrollo se realizarán subetapas:

* Planeación
* Desarrollo
* Presentación interna
* Correcciones
* Presentación Externa
* Aprobación

Cada una de estas etapas son mencionadas y recomendadas por la metodología scrum para así poder llevar un mejor control del proyecto y verificar la aprobación del cliente en cada iteración del proyecto para así evitar la redundancia

Es imprescindible realizar las entrevistas iniciales, porque son estas nuestra ruta clave ya que sin ellas no podemos realizar ninguna planeación y de esta manera no se podrá empezar el desarrollo de software



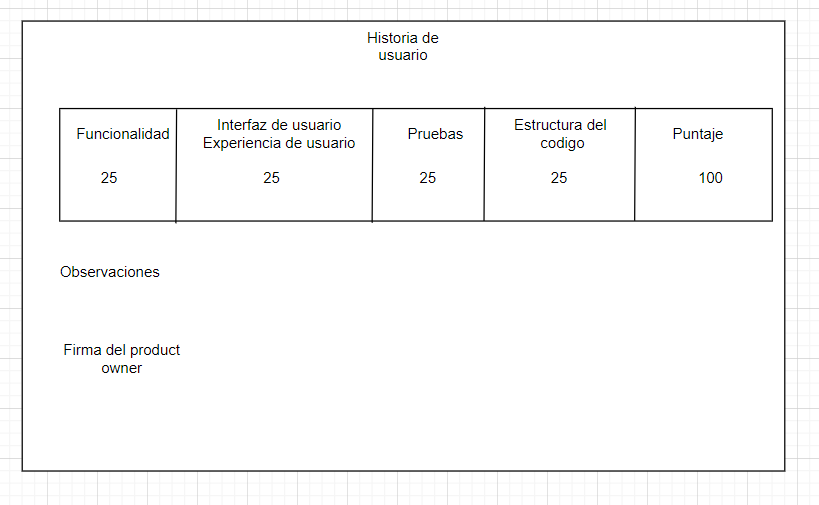


### 

### Evaluación

Debido a que el proyecto esta dividido en sprints o etapas de un periodo corto de no mas de 3 semanas, al finalizar cada sprint se realiza una retroalimentación sobre las tareas finalizadas, de esta manera podemos revisar todas las tareas sin esperar al final del producto, esto es muy útil debido a que nos permite corregir errores desde su inicio , cuando su implicación en el sistema no es de gran magnitud, cuando no se realizan estas evaluaciones continuas generalmente se encuentran errores que afectan todo el sistema y son muy difíciles de cambiar.

Como herramienta para realizar a la evaluación se usarán los mismos criterios de aceptación definidos al momento de crear la historia de usuario. Sin embargo, además se pueden añadir nuevos criterios, estos nuevos criterios constituirán una nueva historia de usuario que será priorizada y organizada para un siguiente sprint y así no retrasar el avance del producto. Para generar documentación sobre la presentación de tareas completadas se utilizará la siguiente plantilla.



### Construcción de la base de datos

Para el proceso de la información utilizaremos como mencionado anteriormente las historias de usuario, las cuales para almacenarlas utilizaremos el software Trello, el cual nos permite guardar nuestra información como simples tarjetas de notas.

Alguna de las ventajas que nos ofrece Trello es la integración con otras herramientas muy útiles como ser Google Calendar, Slack, Dropbox y otras herramientas que nos facilitan para llevar el control sobre nuestro proyecto. Esta plataforma nos permite exportar nuestros datos en formato JSON y formato CSV para poder abrirlo en Excel u otras plataformas.



Para presentar los datos utilizaremos el programa Word de Microsoft Office, debido a su popularidad y fácil acceso para las personas.

## Procesamiento de la información

Para el procesamiento de la información utilizaremos como herramienta el grafico Burtdown Chart, el cual nos permitirá estimar el tiempo para la finalización del proyecto y además nos permitirá marcar nuestro avance en el mismo grafico para así poder contrastar nuestro avance con el estimado real, lo cual nos permitirá evaluar y mejorar para las siguientes iteraciones.

Un diagrama burn down o diagrama de quemado es una representación gráfica del trabajo por hacer en un proyecto en el tiempo. Usualmente el trabajo remanente (o backlog) se muestra en el eje vertical y el tiempo en el eje horizontal. Es decir, el diagrama representa una serie temporal del trabajo pendiente. Este diagrama es útil para predecir cuándo se completará todo el trabajo. Usualmente se usa en el desarrollo ágil de software, especialmente con Scrum.

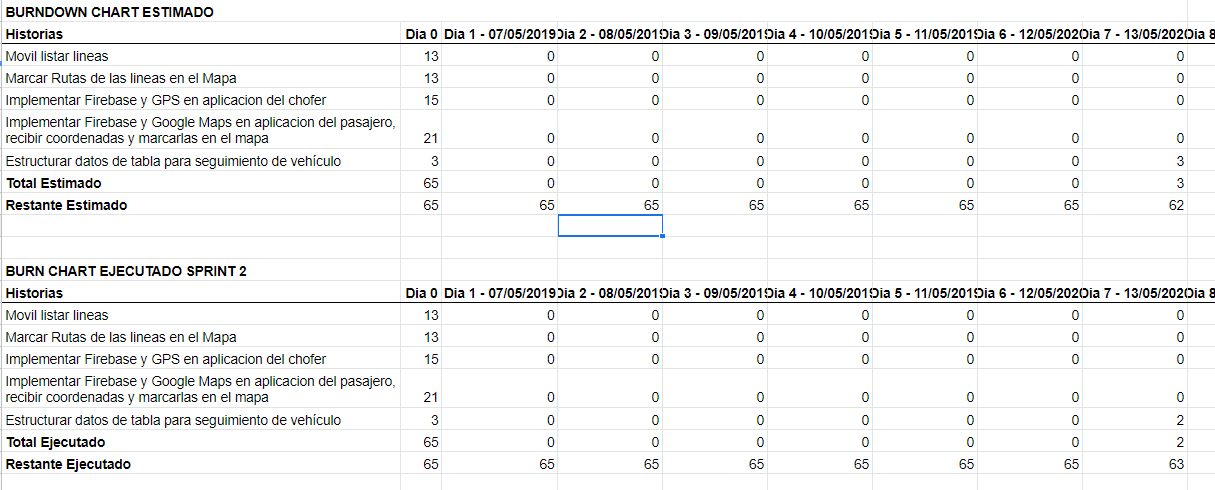
Para la construcción del diagrama de burn down necesitas primero haber valorado nuestras historias de usuario y haberlas organizado en un sprint. Una vez, estas historias estén organizadas y valoradas nos permite obtener un total de puntos que se quiere realizar en un sprint y más específicamente en cada día o semana, dependiendo el intervalo que se quiera utilizar.

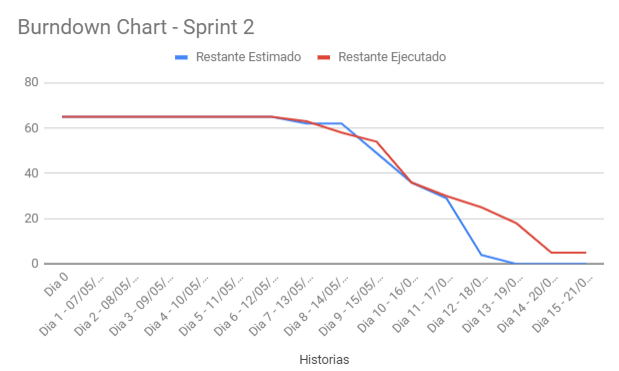
## Objetivo

Realizar una planificación sobre el avance que se espere realizar en cada Sprint y a su vez llevar un control sobre el avance real para así poder contrastar nuestra planificación con el avance real y poder hacer análisis y mejoras.

## Requerimientos

Para poder realizar el diagrama de burn down primero necesitamos tener definida las historias de usuario para el sprint y tener valoradas estas mismas historias de usuario.





En este ejemplo podemos ver la línea azul como el estimado y la línea roja como el ejecutado, podemos ver un claro retraso entre la fecha de finalización ejecutada y la estimada, también podemos observar que no se pudo completar el 100% de las historias de usuario lo cual nos permite ver que existen fallas en la planificación y problemas que están impidiendo el avance del proyecto y que requieren atención.

### Índice tentativo o esquema de contenido

* Capítulo I Marco Introductorio
  + Introducción
  + Problema
  + Planteamiento del problema
  + Problemas específicos
  + Formulación del problema
  + Objetivos
  + Objetivo general
  + Objetivos específicos
  + Delimitación
  + Delimitación espacial
  + Delimitación temporal
  + Delimitación sustantiva
  + Justificación teórica
  + Justificación social
  + Justificación individual
  + Estrategia metodológica
  + Investigación aplicada
  + Factibilidad
* Capitulo II Marco teórico
  + Marco contextual
* Capitulo III Marco aplicativo
  + Delineación de la investigación
    - Objetivo
    - Alcance
  + Planificación
  + Cronograma
* Capítulo VI Conclusiones
  + Glosarios
  + Anexos

### Costos

Los costos del proyecto se verán definidos en tres áreas

* Recursos humanos
* Servicios
* Materiales

Debido a que a diferencia de otras construcciones la del software es muy distinta ya que no se invierte mucho en materia prima, o productos para el desarrollo, sino se invierte principalmente en la mano de obra.

Para este proyecto trabajaremos sobre horas de trabajo, un total de 2800 horas de trabajo que serían el equivalente al trabajo de 3 personas por 6 meses, según podemos apreciar en la siguiente tabla este es el gasto más significativo para el proyecto, luego el alquiler y las maquinas.

|  |  |  |  |
| --- | --- | --- | --- |
| Descripción de gatos | Costo unitario Bs | Cantidad | Costo total Bs |
| Internet de 10 Mbps | 300 | 3 | 1800 |
| Laptop | 3500 | 1 | 10500 |
| Material de oficina | 200 | 6 | 1200 |
| Horas de trabajo | 30 | 540 | 16200 |
| Gastos de entrevistas | 100 | 10 | 1000 |
| Servidor para pruebas | 350 | 3 | 1050 |
| Total (Bs) | | | 37750 |

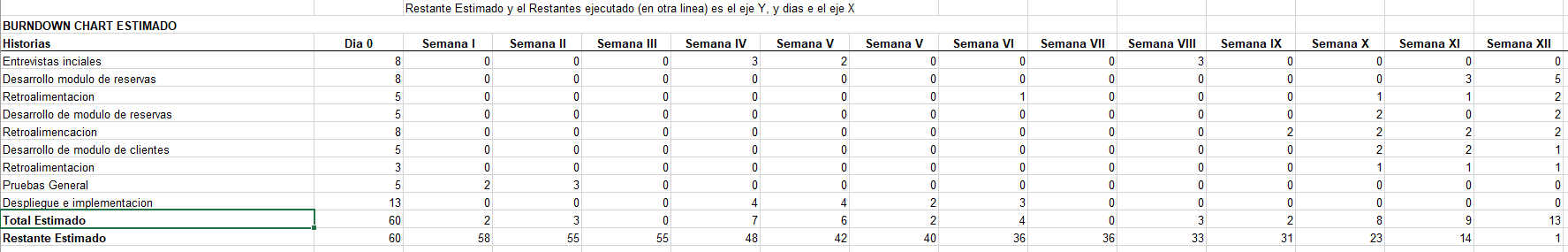
## Cronograma

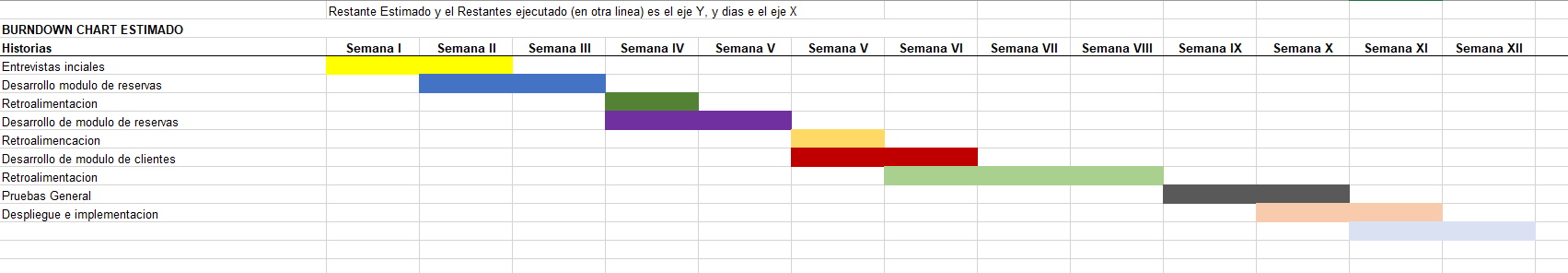
Nuestro cronograma estará basado en nuestras principales actividades las cuales son:

* Entrevistas iniciales
* Desarrollo Modulo
* Retroalimentación y pruebas del desarrollo
* Pruebas generales
* Implementación y despliegue

Para mostrar el cronograma utilizaremos el diagrama de Gantt el cual nos permitirá observar cómo nuestras actividades coinciden para así identificar la ruta critica y planificar de una mejor manera el proyecto, el diagrama de Gantt generalizado es el siguiente, indicando a grandes rasgos las etapas de desarrollo e implementación del sistema. A su vez, el proyecto se dividirá en etapas:

* Entrevistas
* Desarrollar modelo de dominio, modelo de negocio
* Desarrollo
* Pruebas y correcciones
* Despliegue





### Glosario

**1. Software**

Software es un término informático que hace referencia a un programa o conjunto de programas de cómputo, así como datos, procedimientos y pautas que permiten realizar distintas tareas en un sistema informático.

Comúnmente se utiliza este término para referirse de una forma muy genérica a los programas de un dispositivo informático, sin embargo, el software abarca todo aquello que es intangible en un sistema computacional.

Software es un término procedente del idioma inglés, que es aceptado por la RAE y que no posee una traducción que se ajuste al español.

Fecha de actualización: 01/08/2019. Cómo citar: "Software". En: Significados.com. Disponible en: https://www.significados.com/software/ Consultado: 1 de octubre de 2019, 06:28 am.

**2. Inteligencia Artificial**

Disciplina científico-técnica que trata de crear sistemas artificiales capaces de comportamientos que, de ser realizados por seres humanos, se diría que requieren inteligencia

Henao, D. (2009). Inteligencia artificial. Santa Fe, Argentina, Argentina: El Cid Editor | apuntes. Recuperado de https://elibro.net/es/ereader/biblionur/28859?page=5.

**3. HTML**

HTML (Hyper Text Markup Language) es un lenguaje de marcas que permite desarrollar páginas web que sean accesibles a través de la WWW (World Wide Web), también conocida comúnmente como web.

**4. DOM**

EI DOM (Document Object Model) se puede deﬁnir como un conjunto de utilidades diseñadas principalmente para manipular documentos en XML, pudiéndose utilizar también para manipular documentos XHTMLy HTML de una manera rápida y eﬁciente.

Guerrero Pérez, R. (2015). Creación de páginas web con el lenguaje de marcas: confección y publicación de páginas web (UF1302). Antequera, Málaga, Spain: IC Editorial. Recuperado de https://elibro.net/es/ereader/biblionur/44106?page=19.

Guerrero Pérez, R. (2015). Creación de páginas web con el lenguaje de marcas: confección y publicación de páginas web (UF1302). Antequera, Málaga, Spain: IC Editorial. Recuperado de https://elibro.net/es/ereader/biblionur/44106?page=13.

**5. AJAX**

Proviene del inglés Asynchronous JavaScript And XML (JavaScript y XML con ejecución asíncrona). Al fin y al cabo, AJAX es una forma de desarrollo o programación de aplicaciones web que combina: - Una presentación basada en los estándares XHTML y CSS - Utiliza el DOM para interactuar con la página web - Para el intercambio de datos se basa en XML y XSLT - Solicita datos a un servidor de forma asíncrona utilizando XMLHttpRequest - Junta todo y lo presenta al usuario utilizando JavaScript

**6. Internet**

Es una red de cómputo a nivel mundial que agrupa a distintos tipos de

redes usando un mismo protocolo de comunicación. Los usuarios en Internet

pueden compartir datos, recursos y servicios. Internet se apoya en el conjunto de

protocolos TCP/IP. De forma más específica, Internet es la WAN más grande que

hay en el planeta, e incluye decenas de MAN´S y miles de LAN´S. El organismo

que se encarga de regular, establecer estándares, administrar y hacer

operacional a Internet es la ISOC (Internet Society).

Bibliografía Cela Rodríguez, Julia. Internet para periodistas (con especial

incidencia en el ámbito español) En: Cuadernos de documentación

multimedia. Facultad de Ciencias de la Información de la UCM, nº 5.1996.

http://www.ucm.es/info/multidoc/multidoc/revista/cuadern5/julia.htm

[Consultado:17,12,2007]

**7. Base de datos**

Una base de datos (BD) es la representación de una colección de datos estructura- rada que describe las actividades de una organización. ‘Esta representación incluye entidades del mundo real y sus interrelaciones y tiene que permitir diversas utilizaciones.

Rodríguez González, M. E. (2013). Gestión de datos: bases de datos y sistemas gestores de bases de datos. Barcelona, Editorial UOC. Recuperado de https://elibro.net/es/ereader/biblionur/101122?page=9.

**8. MVC**

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.

El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Glosario de términos relacionados con Internet. (2018, 7 diciembre). Recuperado 1 octubre, 2019, de https://disenowebakus.net/glosario-diseno-web.php

**9. Backup**

Copia de Respaldo o Seguridad. Acción de copiar archivos o datos de forma que estén disponibles en caso de que un fallo produzca la perdida de los originales. Esta sencilla acción evita numerosos, y a veces irremediables, problemas si se realiza de forma habitual y periódica.

Glosario de términos relacionados con Internet. (2018, 7 diciembre). Recuperado 1 octubre, 2019, de https://disenowebakus.net/glosario-diseno-web.php

**10. Linux**

Sistema operativo (en realidad el kernel o núcleo del sistema) de software libre, es decir, que se pueden ver y modificar su código fuente (sus tripas). A veces se emplea Linux y software libre como sinónimos.

Glosario de términos relacionados con Internet. (2018, 7 diciembre). Recuperado 1 octubre, 2019, de https://disenowebakus.net/glosario-diseno-web.php

**11. Interface de usuario (UI)**

La interface de usuarios es el conjunto de elementos que ve la persona que visita tu página Web, desde los colores hasta los botones y el contenido. La idea es que la interface sea amigable con el usuario, incluso que sea atractiva y que invite a navegar. De la necesidad de una buena interface es que se genera la importancia del trabajo de los diseñadores Web.

Xplora. (2019, 7 agosto). Glosario de términos de diseño y desarrollo de páginas Web avanzado. Recuperado 1 octubre, 2019, de https://www.xplora.eu/glosario-diseno-web-avanzado/

**12. Experiencia de usuario (UX)**

La experiencia del usuario es un concepto que ha ido ganando relevancia en los últimos años. Ya no se trata de servir la información, sino de que ésta sea lo más agradable para el usuario. Esta experiencia depende de muchas cosas, no solamente del diseño, por ejemplo, depende de la velocidad de carga del sitio y de la calidad de la información.

Xplora. (2019, 7 agosto). Glosario de términos de diseño y desarrollo de páginas Web avanzado. Recuperado 1 octubre, 2019, de https://www.xplora.eu/glosario-diseno-web-avanzado/

**13. Programación**

Programación es la acción de programar que implica ordenar, estructurar o componer una serie de acciones cronológicas para cumplir un objetivo. La programación puede ser aplicado para eventos socia

Fecha de actualización: 07/07/2017. Cómo citar: "Programación". En: Significados.com. Disponible en: https://www.significados.com/programacion/ Consultado: 1 de octubre de 2019, 06:29 am.

**14. Lenguaje de programación**

Un lenguaje de programación es, en la ciencia de la computación, la herramienta para automatizar informaciones y acciones a través de una computadora. Los lenguajes de programación más conocidos son: Basic (1964), C++ (1983), Phyton (1991), Java (1995), C# (2000), entre otros. lenguajes de programación.

Fecha de actualización: 07/07/2017. Cómo citar: "Programación". En: Significados.com. Disponible en: https://www.significados.com/programacion/ Consultado: 1 de octubre de 2019, 06:29 am.

**15. React**

Una biblioteca de JavaScript para construir interfaces de usuario. React te ayuda a crear interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en tu aplicación, y React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien.

Facebook. (s.f.). React – Una biblioteca de JavaScript para construir interfaces de usuario. Recuperado 1 octubre, 2019, de https://es.reactjs.org/

**16. Administración**

Administración es el acto de administrar, planificar, controlar y dirigir los diversos recursos con los que cuenta una persona, empresa, negocio u organización, con el fin de alcanzar una serie de objetivos.

Fecha de actualización: 21/06/2019. Cómo citar: "Administración". En: Significados.com. Disponible en: https://www.significados.com/administracion/ Consultado: 1 de octubre de 2019, 06:36 am.

**17. Contabilidad**

La contabilidad es la parte de las finanzas que estudia las distintas partidas que reflejan los movimientos financieros de una empresa o entidad.

Es una herramienta clave para conocer en qué situación y condiciones se encuentra una empresa y, con esta documentación, poder establecer las estrategias necesarias con el objeto de mejorar su rendimiento económico. Por ejemplo, si compramos madera para fabricar sillas tendremos que contabilizar esa compra para saber qué cantidad tenemos, cuanto nos ha costado, quién es el vendedor, en qué fecha la compramos, etc. De todo eso y más se encarga la contabilidad.

Susana Gil, S. G. (2019, 14 septiembre). Contabilidad - Definición, qué es y concepto | Economipedia. Recuperado 1 octubre, 2019, de https://economipedia.com/definiciones/contabilidad.html

Comisión

Una comisión es un porcentaje sobre el valor de una transacción que se cobra al cliente y/o se paga al Paula Nicole Roldán, P. N. R. (2017, 29 junio). Comisión - Definición, qué es y concepto | Economipedia. Recuperado 1 octubre, 2019, de https://economipedia.com/definiciones/comision.html vendedor.

**18. Marketing**

El marketing, mercadotecnia o mercadeo es el conjunto de actividades destinadas a identificar y satisfacer las necesidades y deseos de los consumidores. También se conoce como mercadeo o mercadología.

Pablo Sevilla Arias, P. S. A. (2019, 25 agosto). Marketing - Definición, qué es y concepto | Economipedia. Recuperado 1 octubre, 2019, de https://economipedia.com/definiciones/mercadotecnia-marketing.html

**19. Hardware**

Hardware es la parte física de un ordenador o sistema informático. Está formado por los componentes eléctricos, electrónicos, electromecánicos y mecánicos, tales como circuitos de cables y luz, placas, memorias, discos duros, dispositivos periféricos y cualquier otro material en estado físico que sea necesario para hacer que el equipo funcione.

Fecha de actualización: 10/07/2019. Cómo citar: "Hardware". En: Significados.com. Disponible en: https://www.significados.com/hardware/ Consultado: 1 de octubre de 2019, 06:46 am.

Patrón de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.

Antonio Leiva, A. L. (2016, 5 marzo). Patrones de diseño de software - DevExperto. Recuperado 1 octubre, 2019, de <https://devexperto.com/patrones-de-diseno-software/>

### Fichas bibliográficas

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Sam Ruby | **Editorial** | Pragmatic Bookshelf |
| **Título** | Agile Web Development with Rails 5 | **Ciudad, país** | Estados Unidos, Raleigh, North Carolina |
| **Año** | 2016 |  | |
| **Tema: Patrón de Diseño MVC**  **p.42**  “The model is responsible for maintaining the state of the application. Sometimes this state is transient, lasting for just a couple of interactions with the user. Sometimes the state is permanent and is stored outside the application, often in a database. A model is more than data; it enforces all the business rules that apply to that data. For example, if a discount shouldn’t be applied to orders of less than $20, the model enforces the constraint. This makes sense; by putting the implementation of these business rules in the model, we make sure that nothing else in the application can make our data invalid. The model acts as both a gatekeeper and a data store. The view is responsible for generating a user interface, normally based on data in the model. For example, an online store has a list of products to be displayed on a catalog screen. This list is accessible via the model, but it’s a view that formats the list for the end user. Although the view might present the user with various ways of inputting data, the view itself never handles incoming data. The view’s work is done once the data is displayed. There may well be many views that access the same model data, often for different purposes. The online report erratum • discuss store has a view that displays product information on a catalog page, and another set of views used by administrators to add and edit products. Controllers orchestrate the application. Controllers receive events from the outside world (normally, user input), interact with the model, and display an appropriate view to the user.” | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Monte Galiano, J | **Editorial** | Ednmnal UOC (Oberta UOC Pubhshmg, SL) |
| **Título** | Implantar scrum con éxito | **Ciudad, país** | Barcelona, España |
| **Año** | 2016 |  | |
| **Tema: Scrum**  **p.21**  Scrum esta basado, por un lado, en la teoría del control empírico de procesos para la gestión de sistemas adaptativos complejos Los tres pilares de este proceso son los siguientes: ' Transparencia: los aspectos signiﬁcativos del proceso tienen que ser conocidos por todo aquel que participa, lo cual conlleva que estos aspectos estén deﬁnidos mediante un estándar común, de forma que todo el mundo tenga la misma percepción de las características de cada aspecto (por ejemplo, la deﬁnición de grabado). ' Inspección: todo proceso persigue un objetivo y, para llei gar a ese objetivo, hace falta que los parúcipantes en el proceso evalúen de manera continua sus resultados, y el proceso mismo, para detectar posibles desviaciones tan pronto como sea posible. ' Adaptación: cuando se detecta una desviación, la respuesi ta debe ser la adaptación; es decir, la adopción de acciones o planes que, o bien ayuden a corregir la desviación, o bien reconﬁguren el objetivo | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Ian Sommerville | **Editorial** | Pearson educacion Mexico |
| **Título** | Ingenieria de Software | **Ciudad, país** | Juarez,Mexico |
| **Año** | 2011 |  | |
| **Tema: Ingenieria de Software**  **p.35**  La ingeniería de software (el término es discutido por cuanto el desarrollo de software no es en muchas ocasiones considerado como una ingeniería) es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software,​ y el estudio de estos enfoques, es decir, el estudio de las aplicaciones de la ingeniería al software. Integra matemáticas, ciencias de la computación y prácticas cuyos orígenes se encuentran en la ingeniería.  La creación del software es un proceso intrínsecamente creativo y la ingeniería del software trata de sistematizar este proceso con el fin de acotar el riesgo de fracaso en la consecución del objetivo, por medio de diversas técnicas que se han demostrado adecuadas sobre la base de la experiencia previa. | | | |
| **Edición** | 9na edición | **Numero de ficha bibliográfica** | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL : <https://www.atlassian.com/git/tutorials/why-git> | |
| **Tema: GIT**  **p.35**  Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning your local copy of code is a complete version control repository. These fully-functional local repositories make it is easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server. This paradigm differs from centralized version control where clients must synchronize code with a server before creating new versions of code.  Git’s flexibility and popularity make it a great choice for any team. Many developers and college graduates already know how to use Git. Git’s user community has created many resources to train developers and Git’s popularity make it easy to get help when you need it. Nearly every development environment has Git support and Git command line tools run on every major operating system.  Unlike some version control software, Git is not fooled by the names of the files when determining what the storage and version history of the file tree should be, instead, Git focuses on the file content itself. After all, source code files are frequently renamed, split, and rearranged. The object format of Git's repository files uses a combination of delta encoding (storing content differences), compression and explicitly stores directory contents and version metadata objects.  Security  Git has been designed with the integrity of managed source code as a top priority. The content of the files as well as the true relationships between files and directories, versions, tags and commits, all of these objects in the Git repository are secured with a cryptographically secure hashing algorithm called SHA1. This protects the code and the change history against both accidental and malicious change and ensures that the history is fully traceable.  With Git, you can be sure you have an authentic content history of your source code.  Some other version control systems have no protections against secret alteration at a later date. This can be a serious information security vulnerability for any organization that relies on software development.  Flexibility  One of Git's key design objectives is flexibility. Git is flexible in several respects: in support for various kinds of nonlinear development workflows, in its efficiency in both small and large projects and in its compatibility with many existing systems and protocols.  Git has been designed to support branching and tagging as first-class citizens (unlike SVN) and operations that affect branches and tags (such as merging or reverting) are also stored as part of the change history. Not all version control systems feature this level of tracking. | | | |
|  | | **Numero de ficha bibliográfica** | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL : <https://es.reactjs.org/> | |
| **Tema: React**  React is a popular library used to create user interfaces. It was built at Facebook to address some of the challenges associated with large-scale, data-driven websites. When React was released in 2013, the project was initially viewed with some skepti‐ cism because the conventions of React are quite unique. In an attempt to not intimidate new users, the core React team wrote an article called “Why React?” that recommended that you “Give It [React] Five Minutes.” They wanted to encourage people to work with React first before thinking that their approach was too crazy. Yes, React is a small library that doesn’t come with everything you might need out of the box to build your application. Give it five minutes. Yes, in React, you write code that looks like HTML right in your JavaScript. And yes, those tags require preprocessing to run in a browser. And you’ll probably need a build tool like webpack for that. Give it five minutes. If you read that article—as we did—you may have been dazzled by the promise of a new JavaScript library—a library that would solve all of our problems with the DOM; a library that would always be easy to work with and would never hurt us. Then the questions start to arise: how do I convert this JSX? How do I load data? Where does the CSS go? What is declarative programming? Every path leads to more questions about how to incorporate this library in your actual day to day work. Every conversation introduces new terminology, new techniques, and more questions. | | | |
|  | | **Numero de ficha bibliográfica** | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://es.redux.js.org/> | |
| **Tema: Redux**  Redux es un contenedor predecible del estado de aplicaciones JavaScript.  Te ayuda a escribir aplicaciones que se comportan de manera consistente, corren en distintos ambientes (cliente, servidor y nativo), y son fáciles de probar. Además de eso, provee una gran experiencia de desarrollo, gracias a edición en vivo combinado con un depurador sobre una línea de tiempo.  Puedes usar Redux combinado con React, o cual cualquier otra librería de vistas. Es muy pequeño (2kB) y no tiene dependencias.  Redux has emerged as one of the clear winners in the field of Flux or Flux-like libra‐ ries. Redux is based on Flux, and it was designed to tackle the challenge of under‐ standing how data changes flow through your application. Redux was developed by Dan Abramov and Andrew Clark. Since creating Redux, both have been hired by Facebook to work on the React team. Andrew Clark was working on version 4 of Flummox, another Flux-based library, when he started assisting Dan with the task of completing Redux. The message on the npm page for Flummox reads: Eventually 4.x should be the last major release but it never happened. If you want the latest features, then use Redux instead. It’s really great.1 Redux is surprisingly small, only 99 lines of code. We have mentioned that Redux is Flux-like, but it is not exactly Flux. It has actions, action creators, a store, and action objects that are used to change state. Redux sim‐ plifies the concepts of Flux a bit by removing the dispatcher, and representing appli‐ cation state with a single immutable object. Redux also introduces reducers, which are not a part of the Flux pattern. Reducers are pure functions that return a new state based on the current state and an action: (state, action) => newState. | | | |
|  | | **Numero de ficha bibliográfica** | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL : <https://facebook.github.io/react-native/docs/tutorial> | |
| **Tema: React native**  React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It’s based on React, Facebook’s JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly “native,” all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS.  Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque markup, known as JSX. Then, under the hood, the React Native “bridge” invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, not webviews, and will look and feel like any other mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user’s location.  React Native currently supports both iOS and Android, and has the potential to expand to future platforms as well. In this book, we’ll cover both iOS and Android. The vast majority of the code we write will be cross-platform. And yes: you can really use React Native to build production-ready mobile applications! Some anecdota: Facebook, Palantir, and TaskRabbit are already using it in production for user-facing applications.  Advantages of React Native  The fact that React Native actually renders using its host platform’s standard rendering APIs enables it to stand out from most existing methods of cross-platform application development, like Cordova or Ionic. Existing methods of writing mobile applications using combinations of JavaScript, HTML, and CSS typically render using webviews. While this approach can work, it also comes with drawbacks, especially around performance. Additionally, they do not usually have access to the host platform’s set of native UI elements. When these frameworks do try to mimic native UI elements, the results usually “feel” just a little off; reverse-engineering all the fine details of things like animations takes an enormous amount of effort, and they can quickly become out of date.  In contrast, React Native actually translates your markup to real, native UI elements, leveraging existing means of rendering views on whatever platform you are working with. Additionally, React works separately from the main UI thread, so your application can maintain high performance without sacrificing capability. The update cycle in React Native is the same as in React: when props or state change, React Native re-renders the views. The major difference between React Native and React in the browser is that React Native does this by leveraging the UI libraries of its host platform, rather than using HTML and CSS markup.  For developers accustomed to working on the Web with React, this means you can write mobile apps with the performance and look and feel of a native application, | | | |
|  | | **Numero de ficha bibliográfica** | 7 |

|  |  |  |
| --- | --- | --- |
| **Autor** | Pryyesh patel |  |
| **Título** | Hybrid vs native |
| **Fecha** | 04/12/2018 | URL :<https://blog.techmagic.co/native-vs-hybrid-apps/> |

|  |  |  |
| --- | --- | --- |
| **Tema: Hybrid vs native**  Native apps  Native apps are written in a programming language specific to the platform they’re being developed for. This would typically be Objective-C or Swift for iOS and Java for Android. Native apps typically have better performance with rendering and animations than hybrid apps.  Hybrid apps  A hybrid app is a mobile app that contains a web view (essentially an isolated browser instance) to run a web application inside of a native app, using a native app wrapper that can communicate with the native device platform and the web view. This means web applications can run on a mobile device and have access to the device, such as the camera or GPS features.  Hybrid apps are possible because of tools that have been created that facilitate the communication between the web view and the native platform. These tools are not part of the official iOS or Android platforms, but are third party tools such as Apache Cordova, which is used in this book. When a hybrid app is built, it will be compiled, transforming your web application into a native app.  There are so many hybrid mobile frameworks such as Ionic, NativeScript, React Native, Xamarin, PhoneGap etc. We decided to compare native apps with top 2 hybrid frameworks — Ionic and React Native.  Ionic  Ionic is a framework that essentially allows developers to create hybrid mobile apps using web technologies like HTML, CSS and JavaScript.  A hybrid mobile app is built using technologies typically used for the web. Hybrid apps are hosted inside native applications that allow them to access the device’s camera, pedometer and other functionalities,removing the need to develop for any specific device or operating system.  This basically means that you are creating a website wrapped up inside an app.  React Native  React Native is a framework developed by Facebook for creating native-style apps for iOS & Android under one common language, JavaScript.  Unlike hybrid apps, native apps are built especially for the platform they’re to be used on (iOS,Android etc). React Native allows a proportion of the code to be shared between platforms and empowers developers to create apps which feel less clunky and perform better than hybrid apps.  With both React Native and Native apps, gestures like pinching or double tapping work in the way you’d expect them to work on your operating system.  We made table with comparing each technology and | | |
|  | **Numero de ficha bibliográfica** | 9 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://www.xplenty.com/blog/the-sql-vs-nosql-difference/> | |
| **Tema: SQL vs No SQL**  The Big Picture Differences Between SQL and NoSQL  The Language  Think of a town - we’ll call it Town A - where everyone speaks the same language. All of the businesses are built around it, every form of communication uses it. In short, it’s the only way that the residents understand and interact with the world around them. Changing that language in one place would be confusing and disruptive for everyone.  Now, think of another town, Town B, where every home can speak a different language. Everyone interacts with the world differently, and there’s no “universal” understanding or set organization. If one home is different, it doesn’t affect anyone else at all.  This helps illustrate one of the fundamental differences between SQL (relational) and NoSQL (non-relational) databases, and this distinction has big implications. Let’s explain:  SQL databases: SQL databases use structured query language (SQL) for defining and manipulating data. On one hand, this is extremely powerful: SQL is one of the most versatile and widely-used options available, making it a safe choice and especially great for complex queries. On the other hand, it can be restrictive. SQL requires that you use predefined schemas to determine the structure of your data before you work with it. In addition, all of your data must follow the same structure. This can require significant up-front preparation, and, as with Town A, it can mean that a change in the structure would be both difficult and disruptive to your whole system.  NoSQL databases: NoSQL databases, on the other hand, have dynamic schemas for unstructured data, and data is stored in many ways: They can be column-oriented, document-oriented, graph-based or organized as a KeyValue store. This flexibility means that:  The Scalability  In most situations, SQL databases are vertically scalable, which means that you can increase the load on a single server by increasing things like CPU, RAM or SSD. NoSQL databases, on the other hand, are horizontally scalable. This means that you handle more traffic by sharding, or adding more servers in your NoSQL database. It’s like adding more floors to the same building versus adding more buildings to the neighborhood. The latter can ultimately become larger and more powerful, making NoSQL databases the preferred choice for large or ever-changing data sets.  The Structure  SQL databases are table-based, while NoSQL databases are either document-based, key-value pairs, graph databases or wide-column stores. This makes relational SQL databases a better option for applications that require multi-row transactions - such as an accounting system - or for legacy systems that were built for a relational structure.  Some examples of SQL databases include MySQL, Oracle, PostgreSQL, and Microsoft SQL Server. NoSQL database examples include MongoDB, BigTable, Redis, RavenDB Cassandra, HBase, Neo4j and CouchDB. | | | |
|  | | **Numero de ficha bibliográfica** | 10 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Kayla Ngan |  | |
| **Título** | Ingenieria de Software |
| **Fecha** | 05/03/2018 | URL :<https://www.postgresql.org/about/> | |
| **Tema: Postgres**  PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 30 years of active development on the core platform.  PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organisations.  Getting started with using PostgreSQL has never been easier - pick a project you want to build, and let PostgreSQL safely and robustly store your data.  PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible. For example, you can define your own data types, build out custom functions, even write code from different programming languages without recompiling your database!  PostgreSQL tries to conform with the SQL standard where such conformance does not contradict traditional features or could lead to poor architectural decisions. Many of the features required by the SQL standard are supported, though sometimes with slightly differing syntax or function. Further moves towards conformance can be expected over time. As of the version 12 release in October 2019, PostgreSQL conforms to at least 160 of the 179 mandatory features for SQL:2016 Core conformance. As of this writing, no relational database meets full conformance with this standard. | | | |
|  | | **Numero de ficha bibliográfica** | 11 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Monte Galiano, J | **Editorial** | Ednmnal UOC (Oberta UOC Pubhshmg, SL) |
| **Título** | El Proceso  Unificado de  Desarrollo de  Software | **Ciudad, país** | Barcelona, España |
| **Año** | 2016 |  | |
| **Tema: Proceso unificado**  **p.21**  El Proceso Unificado es un proceso de desarrollo de software: “conjunto de  actividades necesarias para transformar los requisitos del usuario en un  sistema software”.  · RUP es un marco genérico que puede especializarse para una variedad de  tipos de sistemas, diferentes áreas de aplicación, tipos de organizaciones,  niveles de aptitud y diferentes tamaños de proyectos.  · RUP está basado en componentes. El sw esta formado por componentes  software interconectados a través de interfaces.  · RUP está dirigido por casos de uso, centrado en la arquitectura, y es  iterativo e incremental.  Dirigido por Casos de Uso  · Un caso de uso es un fragmento de funcionalidad del sistema que proporciona  un resultado de valor a un usuario. Los casos de uso modelan los  requerimientos funcionales del sistema.  · Todos los casos de uso juntos constituyen el modelo de casos de uso.  · Los casos de uso también guían el proceso de desarrollo (diseño,  implementación, y prueba). Basándose en los casos de uso los desarrolladores  crean una serie de modelos de diseño e implementación que llevan a cabo los  casos de uso. De este modo los casos de uso no solo inician el proceso de  desarrollo sino que le proporcionan un hilo conductor, avanza a través de una  serie de flujos de trabajo que parten de los casos de uso.  Centrado en la Arquitectura  La arquitectura de un sistema software se describe mediante diferentes vistas del  sistema en construcción.  El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más  significativos del sistema.  La arquitectura es una vista del diseño completo con las características más  importantes resaltadas, dejando los detalles de lado. | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 12 |

|  |  |  |
| --- | --- | --- |
| **Autor** | Juan Sebastián Villanueva |  |

|  |  |  |
| --- | --- | --- |
| **Título** | SCRUM Y RUP: COMPARATIVA Y PROPUESTA METODOLÓGICA |  |
| **Fecha** | 05/03/2018 | URL :<https://flutter-es.io/> |

|  |  |  |
| --- | --- | --- |
| **Tema: Scrum vs RUP**  Ambas metodologías tienen sus limitaciones y debilidades, así como las metodologías ágiles son las más adecuadas para proyectos pequeños y medianos, no son las más adecuadas para sistemas de gran escala que requieran de interacciones complejas con otros sistemas, debido a que estos sistemas requieren de un nivel de precisión bastante alto y tienen un gran riesgo de construcción. No sería conveniente implementar una metodología ágil para el desarrollo de un sistema crítico en el cual es necesario el análisis detallado de todos los requerimientos para comprender su complejidad e implicaciones, debido a la complejidad y la extrema precisión quepueda tener la captura de requerimientos, en los cuáles las metodologías agiles como SCRUM ofrecen demasiada flexibilidad.Karlstrm y Runeson [9] encontraron que los métodos agiles proveen herramientas ponderosas para la planeación a pequeña escala, control del trabajo diario, reporte de progreso y la mejora en los canales de comunicación del equipo.Los procesos de desarrollo agiles son una gran opción cuando el objetivo es el incremento de la productividad [10], [11] ya que se enfoca en la importancia del manejo del equipo y de personas, o el mejoramiento de la capacidad de respuesta a peticiones de cambio hechas a través del ciclo de desarrollo de software [12]. | | |
|  | **Numero de ficha bibliográfica** | 13 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Sam Ruby | **Editorial** | Pragmatic Bookshelf |

|  |  |  |  |
| --- | --- | --- | --- |
| **Título** | Agile Web Development with Rails 5 | **Ciudad, país** | Estados Unidos, Raleigh, North Carolina |
| **Año** | 2016 |  | |
| **Tema: Rails**  **p.42**  Ruby on Rails is a framework that makes it easier to develop, deploy, and maintain web applications. During the ten years since its initial release, Rails went from being an unknown toy to being a worldwide phenomenon; more important, it has become the framework of choice for the implementation of a wide range of applications. Why is that? Rails Simply Feels Right A large number of developers were frustrated with the technologies they were using to create web applications. It didn’t seem to matter whether they used Java, PHP, or .NET—there was a growing sense that their jobs were just too damn hard. And then, suddenly, along came Rails, and Rails was easier. But easy on its own doesn’t cut it. We’re talking about professional developers writing real-world websites. They wanted to feel that the applications they were developing would stand the test of time—that they were designed and implemented using modern, professional techniques. So, these developers dug into Rails and discovered it wasn’t just a tool for hacking out sites. For example, all Rails applications are implemented using the Model-ViewController (MVC) architecture. Java developers are used to frameworks such as Tapestry and Struts, which are based on MVC. But Rails takes MVC further: when you develop in Rails, you start with a working application, there’s a place for each piece of code, and all the pieces of your application interact in a standard way. Professional programmers write tests. And again, Rails delivers. All Rails applications have testing support baked right in. As you add functionality to the code, Rails automatically creates test stubs for that functionality. The framework makes it easy to test applications, and as a result, Rails applications tend to get tested. | | | |
| **Edición** | Primera edición | **Numero de ficha bibliográfica** | 17 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | Juan Sebastián Villanueva |  | |
| **Título** | SCRUM Y RUP: COMPARATIVA Y PROPUESTA METODOLÓGICA |
| **Fecha** | 05/03/2018 | URL :<https://flutter-es.io/> | |
| **Tema: Scrum vs RUP**  Ambas metodologías tienen sus limitaciones y debilidades, así como las metodologías ágiles son las más adecuadas para proyectos pequeños y medianos, no son las más adecuadas para sistemas de gran escala que requieran de interacciones complejas con otros sistemas, debido a que estos sistemas requieren de un nivel de precisión bastante alto y tienen un gran riesgo de construcción. No sería conveniente implementar una metodología ágil para el desarrollo de un sistema crítico en el cual es necesario el análisis detallado de todos los requerimientos para comprender su complejidad e implicaciones, debido a la complejidad y la extrema precisión que pueda tener la captura de requerimientos, en los cuáles las metodologías agiles como SCRUM ofrecen demasiada flexibilidad.Karlstrm y Runeson [9] encontraron que los métodos agiles proveen herramientas ponderosas para la planeación a pequeña escala, control del trabajo diario, reporte de progreso y la mejora en los canales de comunicación del equipo.Los procesos de desarrollo agiles son una gran opción cuando el objetivo es el incremento de la productividad [10], [11] ya que se enfoca en la importancia del manejo del equipo y de personas, o el mejoramiento de la capacidad de respuesta a peticiones de cambio hechas a través del ciclo de desarrollo de software [12]. | | | |
|  | | **Numero de ficha bibliográfica** | 13 |