UNIVERSIDAD NACIONAL
"PEDRO RUIZ GALLO"

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA PROFESIONAL DE INGENIERÍA EN COMPUTACIÓN E INFORMÁTICA

TESIS

"Desarrollo del generador de código GENCODE para facilitar la codificación de interfaces web con acceso a datos"

PRESENTADO PARA OPTAR EL TÍTULO PROFESIONAL DE:

Ingeniero(a) en Computación e Informática

Investigadores:

Bach. Facundo Barboza, Fiorella Mayté

Bach. Ramos Benites, Richard Andree

Asesor:

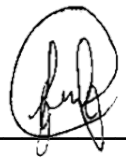
Ing. Terán Santa Cruz, Franklin Edinson

Lambayeque – Perú

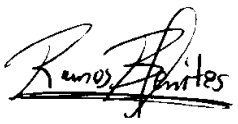
2022

“Desarrollo del generador de código GENCODE para facilitar la codificación de interfaces web con acceso a datos”

Tesis para optar el Título Profesional de
Ingeniero(a) en Computación e Informática, que presentan:



Bach. Fiorella Mayté Facundo Barboza
Autora



Bach. Richard Andree Ramos Benites
Autor

Asesorado por:

Ing. Franklin Edinson Terán Santa Cruz
Asesor

“Desarrollo del generador de código GENCODE para facilitar la codificación de interfaces web con acceso a datos”

Tesis para optar el Título Profesional de Ingeniero(a) en Computación e Informática

Aprobado por:

Dr. Ing. Gilberto Carrión Barco
Presidente

M. Sc. Ing. Carlos Alberto Valdivia Salazar
Secretario

M. Sc. Ing. Roger Ernesto Alarcón García
Vocal

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO
FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
DECANATO
Ciudad Universitaria - Lambayeque

ACTA DE SUSTENTACIÓN VIRTUAL Nº 011-2022-D/FACFyM

Siendo las 10 am del día 16 de marzo del 2022, se reunieron vía plataforma virtual, https://meet.google.com/fjh-gvcz-qia los miembros del jurado evaluador de la Tesis titulada: **"Desarrollo del generador de código GENCODE para facilitar la codificación de interfaces web con acceso a datos"**

Designados por Resolución N° N° 053-2020-D/FACFyM de fecha 14 de enero de 2020

Con la finalidad de evaluar y calificar la sustentación de la tesis antes mencionada, conformada por los siguientes docentes:

Dr. Ing. Gilberto Carrión Barco	Presidente
M.Sc. Ing. Carlos Alberto Valdivia Salazar	Secretario
M.Sc. Ing. Roger Ernesto Alarcón García	Vocal

La tesis fue asesorada por el Ing. Franklin Edinson Terán Santa Cruz, nombrado por **Resolución N° 1468-2019-D/FACFyM** de fecha 15 de noviembre de 2019.

El Acto de Sustentación fue autorizado por **Resolución N° 229-2022-VIRTUAL-D/FACFyM** de fecha 7 de marzo de 2022.

La Tesis fue presentada y sustentada por los Bachilleres: **Facundo Barboza Fiorella Mayté y Ramos Benites Richard Andree**, y tuvo una duración de 60 minutos.

Después de la sustentación, y absueltas las preguntas y observaciones de los miembros del jurado se procedió a la calificación respectiva, otorgándole el Calificativo de **18 (Dieciocho)** en la escala vigesimal, mención **Muy Bueno**.

Por lo que quedan **APTOS** para obtener el Título Profesional de **Ingeniero(a) en Computación e Informática**, de acuerdo con la Ley Universitaria 30220 y la normatividad vigente de la Facultad de Ciencias Físicas y Matemáticas y la Universidad Nacional Pedro Ruiz Gallo.

Siendo las 11 am se dio por concluido el presente acto académico, dándose conformidad al presente acto con la firma de los miembros del jurado.

Dr. Ing. Gilberto Carrión Barco
Presidente

M.Sc. Ing. Carlos Alberto Valdivia Salazar
Secretario

M.Sc. Ing. Roger Ernesto Alarcón García
Vocal

Ing. Franklin Edinson Terán Santa Cruz
Asesor

CERTIFICO: Que, es copia fiel del original
Fecha: 23 / 03 / 2022
M.Sc. Marco Antonio Martín Peralta Lui
SECRETARIO DOCENTE - FACFyM
VÁLIDO PARA TRÁMITES INTERNOS DE LA UNPRG

Informe final

INFORME DE ORIGINALIDAD

16%	14%	1%	8%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

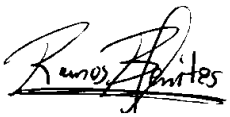
1	repositorio.unprg.edu.pe:8080 Fuente de Internet	3%
2	Submitted to Universidad Cesar Vallejo Trabajo del estudiante	2%
3	Submitted to Universidad Nacional de San Cristóbal de Huamanga Trabajo del estudiante	1%
4	docplayer.es Fuente de Internet	1%
5	repositorio.ug.edu.ec Fuente de Internet	1%
6	Submitted to Universidad Nacional Pedro Ruiz Gallo Trabajo del estudiante	1%
7	hdl.handle.net Fuente de Internet	<1%
8	www.slideshare.net Fuente de Internet	<1%

DECLARACIÓN JURADA DE ORIGINALIDAD

Nosotros, **Fiorella Mayté Facundo Barboza** y **Richard Andree Ramos Benites**, Investigadores Principales y **Ing. Franklin Edinson Terán Santa Cruz**, Asesor del Trabajo de Investigación **"Desarrollo del generador de código GENCODE para facilitar la codificación de interfaces web con acceso a datos"**, declaramos bajo juramento que este trabajo no ha sido plagiado, ni contiene datos falsos. En caso se demostrara lo contrario, asumimos responsablemente la anulación de este informe y por ende el proceso administrativo a que hubiera lugar. Que puede conducir a la anulación del título o grado emitido como consecuencia de este informe.

Lambayeque, enero de 2022.

Bach. Fiorella Mayté Facundo Barboza
Autora



Bach. Richard Andree Ramos Benites
Autor

Ing. Franklin Edinson Terán Santa Cruz
Asesor

DEDICATORIA

A mis padres, quienes con su amor y apoyo incondicional me han permitido llegar a cumplir una de mis grandes metas.

Fiorella Mayté Facundo Barboza

A mis padres y hermana, que son la principal razón de mi vida, por haberme apoyado en todo momento, sin ellos no lo hubiese logrado.

Richard Andree Ramos Benites

AGRADECIMIENTO

A Dios, por darnos la vida y permitirnos estar presentes en vida para lograr todos nuestros objetivos, como es la tesis.

A nuestros padres, por apoyarnos en todo momento, por darnos ese apoyo incondicional que necesitamos para poder cumplir nuestras metas, por inculcarnos de valores y principios para saber llevar de la mejor manera todo lo que se nos presenta en la vida.

A los docentes de la Escuela Profesional de Ingeniería en Computación e Informática, por habernos dado ese apoyo, brindarnos los conocimientos necesarios para nuestro desarrollo como profesionales. En especial a nuestro asesor, el ingeniero Franklin Edinson Terán Santa Cruz, por todo el apoyo brindado, no solo para el desarrollo de la tesis, sino durante toda la carrera universitaria, siempre nos ha guiado tanto personal como académicamente.

Los Autores.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Este trabajo de investigación surge al observar los problemas al momento de desarrollar software. Existen procesos de codificación muy repetitivos, teniendo que dedicar tiempo innecesario, la falta de uso de buenas prácticas ocasionando código confuso y de difícil mantenimiento, así como también el retraso de entrega del mismo a causa del tiempo empleado por los desarrolladores en la codificación.

Debido a lo anterior, nace la idea de desarrollar un generador de código de interfaces CRUD basado en React.js, teniendo como finalidad disminuir el tiempo de codificación, además de permitir a los desarrolladores centrarse en requerimientos que conlleven esfuerzo lógico para construir software de calidad en menor tiempo.

En esta investigación, se aplicó la metodología RUP, se capturó los requerimientos para la creación del generador, se estandarizó el código, se realizó un diseño de interfaz de fácil uso y finalmente se probó la herramienta obteniendo resultados satisfactorios.

Palabras claves: desarrollo de software, herramienta CASE, generador de código.

ABSTRACT

This research work arises from observing the problems when developing software. There are very repetitive coding processes, having to spend unnecessary time, the lack of use of good practices causing confusing code and difficult to maintain, as well as the delay in its delivery due to the time spent by developers in coding.

Due to the above, comes from the idea of developing a CRUD interface code generator based on React.js, with the aim of reducing coding time, in addition to allowing developers to focus on requirements that involve a logical effort to build quality software. in less time.

In this research, the RUP methodology was applied, the requirements for creating the generator were captured, the code was standardized, an easy-to-use interface design was made, and finally the tool was tested, obtaining satisfactory results.

Keywords: software development, CASE tool, code generator.

INTRODUCCIÓN

Hoy en día los negocios, necesitan que sus datos sean controlados con efectividad para así llevar sus procesos más rápidos y a la vez tener a sus clientes satisfechos según el servicio que las empresas ofrecen.

Los sistemas de información llegan para solucionar problemas como, lentitud en el desarrollo de procesos del negocio, inefectividad en la atención, molestias en sus usuarios finales, además de haber invertido costos muy elevados sin ningún efecto. En consecuencia, los negocios cada vez demandan más software especializado y las empresas desarrolladoras están sometidas a entregar sistemas informáticos de calidad en corto tiempo.

Por ello, las empresas desarrolladoras requieren herramientas capaces de agilizar los procesos, pues el mercado actual, es cada vez más exigente debido a la gran cantidad de competencia existente.

El siguiente proyecto se trata de un generador de código de interfaces CRUD basado en React.js teniendo como finalidad disminuir el tiempo de codificación, entregando así software en corto tiempo, cumpliendo con las exigencias del mercado.

Así mismo, esta herramienta disminuirá errores debido a que se tendrá en cuenta las buenas prácticas, haciendo uso de normas de codificación, obteniendo así, una escritura más legible y flexible.

Capítulo I. Diseño Teórico

1.1. Problemática de la investigación

1.1.1. Planteamiento del problema

Hoy en día, en el ámbito empresarial, los sistemas de información son de vital importancia dado que otorgan soluciones efectivas en la toma de decisiones del negocio ayudando a mejorar la productividad, reduciendo costos y agilizando procesos. En consecuencia, el mercado exige cada vez más software especializado por lo cual las empresas desarrolladoras están sometidas a entregar sistemas informáticos de calidad en corto tiempo.

La ingeniería de software es una disciplina que brinda soluciones a varios problemas existentes cuando se implementa software proporcionando métodos y técnicas para su desarrollo. A la vez, ayuda a mejorar la calidad de productos informáticos, aumentar la productividad del sistema y facilitar el control en el proceso de desarrollo.

En la construcción de software surgen problemas significativos; por lo cual se realizó una encuesta a 24 desarrolladores donde se pudo identificar lo siguiente:

El primer problema encontrado es acerca de los tiempos largos en la entrega de funcionalidad CRUD (Crear, Leer, Modificar y Eliminar). Como se observa en la figura 1 el 66.7% de los desarrolladores emplea un tiempo de 150 minutos a más en desarrollar las funcionalidades básicas de un sistema, además, el 91.6% considera que el tiempo es un factor fundamental en el desarrollo del software.

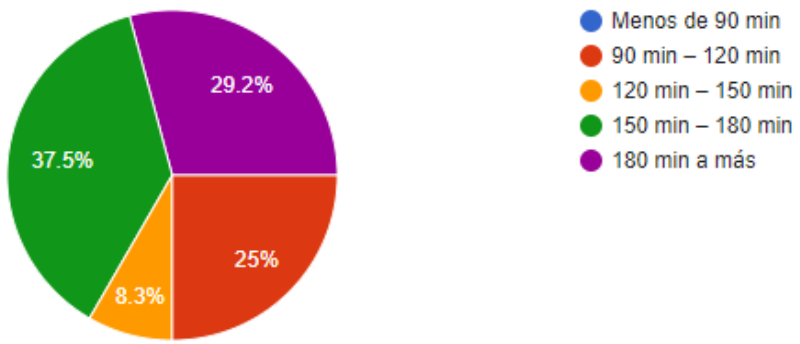


Figura 1. Resultado estadístico del tiempo que se emplea en desarrollar las funcionalidades básicas de un sistema

Fuente. Elaboración propia

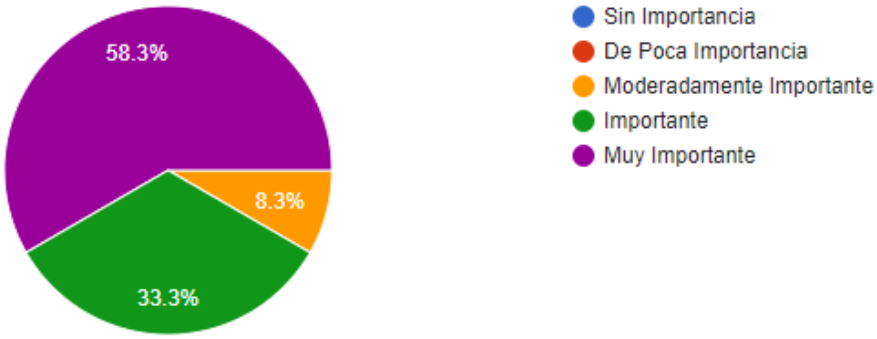


Figura 2. Resultado estadístico del grado de importancia del tiempo en el desarrollo del software

Fuente. Elaboración propia

Otro de los problemas obtenidos, es la frecuencia de reutilización de código por parte de los desarrolladores, debido a la existencia de funcionalidades repetitivas como CRUD. El 79.2% copia y adapta piezas de código existente, dicha práctica ocasiona software propenso a errores. Además; esto genera que los desarrolladores empleen parte de su tiempo codificando procedimientos donde no se requiere esfuerzo lógico ni capacidades analíticas.

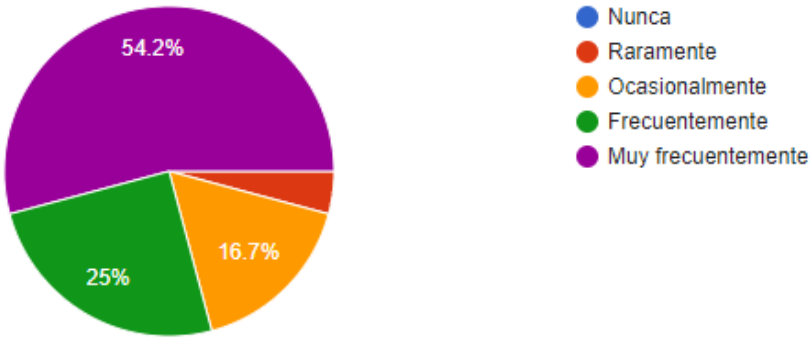


Figura 3. Resultado estadístico de la frecuencia que los desarrolladores de software realizan adaptaciones de código existente

Fuente. Elaboración propia

Finalmente, otro de los factores que afectan al desarrollo de software es la falta de estandarización, como se observa en la figura 4, al no hacer uso de buenas prácticas traerá como consecuencia un software difícil de mantener.

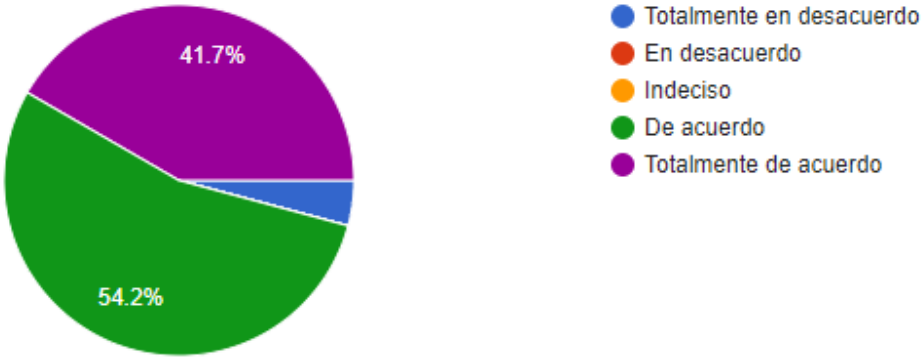


Figura 4. Resultado estadístico acerca de la estandarización en el mantenimiento de código

Fuente. Elaboración propia

De acuerdo a lo mencionado anteriormente, el 83.3% considera que sea posible generar código automáticamente para el desarrollo eficaz y eficiente de un sistema y el 70.8% estaría dispuesto a usar una herramienta CASE que genere las funcionalidades básicas, por ello, este proyecto propone el desarrollo de un generador de código de interfaces web basadas en React.js para facilitar el proceso de

codificación correspondiente a las funcionalidades básicas (CRUD), así como favorecer la estandarización del código fuente generado.

1.2. Formulación del problema de investigación

¿El desarrollo del generador de código GENCODE facilitará el proceso de codificación de interfaces web con React.js para acceso a datos?

1.3. Objetivos

1.3.1. Objetivo general

Implementar el generador de código de interfaces web denominado GENCODE basadas en React.js.

1.3.2. Objetivos específicos

- Analizar los estilos de codificación de interfaces en que utilizan React.js.
- Estandarizar la codificación del diseño de interfaces web usando React.js.
- Diseñar la herramienta CASE GENCODE según la metodología RUP.
- Codificar la herramienta CASE para la generación de interfaces web.
- Probar la herramienta CASE para la verificación de su correcta funcionalidad.

1.4. Marco metodológico

1.4.1. Tipo de investigación

Tipo pre experimental

1.4.2. Hipótesis

El desarrollo del generador de código GENCODE facilitará el proceso de codificación de interfaces web con acceso a datos acelerando el tiempo de desarrollo.

1.4.3. Definición y operacionalización de variables

Tabla 1. Definición y operacionalización de variables

<table>
<tr><td colspan="6">Título: “Desarrollo del generador de código GENCODE para facilitar la codificación de interfaces web con acceso a datos.”</td></tr>
<tr><td rowspan="2">Problema</td><td rowspan="2">Objetivos</td><td rowspan="2">Hipótesis</td><td colspan="3">Operacionalización</td></tr>
<tr><td>Variables</td><td>Definición</td><td>Dimensión</td><td>Indicadores</td></tr>
<tr><td rowspan="5">¿El desarrollo del generador de código GENCODE facilitará el proceso de codificación de interfaces web con React.js para acceso a datos?</td><td rowspan="5">Objetivo General:
• Implementar el generador de código de interfaces web denominado GENCODE basadas en React.js.

Objetivos Específicos:
• Analizar los estilos de codificación de interfaces en que utilizan React.js.
• Estandarizar la codificación del diseño de interfaces web usando React.js.
• Diseñar la herramienta CASE GENCODE según la metodología RUP.
• Codificar la herramienta CASE para la generación de interfaces web.
• Probar la herramienta CASE para la verificación de su correcta funcionalidad.</td><td rowspan="5">El desarrollo del generador de código GENCODE facilitará el proceso de codificación de interfaces web con acceso a datos acelerando el tiempo de desarrollo.</td><td rowspan="2">(Variable Independiente)
Generador de código de interfaces web denominado GENCODE</td><td rowspan="2">Herramienta para crear interfaces web haciendo uso de React.js.</td><td>Requerimientos</td><td>Cumplimento con los requerimientos del usuario.</td></tr>
<tr><td>Estandarización</td><td>Cumplimiento de reglas y estructura de codificación.</td></tr>
<tr><td rowspan="3">(Variable Dependiente)
Proceso de codificación de interfaces web</td><td rowspan="3">Conjunto de actividades orientadas a la creación de funcionalidades web.</td><td rowspan="3">Rapidez</td><td>Tiempo mínimo codificando interfaces web.</td></tr>
<tr><td>Tiempo máximo codificando interfaces web.</td></tr>
<tr><td>Tiempo promedio codificando interfaces web.</td></tr>
</table>

Fuente. Elaboración propia

1.5. Marco teórico

1.5.1. Antecedentes

1.5.1.1. Internacionales

Sayago Heredia (2018). ***Generador de Código Utilizando el Paradigma de Líneas de Producto Software.*** **Pontificia Universidad Católica del Ecuador, Esmeraldas, Ecuador**. El objetivo de este trabajo fue demostrar las ventajas como la potencialidad que tienen los softwares basados en Java EE. El generador pretendió automatizar el proceso de codificación al momento de desarrollar una aplicación web con acceso a base de datos. Se concluyó que el generador de código cumplió con los requerimientos que se llegaron a formular, además automatizar aplicaciones basadas en Java EE. Cuando se llegó a implementar el generador se logró representar un aumento en el desempeño de desarrollo y una disminución del tiempo de codificación de este tipo de aplicación.

Martínez Unaucho y Rodríguez Chalá (2014). ***Desarrollo de un prototipo de un generador de código para aplicaciones JEE6 para la empresa Clear Minds Consultores CIA LTDA*** **(tesis de pregrado). Escuela Politécnica Nacional, Quito, Ecuador.** En este proyecto de tesis se realiza el desarrollo de un generador de código de aplicaciones web y está enfocada al módulo de gestión de un sistema base.

En el resultado obtenido se concluyó que el generador cumple de manera eficiente y satisfactoria con los requerimientos que se planteó al inicio y está en la capacidad de aplicarlo en proyectos empresariales. A las conclusiones que se llagaron en este proyecto fueron: La intervención constante por parte del cliente, esto gracias a SCRUM, todos los requerimientos especificados se cumplieron de

manera satisfactoria, el desarrollo de estos tipos de aplicaciones se volvió un proceso más eficiente.

Izquierdo Herrera, Quiñonez Ku y Casierra Cavada (2018). ***Generador automático de aplicaciones web e interfaces de usuarios con funcionalidad responsiva en el lenguaje python.*** **Pontificia Universidad Católica del Ecuador, Esmeraldas, Ecuador**. En esta investigación el objetivo fue evitar la codificación repetitiva produciendo pérdidas de tiempo y retrasos en los periodos de entrega. Se desarrolló un generador basado en buenas prácticas al momento de construir una aplicación web e implementar las operaciones CRUD mediante una interfaz amigable con el usuario. Con el generador de código se logra obtener aplicaciones web con funcionalidades en base a buenas prácticas, con una interfaz amigable que permite crear código fuente de manera rápida.

1.5.1.2. Nacionales

Mamani Rodrigo (2015). ***Sistema Generador de Aplicaciones Web a partir de Modelos Físicos de Datos*** **(tesis de maestría). Universidad Andina "Néstor Cáceres Velásquez", Juliaca, Perú.** El objetivo de eesta tesis es desarrollar un sistema generador de aplicaciones web a partir de modelos físicos de datos. Se concluye que desarrollar un generador de aplicaciones web es importante debido a la existencia de muchos procesos repetitivos y mecánicos que toman tiempo en la construcción de software, el cuál puede ser ahorrado si es que se establece una manera de automatizar estos procesos. Además, utilizar la metodología de desarrollo iterativo incremental, permite centrarse más en el producto final. Como también a partir de un modelo físico de datos se podrá capturar las necesidades del cliente y así elaborar un buen diseño del sistema propuesto.

Becerra Urbina (2019). ***Generador de Código de Funcionalidades Tipo CRUD en la Mantenibilidad de Software Aplicado a Sistemas de Información Empresariales*** **(tesis de pregrado). Universidad Privada del Norte, Trujillo, Perú.** Este trabajo de investigación consiste en el desarrollo de un generador de código CRUD teniendo como objetivo mejorar el desarrollo de sistemas empresariales, dividiendo componentes, manteniendo una estructura de convenciones y métricas en el código, dando por resultado la facilidad de realizar mantenimientos en el software. Se concluye que el generador de código CRUD influye positivamente en el desarrollo de software empresarial ya que respeta principios de codificación para facilitar su mantenimiento.

1.6. Bases teóricas

1.6.1. Conceptos Generales

1.6.1.1. Ingeniería del software

La Ingeniería del Software es la aplicación de herramientas, métodos y disciplinas para la producción y mantenimiento de soluciones automáticas a problemas del mundo real (Alonso Amo, Martínez Normand y Segovia Pérez, 2005).

Objetivos

- Ofrece estándares, modelos facilitando la comunicación en el equipo del proyecto.
- Aporta métodos, herramientas, procedimientos para la construcción del software.
- Proporciona criterios de evaluación.

1.6.1.2. Proceso de desarrollo de software

También llamada “ciclo de vida” o “proceso de software”. Es un conjunto de actividades utilizados por los ingenieros de software para el desarrollo de un producto. Existen diversos modelos para llevar a cabo estos procesos, cada uno de ellos describe sus propios enfoques para las diversas tareas a ejecutar (Noriega Martínez, 2017).

1.6.1.3. Desarrollo web

Actualmente, el desarrollo de software se enfoca en la creación de aplicaciones web, poco a poco se está dejando de lado los sistemas de escritorio a causa de que tendríamos que enfocarnos en especificaciones de sistema operativo para poder ejecutarlo, en cambio en las páginas web solo necesitamos de un navegador e internet para acceder a ellas.

Las ventajas de contar con una página web son:

- La manipulación de los datos se puede realizar desde distintos puntos de ubicaciones para realizar las operaciones que se desee.
- Una empresa, al contar con una página web, obtiene distintas posibilidades de acceso para contar con su servicio.
- Existe una gran comunicación entre los usuarios que cuentan con esta página.
- Toda la información que se tenga de una empresa, puede ser publicada para el conocimiento de todo el mundo.

1.6.1.4. Front-end

Es la parte frontal y enmascara a una aplicación web, con respecto al diseño, fondo, colores, animaciones así como efectos, que interactúa con el usuario mejorando la experiencia al navegar por la aplicación.

1.6.1.5. Paradigma de desarrollo de componentes

La arquitectura de software de una aplicación implementada bajo el paradigma del desarrollo de componentes consiste en ensamblar diferentes componentes prefabricados y que estos proporcionen los servicios necesarios para que la solución de software que se desarrolla cumpla las especificaciones requeridas.

La reutilización o infraestructura de componentes se define dentro de los marcos de trabajo o frameworks que, en definitiva, son el esqueleto de la aplicación que debe ser adaptado en función de las necesidades específicas de la aplicación a la que haya sido destinado. Este framework encapsula el patrón de la arquitectura (Huércano Ruíz y Villar Cueli, 2015).

En general, la arquitectura de software basado en componentes tiene como objetivos:

- Manipular y dar a conocer la estructura de las aplicaciones.
- Ofrecer un sistema que permita reutilizar toda o parte de dicha estructura para aplicaciones similares.
- Identificar las partes fijas y las que pueden sufrir evoluciones en el proceso.
- Permitir descomponer la solución en otras más pequeñas, con la idea de estudiarla de forma independiente.

1.6.1.6. Herramientas CASE

Ingeniería del Software Asistida por Computadora (CASE) brinda métodos y técnicas ayudando en el ciclo de vida del software. Permite algunas mejoras en la productividad y calidad del software obteniendo resultados esperados (Sommerville, 2005).

Herramientas CASE de alto nivel

Este tipo de herramientas otorga al analista la posibilidad de manejar el diseño de un sistema, ayudar en el modelado de requerimientos funcionales de una organización y a definir el alcance del proyecto (Kenneth Kendall y Julie Kendall, 2005).

Herramienta CASE de bajo nivel

Las herramientas CASE de bajo nivel se utilizan para generar código fuente de computadora, eliminando así la necesidad de programar el sistema (Kenneth Kendall y Julie Kendall, 2005).

Ventajas

- El sistema se genera más rápido.
- Reduce el tiempo.

Rational Rose

Rational Rose es una herramienta cuando se desea modelar en orientado a objetos. Está diseñado para el desarrollo de software de modelado visual para dar soluciones sólidas y de manera eficiente para las necesidades reales relacionadas cliente/servidor. Los modelados en este software ayudará en el avance del proyecto y el mejor entendimiento de los procesos cuando se culmine la documentación y se tenga que pasar a la parte del desarrollo del software propuesto (Terry Quatrani, 2003).

1.6.1.7. Capa de acceso a datos

La capa de acceso a datos es aquella que se comunica con la base de datos para que este le permita acceder a toda la data que es requerida al momento de hacer uso de una aplicación.

Bases de datos

Es el conjunto de datos relacionados entre sí. De esta manera se podrá efectuar las distintas operaciones como registrar, actualizar, consultar y eliminar los datos contenidas en la base de datos (García Mariscal, 2015).

Figura 5. Base de datos
Fuente. García Mariscal. (2015). Diseño de bases de datos relacionales

Las bases de datos son una herramienta para recopilar grandes volúmenes de información y administrarla, para ello es necesario utilizar un sistema gestor de base de datos.

Bases de datos relacionales

Están compuestas por un conjunto de tablas de datos relacionadas entre sí, donde cada tabla debe tener un nombre único, donde una fila representa una relación entre un grupo de valores (Jiménez Capel, 2014).

1.6.2. Metodología de desarrollo

1.6.2.1. RUP

Es un proceso de ingeniería de software que nos brinda un enfoque disciplinado para poder evaluar tareas y responsabilidades dentro de una organización. Tiene la finalidad de entregar un software de calidad que cumplas las necesidades de los usuarios finales dentro de un calendario y presupuesto predecible (Kruchten, 2003).

RUP recomienda 6 buenas prácticas para la ingeniería de software (Sommerville, 2005):

- **Desarrolle el software de manera iterativa:**

  Proyecte incrementos del sistema centrándose en las prioridades del usuario.

  Desarrolle y entregue al inicio del proceso de desarrollo características del sistema de acuerdo a la prioridad.

- **Gestione los requerimientos:**

  Documente detalladamente los requerimientos del cliente.

  Manténganse informado de los cambios de estos requerimientos.

  Analice el impacto de los cambios en el sistema antes de llevarlo a cabo.

- **Utilice arquitecturas basadas en componentes**

  Estructure la arquitectura del sistema en componentes.

- **Modele software visualmente**

  Utilice gráficos de modelado UML para presentar vistas estáticas y dinámicas del software.

- **Verifique la calidad del software**

  Asegure el cumplimiento de estándares de calidad organizacional en el software.

- **Controle los cambios del software**

  Gestione los cambios del software usando un sistema de gestión de cambios y procedimientos y herramientas de gestión de configuraciones.

Ciclo de vida RUP

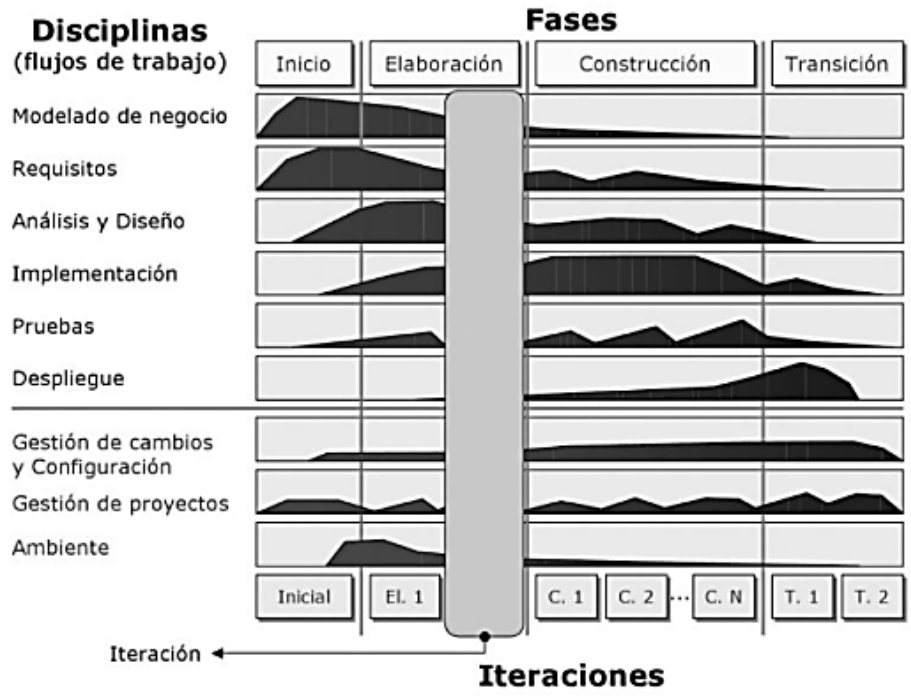


Figura 6. Ciclo de vida RUP

Fuente. Alonso Amo, Martínez Normand y Segovia Pérez. (2005). Introducción a la Ingeniería del Software

Cada iteración consta de seis Flujos de Trabajo Fundamentales o también llamadas Disciplinas de Ingeniería: "Modelado de Negocio, Captura de requisitos, Análisis y Diseño, Implementación, Pruebas y Despliegue" y tres de soporte: "Configuración y Administración de Cambios, Gestión de Proyectos y Ambiente" y finaliza con una versión del producto con un estado cada vez más elaborado.

Una nueva iteración produce una nueva versión que va agregando más funcionalidad al software. RUP tiene 4 fases: "Inicio, Elaboración, Construcción y Transición". Cada fase termina con un hito que representa los objetivos que se han definido para alcanzar. Al finalizar cada hito la dirección del proyecto es la encargada de decidir si el trabajo continúa con la siguiente fase (Alonso Amo, Martínez Normand y Segovia Pérez, 2005).

Fases de RUP

RUP presenta cuatro fases (Sommerville, 2005):

1. **Inicio:** En esta fase se establece un caso de negocio para el sistema. Se identifica las entidades externas y se define sus interacciones con el sistema. Esta información se utiliza para evaluar la aportación del sistema hacia el negocio.
2. **Elaboración:** En la fase de elaboración se debe desarrollar la comprensión del dominio del problema, establecer un marco de trabajo para la arquitectura del sistema, desarrollar el plan del proyecto e identificar los riesgos más importantes. Al finalizar la fase, se debe contar con un modelo de requerimientos del sistema, una descripción de la arquitectura y un plan de desarrollo del software.
3. **Construcción:** Los objetivos de esta fase es diseñar, programar y probar el sistema. Se desarrollan e integran los componentes del sistema. Al culminar esta fase se debe tener el sistema software operativo, así como la documentación necesaria.
4. **Transición:** Es la fase final del RUP, donde el sistema desarrollado se traslada hacia un entorno real. Al terminar la fase de transición, se debe contar con un sistema software documentado funcionando correctamente.

Disciplinas de RUP

Según (Alonso Amo, Martínez Normand y Segovia Pérez, 2005):

1. **Modelado de Negocio:** El modelado de negocio tiene como propósito:
   - Entender los problemas actuales en la organización e identificar las mejoras.
   - Evaluar el impacto que pueda generar el cambio organizacional.

- Los clientes, usuarios y desarrolladores deben entender el funcionamiento de la organización.

2. **Captura de requisitos:** El cliente y los desarrolladores deben de tener en claro lo que debe y no debe hacer el sistema. Se plantea los siguientes pasos:
   - Enumerar los requisitos candidatos
   - Comprender el contexto del sistema
   - Definir los requisitos funcionales y no funcionales
3. **Análisis:** Se analiza los requisitos funcionales y no funcionales del sistema con la finalidad de comprenderlos y refinarlos. El resultado del análisis es el modelo de análisis. Se debe considerar los siguientes pasos:
   - Analizar la arquitectura
   - Analizar los casos de uso
   - Analizar las clases
4. **Diseño:** Se diseña el modelo físico del sistema**.** Se define las siguientes actividades:
   - Diseñar la arquitectura
   - Diseñar los casos de uso
   - Diseñar las clases
   - Diseñar los subsistemas
5. **Implementación:** Se prueba los componentes individualmente y se integran al sistema. Se plantea los siguientes pasos:
   - Implementar la arquitectura
   - Integrar el sistema
   - Implementar los subsistemas
   - Implementar las clases

- Realizar las pruebas de unidad

6. **Pruebas:** Se realiza las pruebas de integración y del sistema; y se evalúa los resultados. Se debe realizar los siguientes pasos:
   - Planificar la prueba de cada iteración
   - Diseñar la prueba
   - Implementar la prueba
   - Realizar las pruebas de integración
   - Evaluar los resultados
7. **Despliegue:** Tiene como finalidad de asegurarse que el sistema se encuentre listo para la entrega al cliente. Se define las siguientes actividades:
   - Plan de despliegue
   - Desarrollo del material del soporte
   - Probar el producto en lugar de desarrollo
   - Crear Release
   - Pruebas de la Release Beta
   - Probar el producto en el lugar de instalación
   - Empaquetar el producto

1.6.3. Sistema de gestión de base de datos

1.6.3.1. MySQL

Es un sistema de administración de base de datos relacional con la capacidad de almacenar grandes volúmenes de datos. MySQL cuenta con todo lo necesario para su instalación, así como también distintos niveles de acceso de usuario, administración y protección de datos.

MySQL emplea SQL (Lenguaje de Consulta Estructurado) permitiendo así la creación de base de datos como la manipulación de datos (Ian Gilfillan, 2003).

Características

Según (Oracle, 2020):

- Las bases de datos MySQL son relacionales.
- Es de código abierto: De esta manera puede ser utilizada y modificada por la comunidad adaptándola a sus necesidades, sin tener que pagar.
- Cuenta con un servidor de base de datos rápido, confiable, escalable y fácil de usar.
- MySQL Server funciona en sistemas cliente / servidor o integrados.

1.6.4. Lenguaje de programación

1.6.4.1. Java

Es un lenguaje de programación multiplataforma y multipropósitos, utilizado para el desarrollo de aplicaciones de escritorio, web, sistemas empresariales, videojuegos, etc.

Características

Según (Groussard, 2012):

- **Sencillo:** Su sintaxis se asemeja a los lenguajes de programación C y C++, evitando características semánticas que puedan convertirlo a un lenguaje confuso, complejo y poco seguro.
- **Orientado a objetos:** Java es un lenguaje orientado principalmente a objetos, el cual tiene las siguientes ventajas: tiene un mejor dominio de la complejidad del problema, reutilización sencilla y fácil de corregir.
- **Interpretado:** Java es interpretado por la JVM o Java Virtual Machine (Máquina virtual de Java), teniendo como ventaja no estar recompilando, basta que cada sistema operativo contenga su propia máquina virtual.

- **Robusto:** Este lenguaje de programación se caracteriza por ser fuertemente tipado y estricto. Verifica el código en el tiempo de compilación y ejecución, evitando errores.
- **Multitarea:** Con Java se puede desarrollar aplicaciones que realicen procesos de manera simultánea ya que cuenta con manejo de hilos y así aprovechar toda la capacidad del CPU.

1.6.5. Arquitectura de software

1.6.5.1. SPA (Single Page Application o Aplicaciones de una sola página)

En el pasar de los tiempos, el desarrollo de aplicaciones web ha avanzado en evolución, centrándose principalmente en otorgar una buena y mejor experiencia al usuario, por lo cual su popularidad en estos últimos tiempos ha estado reforzándose con la aparición de frameworks y librerías.

En un SPA inicialmente se cargan todos los códigos HTML, JavaScript, CSS necesarios y cuando la página lo requiera se cargan solo los recursos dinámicos. No se necesita refrescar la página para ir mostrando su contenido, el manejo del DOM es muy importante (Arizmendi, 2018).

1.6.5.2. Patrón DAO

El patrón DAO o patrón de objeto de acceso a datos, nos brinda una conexión entre la lógica de acceso a datos y la lógica de negocio. Este patrón abstrae y encapsula todo acceso a la base de datos (Trottier, 2003).

Beneficios

Según (Torres Remon, 2012):

- Separa el acceso a datos de la lógica de negocio. Esto favorece cuando se trabaja con proyectos de lógica de negocio compleja.
- Encapsulamiento de la fuente de datos.

- Cuando se trabaja con este patrón los datos no se conectan con la fuente de origen, donde los objetos de las clases son los encargados de obtener los datos de las entidades correspondientes para su mantenimiento.

1.6.5.3. MVC

El Modelo-Vista-Controlador (MVC) es un patrón de diseño que permite construir aplicaciones de forma modularizada, centrándose principalmente en las interfaces de usuario. Tiene como objetivo principal, que al momento de realizar cambios en una parte afecte en lo mínimo en otras partes de la aplicación. Nos permite dividir el proyecto en 3 partes independientes, comunicándose entre sí para llevar a cabo ciertas tareas (Caballé y Xhafa, 2007).

- **Capa Modelo:** Es el modelo del dominio de la aplicación.
- **Capa Vista:** Es la interfaz de usuario.
- **Capa Controlador:** Es la parte de la aplicación que se encarga de gestionar los eventos creados por las acciones que realiza el usuario. Informa a la capa del modelo y/o vista sobre los cambios realizados.

  En otras palabras, al recibir un evento realizado por la acción del usuario (capa vista); el controlador, se encargará de gestionar el evento, accediendo a la capa modelo.

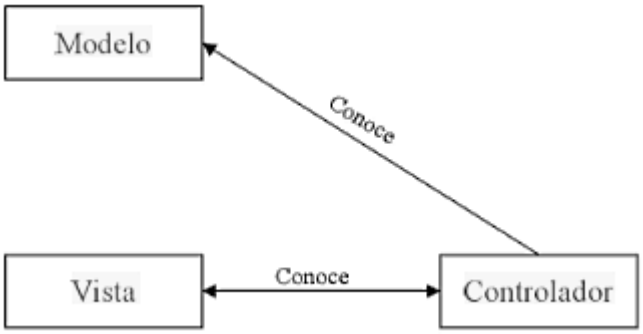


Figura 7. Diseño MVC

Fuente. Alonso Amo, Martínez Normand y Segovia Pérez. (2005). Introducción a la Ingeniería del Software

1.6.6. Servicios web

1.6.6.1. RESTful

REST significa transferencia de representación de estado y es un estilo arquitectónico para diseñar aplicaciones de red distribuida. Roy Fielding acuñó el término REST y propuso los siguientes seis principios (Balaji Varanasi y Sudha Belida, 2015):

- **cliente-servidor:** Se utiliza la arquitectura cliente-servidor para separar los componentes y trabajen de forma independiente.
- **Sin estado:** La comunicación entre el cliente y el servidor debe ser sin estado.

  El servidor no necesita recordar el estado del cliente. En su lugar, los clientes deben incluir toda la información necesaria en la solicitud para que el servidor pueda comprenderla y proceder.
- **Sistema en capas:** Pueden existir múltiples capas jerárquicas como puertas de enlace, firewalls y proxies entre el cliente y servidor. Las capas se pueden agregar, modificar, reordenar o eliminar de forma transparente para mejorar la escalabilidad.
- **Las respuestas del servidor deben declararse como almacenables en caché o no almacenables en caché:** Esto permitiría al cliente o sus componentes intermediarios almacenar en caché las respuestas y reutilizarlas para una solicitud posterior. Esto reduce la carga en el servidor y ayuda a mejorar el rendimiento.
- **Interfaz uniforme:** Todas las interacciones entre el cliente, el servidor y los componentes intermedios se basan en la uniformidad de sus interfaces. Esto

simplifica la arquitectura general, ya que los componentes pueden evolucionar de forma independiente.

- **Código a pedido:** Los clientes pueden ampliar su funcionalidad descargando y ejecutando código a pedido. Los ejemplos incluyen scripts de JavaScript, subprogramas de Java, etc. Esta es una restricción opcional.

Para que una API se considere de RESTful, debe cumplir con los 6 principios de la arquitectura REST.

1.6.7. Entorno de desarrollo

1.6.7.1. Visual Studio Code

Más conocido como VS Code. Es un editor de código fuente muy popular, centrado en la codificación rápida, adaptándose a la comunidad de desarrolladores (Ahmed Khan y Habib, 2020).

Está disponible en los sistemas operativos Windows, macOS y Linux. Tiene soporte para JavaScript, TypeScript y Node.js; cuenta con extensiones para lenguajes como C ++, C #, Java, Python, PHP, Go (Microsoft, 2020). Además, permite la integración con el control de versiones GIT (Ahmed Khan y Habib, 2020).

1.6.7.2. Apache Netbeans 11

Es un entorno de desarrollo integrado (IDE) que nos ayuda en la sintáctica y semántica del código fuente, además de refactorizarlo fácilmente (The Apache Software Foundation, 2020).

1.6.8. Servidor

1.6.8.1. XAMPP

Es un software libre y forma parte de la distribución de Apache (Carrión Bou, Noriega y Del Castillo, 2019).

Se instala de manera sencilla independientemente al sistema operativo (Linux, MAC, Solaris o Windows). XAMPP permite probar tu página web sin tener la necesidad de acceder a internet.

1.6.9. Frameworks y librerías

1.6.9.1. JavaServer Faces 2.3

JSF es un framework para crear interfaces gráficas en aplicaciones web, reutilizando componentes en una página bajo la arquitectura MVC (Martín Sierra, 2011)**.**

Características

JSF presenta las siguientes características:

- El código para crear vistas es similar al HTML estándar.
- Facilita las validaciones, conversiones y mensajes de error.
- Se puede incrustar código JavaScript en la página JSF.
- Simplifica la construcción de interfaces de usuario a partir de componentes reutilizables.

Sin embargo, con la llegada de JSF 2.0 se adiciona más funcionalidades como (Hlavats, 2009):

- Nuevas anotaciones que facilitan la configuración.
- La navegación puede ser implícita, sin necesidad de especificar las reglas de navegación en faces-config.xml.
- Facelets se integra en el marco principal de JSF.
- Nueva biblioteca de etiquetas compuestas para crear componentes compuestos.
- Soporte incorporado para agregar capacidades Ajax, utilizando la etiqueta <f:ajax>.

En JSF 2.3 se hace uso de CDI ya que las anotaciones de los beans gestionados quedan obsoletos (Oracle, 2017).

Ciclo de vida de JSF

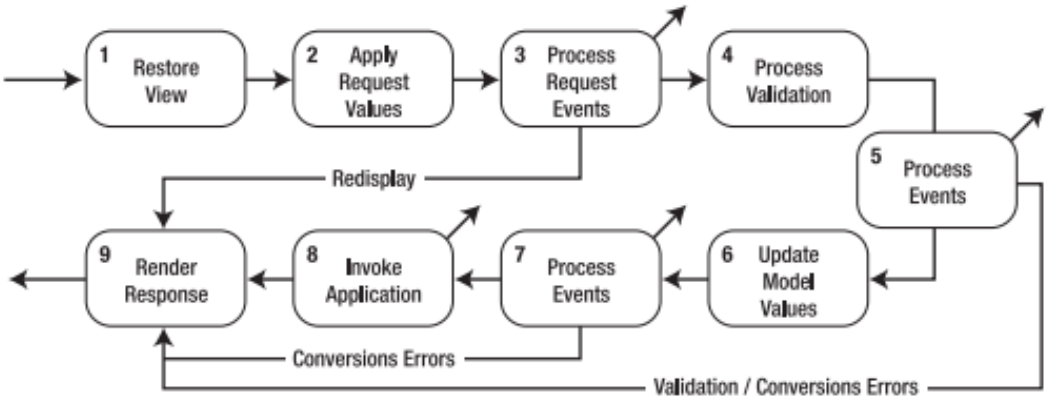


Figura 8. Ciclo de vida de JSF

Fuente. Zambon y Sekler. (2007). Beginning JSP, JSF, and Tomcat Web Development

En la figura 8, en los bloques 3, 5, 7 y 8 observamos flechas hacia arriba, lo cual representa que el ciclo de vida se podría interrumpir. Esto se debe a que la respuesta que se generará no contiene ningún componente JSF.

Dicho lo anterior, JSF ejecuta el método FacesContext.responseComplete abandonando el procesamiento de la solicitud (Zambon y Sekler, 2007).

El ciclo de vida comprende las siguientes fases:

1. **Restaurar Vista:** El servlet JSF crea la vista de la página solicitada en forma de un árbol de componentes donde se almacena la información asociada con todos los componentes. Cuando se crea la vista por primera vez, se guarda en una instancia de FacesContext. Si la petición es por segunda vez se restaura el árbol de componentes.
2. **Aplicar valores de solicitud:** El servlet JSF pasa por el árbol de componentes y ejecuta el método de decodificación de cada componente para extraer los nuevos valores de los parametros de la solicitud

almacenándolos localmente en el componente. Convierte automáticamente los parámetros asociados con las propiedades del objeto. Los errores de conversión hacen que los mensajes de error formen una cola en el objeto FacesContext que posteriormente se visualizará en la fase de renderizado de respuesta. En ciertos casos, cuando el usuario hace click en los controles, se genera eventos de solicitud y se pone en cola en FacesContext.

3. **Procesar eventos de solicitud:** El servlet ejecuta el método processEvent para cada componente con uno o más eventos en cola. Cada componente decide manejar los eventos o delegar su manejo a los controladores de eventos. Si todos los métodos processEvent ejecu tados devuelven falso, el servlet continúa con la siguiente fase, sino salta a la fase de renderizado de respuesta.
4. **Procesar validaciones:** El servlet invoca los métodos de validación de los validadores que se habían registrado durante la fase restaurar vista. Examina las reglas de atributos del componente para su validación y compara estas reglas con el valor local almacenado para el componente. Si el valor local no es válido, el servlet pone en cola un mensaje de error en FacesContext.
5. **Procesar eventos:** Si existieran errores de validación o conversión durante la fase de procesar validaciones, se pasa a la fase de renderizado de respuesta.
6. **Actualizar los valores del modelo:** Se copian los valores locales en los beans mediante la ejecución del método updateModel, realizando también conversiones de tipos cuando sea necesario. Los errores de conversión hacen que los mensajes de error se pongan en cola en FacesContext.

7. **Procesar eventos:** Si se generaron errores de conversión en la fase anterior, se pasa a la fase de renderizado de respuesta.
8. **Invocar la aplicación:** Se invoca el método asociado a la acción del botón, link, etc que realizó el usuario para que se activara el ciclo de vida de la solicitud.
9. **Renderizar la respuesta:** El servidor devuelve la página solicitada al usuario, guardando el estado actual de la vista para restaraurse en una solicitud posterior.

1.6.9.2. Contextos e Inyección de Dependencia (CDI)

CDI proporciona una arquitectura donde todos los componentes de Java EE como: Servlets, Enterprise JavaBeans y beans gestionados siguen el mismo modelo de programación y ciclo de vida, permitiendo que estos componentes se acoplen e inyecten donde se requieran (Saleh, Christensen y Wadia, 2014).

1.6.9.3. Facelets

Es un sistema de plantillas web para JSF basado en composiciones. Una composición define una estructura JSF UIComponents en una página Facelets (Oracle, 2021).

Características

Entre sus características más importantes tenemos (Ceballos Sierra, 2015):

- Reutilización de código con el uso de plantillas.
- Fácil mantenimiento de las aplicaciones debido al uso de plantillas.
- Permite el uso del lenguaje XHTML.
- Permite el uso del lenguaje de expresión (EL – Expression Language).
- Soporta el uso de diferentes bibliotecas de etiquetas como:

  - JavaServer Faces Facelets Tag Library (prefijo ui:)

- JavaServer Faces HTML Tag Library (prefijo h:)
- JavaServer Faces Core Tag Library (prefijo f:)
- JSTL Core Tag Library (prefijo c:)
- JSTL Functions Tag Library (prefijo fn:)

- Las vistas Facelets son generalmente creadas como páginas XHTML.

1.6.9.4. jQuery

jQuery ha sido creada con los propósitos principales de realizar solicitudes Ajax de manera sencilla y tener el control de los elementos de una página web. En base a jQuery se desarrolló JQuery UI, un conjunto de controles, widgets como, por ejemplo: acordeones, pestañas, etc, para ser utilizados, además nos permite implementar nuestras propias funcionalidades.

Esta herramienta ayuda a mejorar la experiencia de usuario realizando un diseño llamativo y moderno (Castledine y Sharkie, 2012).

Características

Según (Pollock, 2014):

- **Acceso a elementos:** No ese necesario aprender todo sobre JavaScript ya que con la sintaxis CSS se puede acceder a los elementos.
- **Cambios en el documento:** jQuery nos ofrece varias funcionalidades para realizar cambios sobre elementos de la página web de manera sencilla.
- **Creación de efectos:** Se puede crear efectos y animaciones, gracias a los diversos métodos que nos ofrece.
- **Ajax:** Esta herramienta nos permite mejorar la carga de código al momento de manipular información del servidor.

1.6.9.5. React.js

React.js es una librería del lenguaje de programación JavaScript. Intenta resolver los problemas desde la capa web. Es utilizado en la vista en cualquiera de los frameworks MVC. No es estricto al momento de usarlo. Divide la vista en varios componentes, estos abarcan tanto la lógica de visualización de la página, así como la propia vista. Puede contener datos que utiliza para representar el estado de la aplicación (Vipul y Prathamesh, 2016).

React se basa en la idea de que la manipulación del DOM es una operación costosa y debe minimizarse. También reconoce que la optimización manual de la manipulación del DOM dará como resultado una gran cantidad de código repetitivo, que es propenso a errores.

Por ello, React resuelve esto, ofreciendo al desarrollador un DOM virtual para renderizar en lugar del DOM real (Vipul y Prathamesh, 2016).

Funciona de la siguiente manera:

1. Al cambiar el estado del modelo de datos, React renderizará la interfaz de usuario a una representación del DOM virtual.
2. Después React realiza la diferencia entre las dos representaciones DOM virtuales; el DOM virtual anterior que se calculó antes de que se cambiaran los datos y la representación del DOM virtual actual que se calculó después de los cambios de los datos. La diferencia entre las dos representaciones de DOM virtuales es lo que se necesita cambiarse en el DOM real.
3. React actualiza lo que es necesario en el DOM real.

El proceso de encontrar la diferencia entre las dos representaciones del DOM virtual y representar solo los cambios actualizados en el DOM real es rápido.

React permite renderizar todo el DOM cuando exista algún cambio en el estado de la aplicación (Fedosejev, 2015).

Características

React tiene las siguientes características principales (Bugl, 2019):

- **Declarativo:** A React le decimos lo que queremos hacer y no cómo hacerlo. De esta manera, podemos diseñar de manera fácil nuestras aplicaciones y React actualizará y renderizará eficientemente los componentes cuando cambien los datos.
- **Basado en componentes:** Encapsula componentes que administran su propio estado y vistas para después unir diferentes partes y crear interfaces de usuario complejas.

  En React existe dos tipos de componentes:

  - **Componentes de clase:** Clases de JavaScript que proporcionan un método llamado render que devuelve la interfaz de usuario.
  - **Componentes de función:** Funciones de JavaScript que toma las props como argumentos y devuelve la interfaz de usuario.
- **Aprenda una vez, escriba en cualquier lugar:** React se asegura de que se pueda desarrollar aplicaciones evitando repetir tanto código existente.

Ventajas

React nos ofrece diversas ventajas entre las más fundamentales tenemos (Rajput, 2019):

- **Rendimiento de las aplicaciones:** El uso del DOM virtual mejora el rendimiento de la aplicación ya que es mucho más rápido que el DOM real.

- **Reutilización de código:** Debido a que React se basa en componentes evita la duplicidad de código, dando la posibilidad de mostrar un componente en diferentes vistas.
- **Legibilidad de código:** El uso de componentes ayuda a mejorar la legibilidad y por lo tanto facilita el mantenimiento de aplicaciones más grandes.

1.6.9.6. React Hooks

Los Hooks aparecen en la versión 16.8 de React, dejando de lado los componentes de clases. Son funciones que se llaman en los componentes de funciones. Ya no es necesario crear componentes de orden superior como se venía utilizando en los componentes de clases, ahora los hooks nos permite reutilizar la lógica con estado entre componentes (Bugl, 2019).

Ventajas

- Son mucho más declarativos por lo que encaja mejor en el ecosistema de React.
- Facilitan la abstracción y el intercambio de lógica de estado común entre componentes.
- Los hooks hacen nuestro código más claro y fácil de optimizar para React.
- Puedes crear tus propios hooks personalizados.

Reglas

Existen ciertas limitaciones con React Hooks que siempre debemos tener en cuenta:

- Los Hooks solo se pueden llamar en componentes de funciones.
- No llamar a los Hooks en condicionales, bucles o funciones anidadas.

1.6.9.7. React Router

La sincronización de la URL del navegador con lo que se representa en la página web se denomina enrutamiento. React Router maneja el enrutamiento de forma declarativa, permitiendo controlar el flujo de datos de la aplicación (Wanyoike, M., Franklin, Teller y Bouchefra, 2017).

Esta biblioteca comprende tres paquetes:

- **react-router:** Paquete principal.
- **react-router-dom:** Se utiliza para aplicaciones web.
- **react-router-native:** Se utiliza para aplicaciones móviles.

1.6.9.8. Bootstrap 4

Es un framework front-end de código abierto que nos ayuda a tener una codificación sencilla y rápida para darle diseño a nuestras páginas web. Ofrece distintos componentes como formularios, botones, cuadros, menús de navegación, entre otros. Se basa en las tecnologías de HTML y CSS, además de extensiones de JavaScript para una mejor interactividad (Carrión Bou, Noriega y Del Castillo, 2019).

1.6.9.9. React Bootstrap 1.6

Es un marco similar a Bootstrap basado en JavaScript para aplicaciones en React. Es una reimplementación completa de los componentes reutilizables front-end de Bootstrap en React. Además, es independiente ya que no necesita incluir jQuery u otro marco como dependencia en su proyecto (Singh y Bhatt, 2016).

1.6.10. Tecnologías web

1.6.10.1. HTML 5

Para crear páginas web, los desarrolladores usan un lenguaje de computadora llamado lenguaje de marcado de hipertexto (HTML) …para definir formato de

contenido (Jamsa, 2014). De esta manera son entendidos perfectamente por los navegadores para construir una página web.

1.6.10.2. JavaScript

JavaScript es un lenguaje de programación cuya finalidad es dar más dinamismo a las páginas web, en otras palabras, anteriormente solo con HTML se podían hacer stios web estáticos y con algunas animaciones básicas, en cambio con JavaScript, gracias a sus funcionalidades, se puede tener más interactividad con los usuarios (Sánchez Maza, 2001).

Por ejemplo:

- Podemos hacer que al pasar el mouse encima de una foto estas cambien.
- Mostrar ventanas emergentes para mostrar publicidad.
- Interactuar con el usuario mediante formularios.

Características

- Es un lenguaje basado en el paradigma orientado a objetos.
- Con esta tecnología se puede llegar a manejar eventos.
- Es independiente de la plataforma ya que es un lenguaje interpretado por el navegador.

1.6.10.3. CSS 3

CSS es la abreviatura de Cascading Style Sheet (Hojas de estilo en cascada). La finalidad de esta tecnología es darle estilos a una página estructurada con HTML. Podemos hacer una infinidad de cosas que con simple HTML no se puede, como poner un tipo de letra, para que sea llamativo al momento que el cliente visite la página, aplicar colores, mejorar la infraestructura, etc. CSS está diseñado para separar el contenido de la presentación de las páginas web.

Estos estilos son almacenados en hojas de estilos y fueron agregados en la codificación HTML para resolver algunos problemas. Las hojas de estilos externos, ahorra y facilita el trabajo (Egea García, 2008).

Ventajas

Según (Durango, 2015):

- Las hojas de estilos externas nos ayudan a disminuir el tiempo ya que cuenta con una gran flexibilidad al momento de desarrollar páginas web.
- Al contar con archivos externos, la modificación de estilos se nos hace muy sencillo.

Además, contamos con las ventajas de:

- Crear páginas web responsivas adaptándose a cualquier dispositivo como móviles, tablets, etc.
- Al crear hojas de estilos externas, se pueden llamar en varias páginas diferentes para su uso y así reutilizar el diseño.

1.6.10.4. SASS

SASS (Syntactically Awesome Style Sheets) es un lenguaje basado en CSS más maduro, estable y poderoso (Sass, 2021). Incluye un compilador para traducir los ficheros en sintaxis SASS a CSS.

Características

- Compatible con todas las versiones de CSS.
- Mayor cantidad de funciones a diferencia de otro lenguaje de extensión CSS.
- Cuenta con el apoyo de una gran comunidad.
- Permite usar funciones, variables, mixins, reglas, etc.

- Facilita compartir el diseño entre proyectos teniendo una mejor organización.

1.6.10.5. ECMAScript

ECMAScript fue creado en el año 1997 bajo el cargo de la organización ECM Internacional. Sus siglas provienen de European Computer Manufacturers Association, estuvo inspirado en lenguajes como Java y C++ (Puciarelli, 2020).

Como JavaScript es un lenguaje interpretado, ECMAScript es el estándar que rige como este lenguaje debe ser interpretado y funcionar.

1.6.10.6. Ajax

Ajax o también llamado JavaScript Asíncrono y XML comprende un conjunto de técnicas que nos ayuda a mejorar el proceso de peticiones al servidor, transmitiendo pequeños paquetes de datos en segundo plano. Modifica los contenidos de la página creando efectos dinámicos con rapidez (Peña, 2019).

Ventajas

- Uso óptimo de los recursos, debido al ahorro de ancho de banda y tiempo reducido que se realiza en cada transacción.
- Las páginas no son recargadas por completo, ya que la información se recibe por segmentos desde el servidor.
- Soportado por la mayoría de navegadores web.

1.6.10.7. Maven

Maven es una herramienta cuyo propósito es gestionar y crear proyectos de software en el lenguaje de programación Java.

Se basa en un archivo llamado POM (Project Object Model) para describir el software que se va a desarrollar, así como dependencias (base de datos,

frameworks, librerías, plugins, etc) que formarán parte del proyecto (Pérez Martínez, 2015).

Características

Según (Turatti y Pillitu, 2013):

- Compilaciones de Maven se pueden iniciar desde el IDE.
- Se puede agregar dependencias de Maven a la compilación desde un IDE.
- Representada por un archivo llamado POM.
- Las fuentes Java de artefactos son descargados automáticamente.

1.6.10.8. React Icons

Es una biblioteca de iconos en formato SVG distribuidos como componentes de React. Cuando lo instalamos, tendremos acceso a una gran cantidad de iconos SVG (Banks y Porcello, 2020).

1.6.10.9. Node.js

Inicialmente JavaScript funcionaba solo en los navegadores web. Este lenguaje se puede usar de diferentes formas por lo que se considera un lenguaje completo. Ahora JavaScript se puede correr en el lado servidor (backend) haciendo uso de Node.js.

Node.js interpreta y ejecuta código JavaScript haciendo uso de la máquina virtual V8 de Google. Además, nos brinda varios módulos, por lo que no se necesita escribir código desde cero (Kiessling, 2011).

1.7. Diseño de arquitectura

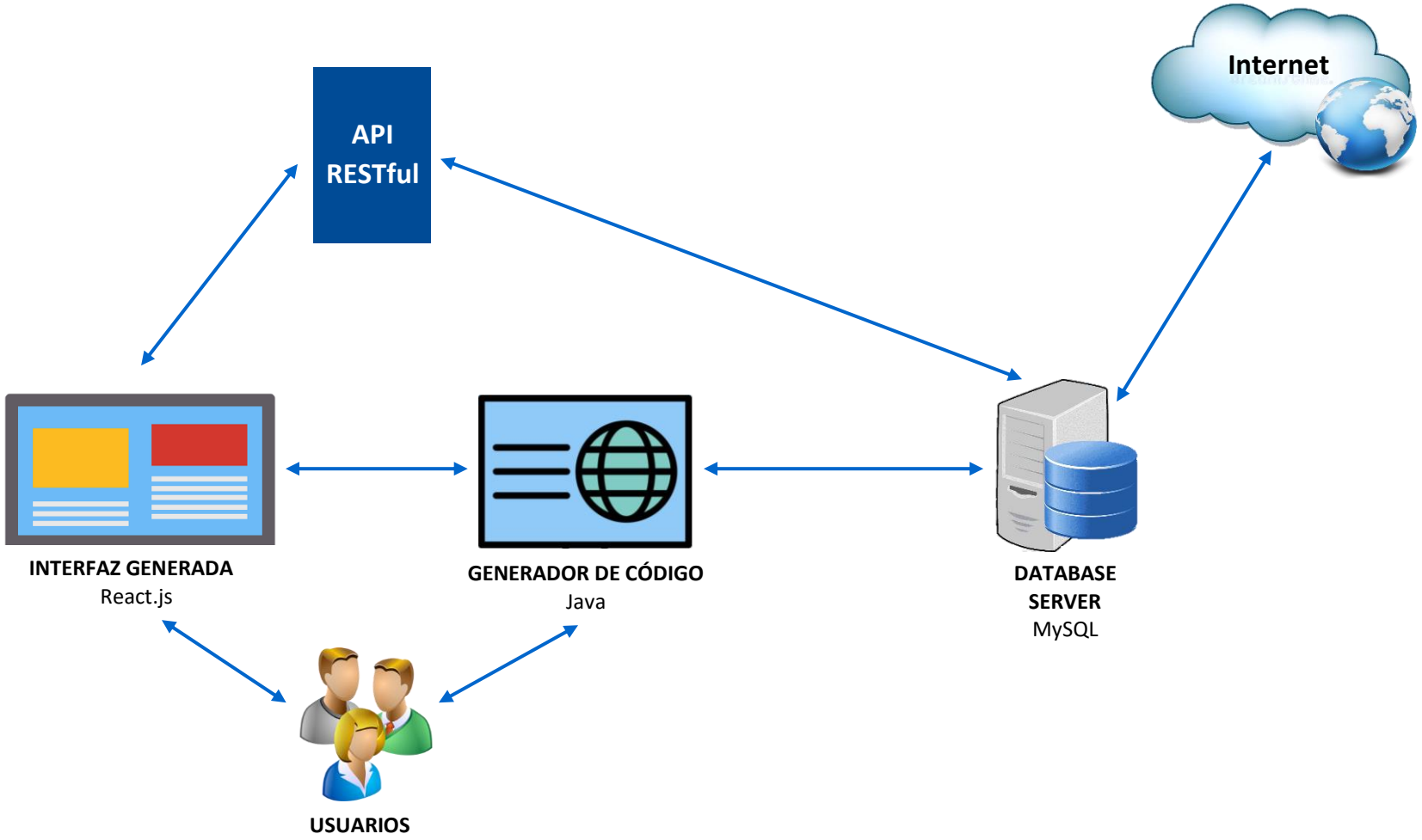


Figura 9. Diseño de arquitectura

Fuente. Elaboración propia

Capítulo II. Métodos y Materiales

2.1. Diseño de contrastación de la hipótesis

La presente investigación es de tipo pre experimental, que tendrá como uno de sus productos la herramienta GENCODE; se plantea contrastar la hipótesis de la siguiente manera:

GE: O1 ⟶ X ⟶ O2

O1 = Tiempo de codificación manual a realizar antes de implantar GENCODE.

X = Herramienta GENCODE.

O2 = Tiempo de codificación apoyado por GENCODE, a realizarse después de la capacitación en el uso de la herramienta.

Las unidades de análisis están conformadas por estudiantes de Ingeniería en Computación e Informática.

2.2. Población y muestra

La población la conformarán todos los programadores que requieran construir aplicativos webs.

La muestra será conformada por 10 estudiantes del X ciclo de la Escuela Profesional de Ingeniería en Computación e Informática de la Universidad Nacional Pedro Ruiz Gallo, con la finalidad de garantizar su experiencia en programación.

2.3. Técnicas e instrumentos de la recolección de datos

2.3.1. Técnicas

- **Observación**

  Es una técnica en la cual se observa detalladamente un fenómeno en la cual se desarrollará un análisis para la investigación (Huamán Valencia, p. 13).

- **Encuesta**

  Garza Mercado (2007) afirma:

  Cuando realizamos algún trabajo de investigación y queremos recolectar testimonios, ya sea de manera oral o escrita, podemos hacer uso de encuestas. Esta técnica tiene como propósito conseguir o averiguar hechos, opiniones como también actitudes. Tenemos 2 tipos: la encuesta de hechos y la encuesta de actitudes y opiniones. El primer tipo de encuesta sirve para recolectar toda información acerca del conocimiento de las personas y la segunda, obtener información de lo que piensan o sienten.

- **Fichaje**

  Huamán Valencia (2005) afirma:

  El fichaje está considerado como una técnica auxiliar. El principal objetivo de esta técnica, es registrar los datos que se van obteniendo mediante el uso del instrumento llamado fichas, las cuales son muy importantes ya que contienen la información obtenida de una investigación.

2.3.2. Instrumentos

- **Ficha de observación**

  La ficha de observación son instrumentos que se utilizan como apoyo para una investigación cuyo objetivo es la recolección de datos referidos a las variables específicas.

- **Cuestionario**

  Este instrumento tiene el objetivo de conseguir información mediante preguntas ordenadas de manera coherente y comprensible. Generalmente no es necesario la intervención de un encuestador, la persona interrogada responde por escrito. (García Córdoba, 2004).

- **Ficha Bibliográfica**

  Gómez Salazar, Gonzáles Téllez y López Gonzáles (2004) afirman:

  > Describe los datos principales de un libro; el autor, el título de la obra, el número de la edición que le corresponde, quién fue su traductor, donde se editó, cuál fue la casa editora, el año en que apareció la obra, cuántas páginas tiene y, en fin, a que colección pertenece (p. 68).

Capítulo III. Resultados y Discusión

3.1. Desarrollo del generador de código

Para el desarrollo del generador de código se utilizó Java EE 8 junto con el kit de desarrollo en su versión 7 (JDK 7). En el diseño de las interfaces de usuario, se escogió Java ServerFaces en la versión 2.3 (JSF 2.3). Además, se emplea el patrón "Modelo-Vista-Controlador" (MVC) y "Objeto de Acceso a Datos" (DAO), teniendo así un proyecto más organizado y fácil de codificar.

"Gencode" se basa en la metodología "Proceso Racional Unificado" (RUP), la cual consta de 4 fases: inicio, elaboración, construcción y transición.

3.1.1. Fase de inicio

- **Modelo de casos de uso del negocio**

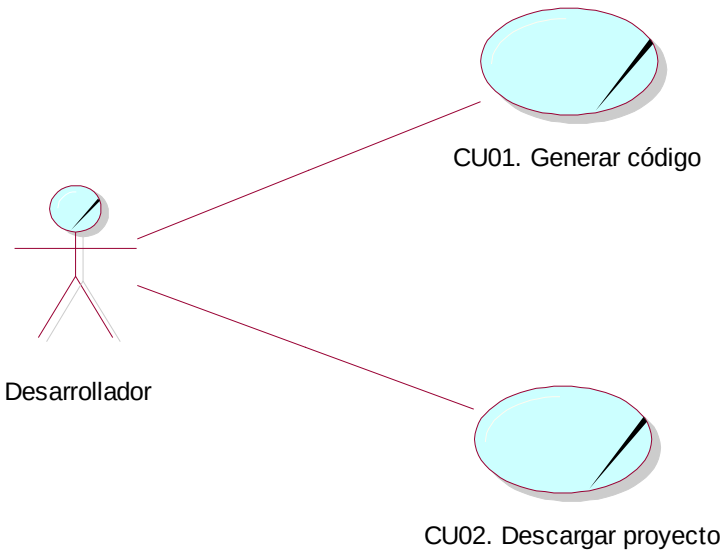


Figura 10. Modelo de casos de uso del negocio

Fuente. Elaboración propia

- **Diagrama de actividades**

DA01: Generar código

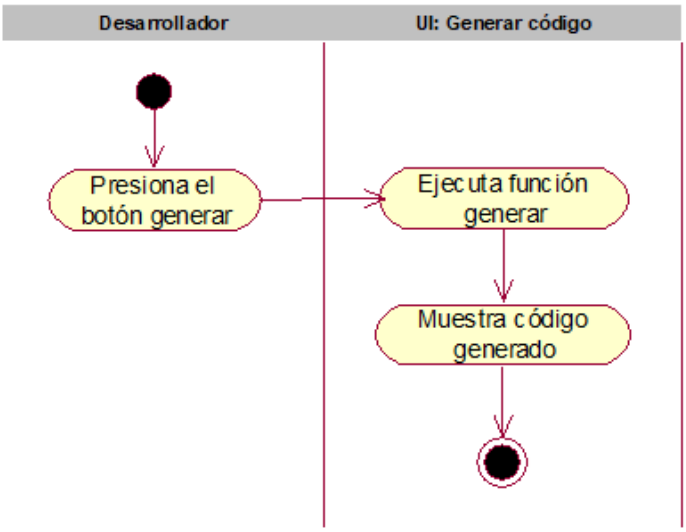


Figura 11. DA01. Generar código

Fuente. Elaboración propia

DA02: Descargar proyecto

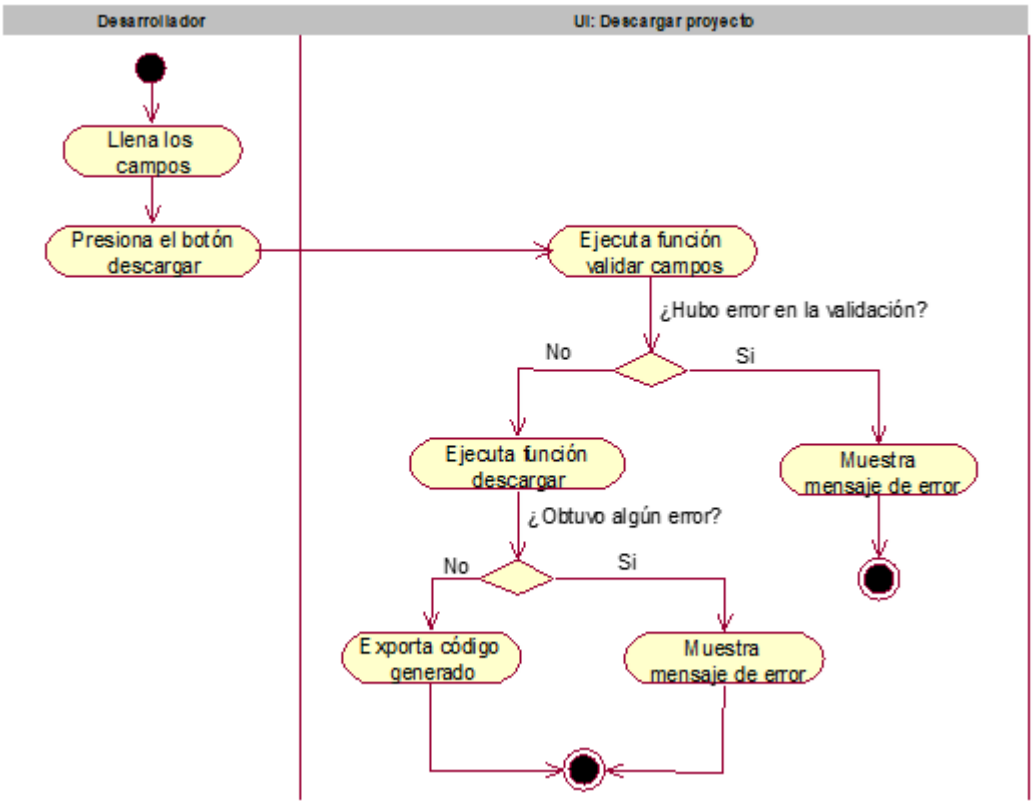


Figura 12. DA02. Descargar proyecto

Fuente. Elaboración propia

- **Modelo de objetos del negocio**

BCU01. Generar código

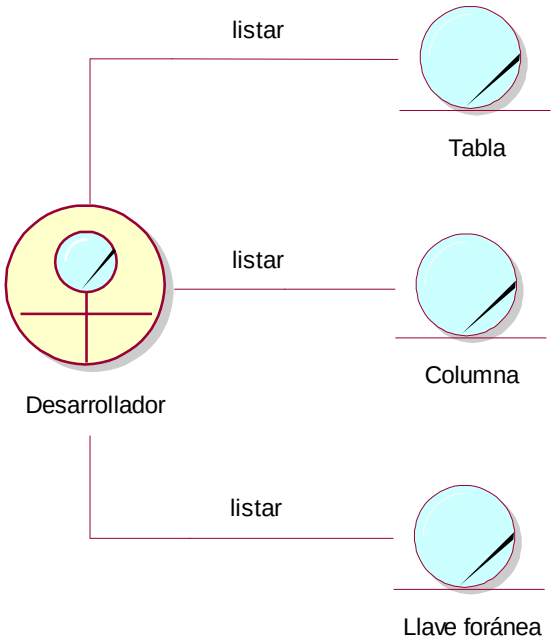


Figura 13. BCU01. Generar código

Fuente. Elaboración propia

BCU02. Descargar proyecto

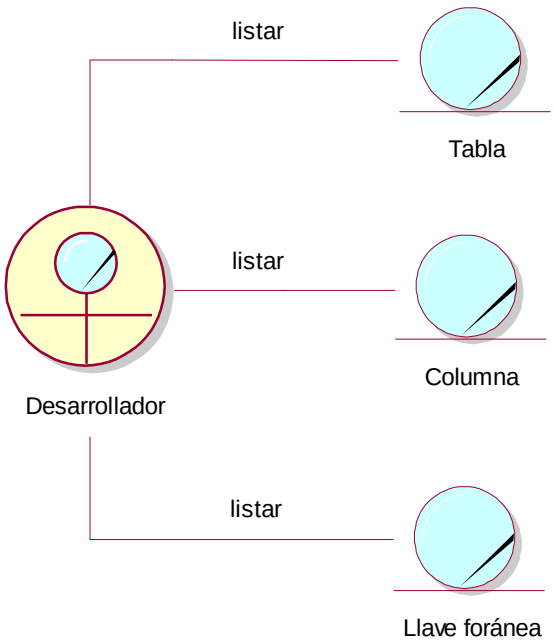


Figura 14. BCU02. Descargar proyecto

Fuente. Elaboración propia

3.1.2. Fase de elaboración

- **Casos de uso**

Los casos de uso describen las interacciones entre el usuario y el sistema. Para el desarrollo del generador de código se identifica 6 casos de uso:

Tabla 2. Casos de uso

N°	Casos de uso
CUS01	Conectar base de datos
CUS02	Guardar datos
CUS03	Generar código
CUS04	Descargar proyecto
CUS05	Descargar código generado

Fuente. Elaboración propia

- **Requisitos funcionales**

En la siguiente tabla, se describe las funcionalidades con las que debe contar la aplicación.

Tabla 3. Requisitos funcionales

N°	Requerimiento Funcional
RF01	Realizar la conexión a la base de datos
RF02	Identificar tablas, columnas y sus tipos de datos.
RF03	Identificar la clave primaria de cada tabla.
RF04	Identificar las claves foráneas y las tablas con las que se crea la dependencia.
RF05	Listar tablas, columnas, claves primarias, claves foráneas.
RF06	Guardar datos
RF07	Generar código
RF08	Descargar proyecto
RF09	Descargar código generado

Fuente. Elaboración propia

- **Requisitos no funcionales**

El generador de código tiene las siguientes características y restricciones.

Tabla 4. Requisitos no funcionales

N°	Requerimiento No Funcional
RNF01	El sistema es compatible JDK 7 y Java EE 8, el generador se puede ejecutar con un navegador actual.
RNF02	El generador de código deberá adaptarse a diferentes dispositivos móviles para una mejor navegabilidad.
RNF03	La base de datos a importar debe ser MySQL v.8.0.12-MariaDB
RNF04	El sistema correrá con el servidor de aplicaciones GlassFish en su versión 5.0.1
RNF05	Todas las tablas de la base de datos deberán contar con llaves primarias autoincrementales.
RNF06	Los tipos de datos soportados deben ser: varchar, char, date, enum, int, decimal, bit.
RNF07	La base de datos debe estar bien diseñada, ya que si contiene errores el código generado podría ser incorrecto.
RNF08	Si se utilizan palabras reservadas en algunos de los campos de las tablas, el sistema no validará, por lo tanto, se generará errores en el código generado.
RNF09	Si el desarrollador no llena campos obligatorios, al dar click en el botón respectivo, deberá pintarse de rojo dichos campos.
RNF10	En cada textarea deberá eliminarse el autocorrector por defecto del navegador.

Fuente. Elaboración propia

- **Descripción de actores**

En la tabla siguiente, se define el actor, que serán las personas que interactuarán con la aplicación.

Tabla 5. Descripción de actores

Actor	Justificación
Desarrollador	Encargado de utilizar la herramienta CASE, importando la base de datos correspondiente para la generación de código de interfaces CRUD.

Fuente. Elaboración propia

- **Modelo de casos de uso del sistema**

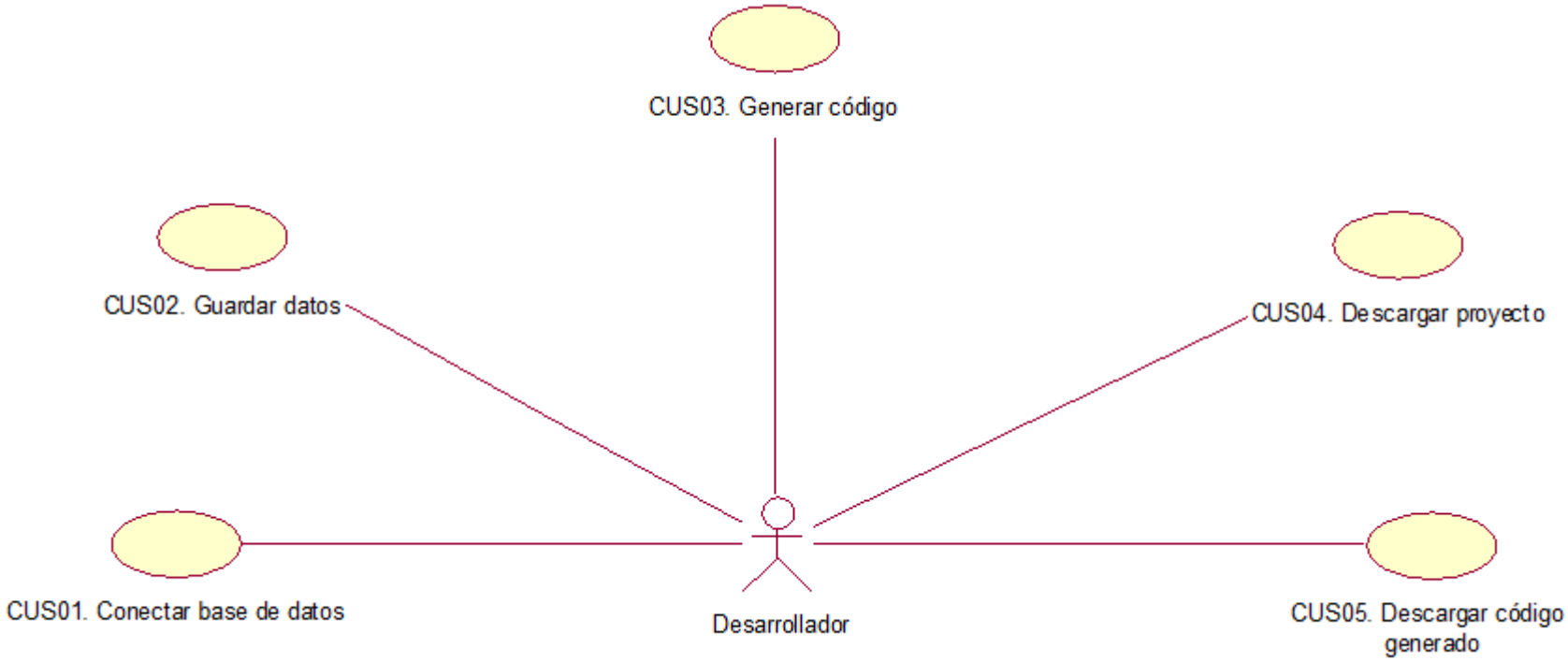


Figura 15. Modelo de casos de uso del sistema

Fuente. Elaboración propia

- **Especificación de Casos de Uso**

CUS01: Conectar base de datos

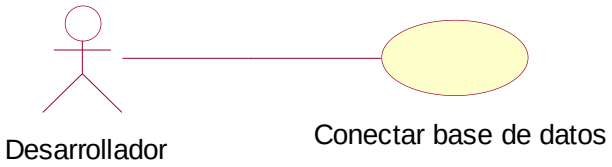


Figura 16. Diagrama CUS01. Conectar base de datos

Fuente. Elaboración propia

Tabla 6. Especificación CUS01. Conectar base de datos

CUS01: CONECTAR BASE DE DATOS		
Descripción: El caso de uso comienza cuando el desarrollador llena todos los campos requeridos y finalmente conecta la base de datos.		
ACTORES	Desarrollador	
PRECONDICIONES	Ninguno	
SECUENCIA	**PASO**	**EVENTO**
FLUJO PRINCIPAL	**1**	El desarrollador ingresa los datos requeridos.
	2	El desarrollador da click en conectar.
	3	El sistema muestra un listado con las tablas, columnas, tipos de datos de la base de datos importada.
FLUJO ALTERNATIVO	**2.1** Si los campos están llenados correctamente, ir al paso 3.	
	2.2 Sino el sistema mostrará una ventana emergente con el error.	
PUNTOS DE EXTENSIÓN	Ninguno	
POSTCONDICIONES	Listar tablas, columnas, tipos de datos.	

Fuente. Elaboración propia

CUS02: Guardar datos

Figura 17. Diagrama CUS02. Guardar datos

Fuente. Elaboración propia

Tabla 7. Especificación CUS02. Guardar datos

CUS02: GUARDAR DATOS		
Descripción: El caso de uso comienza cuando el sistema muestra el listado con los datos de la base de datos y finaliza cuando el desarrollador da click en el botón guardar.		
ACTORES	Desarrollador	
PRECONDICIONES	Ninguno	
SECUENCIA	**PASO**	**EVENTO**
FLUJO PRINCIPAL	**1**	El sistema muestra un listado con las tablas, columnas, tipos de datos de la base de datos importada.
	2	El desarrollador da click en guardar.
	3	El sistema muestra la página de inicio del generador de código.
FLUJO ALTERNATIVO	Ninguno	
PUNTOS DE EXTENSIÓN	Ninguno	
POSTCONDICIONES	Registro de tablas, columnas, llaves foráneas.	

Fuente. Elaboración propia

CUS03: Generar código

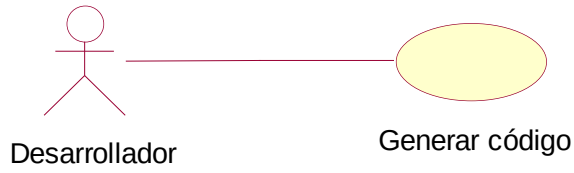


Figura 18. Diagrama CUS03. Generar código

Fuente. Elaboración propia

Tabla 8. Especificación CUS03. Generar código

CUS03: GENERAR CÓDIGO		
Descripción: El caso de uso comienza cuando el desarrollador da click en el botón generar y finaliza cuando se genera el código.		
ACTORES	Desarrollador	
PRECONDICIONES	Ninguno	
SECUENCIA	**PASO**	**EVENTO**
FLUJO PRINCIPAL	**1**	El desarrollador da click en generar.
	2	El sistema genera el código.
FLUJO ALTERNATIVO	Ninguno	
PUNTOS DE EXTENSIÓN	Ninguno	
POSTCONDICIONES	Ninguno	

Fuente. Elaboración propia

CUS04: Descargar proyecto

Figura 19. Diagrama CUS04. Descargar proyecto

Fuente. Elaboración propia

Tabla 9. Especificación CUS04. Descargar proyecto

CUS04: DESCARGAR PROYECTO		
Descripción: El caso de uso comienza cuando el desarrollador llena todos los campos requeridos y da click en el botón descargar.		
ACTORES	Desarrollador	
PRECONDICIONES	Ninguno	
SECUENCIA	**PASO**	**EVENTO**
FLUJO PRINCIPAL	**1**	El desarrollador ingresa los datos requeridos.
	2	El desarrollador da click en el botón descargar.
	3	El sistema muestra el proyecto descargado.
FLUJO ALTERNATIVO	Ninguno	
PUNTOS DE EXTENSIÓN	Ninguno	
POSTCONDICIONES	Descarga del proyecto con código generado.	

Fuente. Elaboración propia

CUS05: Descargar código generado

Figura 20. Diagrama CUS05. Descargar código generado

Fuente. Elaboración propia

Tabla 10. Especificación CUS05. Descargar código generado

CUS05: DESCARGAR CÓDIGO GENERADO		
Descripción: El caso de uso comienza cuando el desarrollador da click en el botón descargar.		
ACTORES	Desarrollador	
PRECONDICIONES	Ninguno	
SECUENCIA	**PASO**	**EVENTO**
FLUJO PRINCIPAL	**1**	El desarrollador da click en el botón descargar.
	2	El sistema muestra el archivo descargado.
FLUJO ALTERNATIVO	Ninguno	
PUNTOS DE EXTENSIÓN	Ninguno	
POSTCONDICIONES	Descarga del archivo con el código generado.	

Fuente. Elaboración propia

3.1.3. Fase de construcción

- **Identificación de paquetes de análisis**

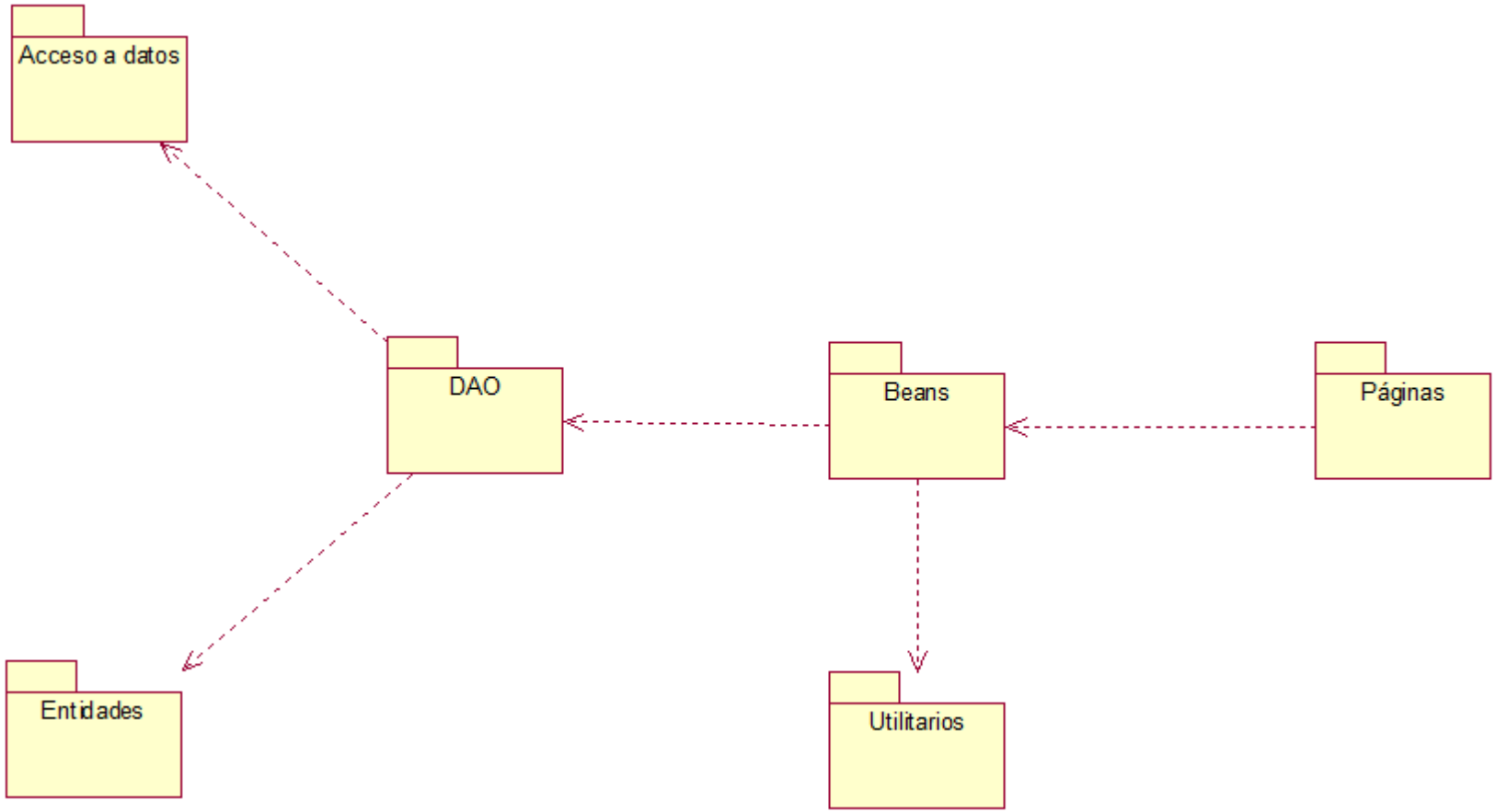


Figura 21. Identificación de paquetes de análisis

Fuente. Elaboración propia

- **Identificación de las Clases de Análisis**

DCU01: Conectar base de datos

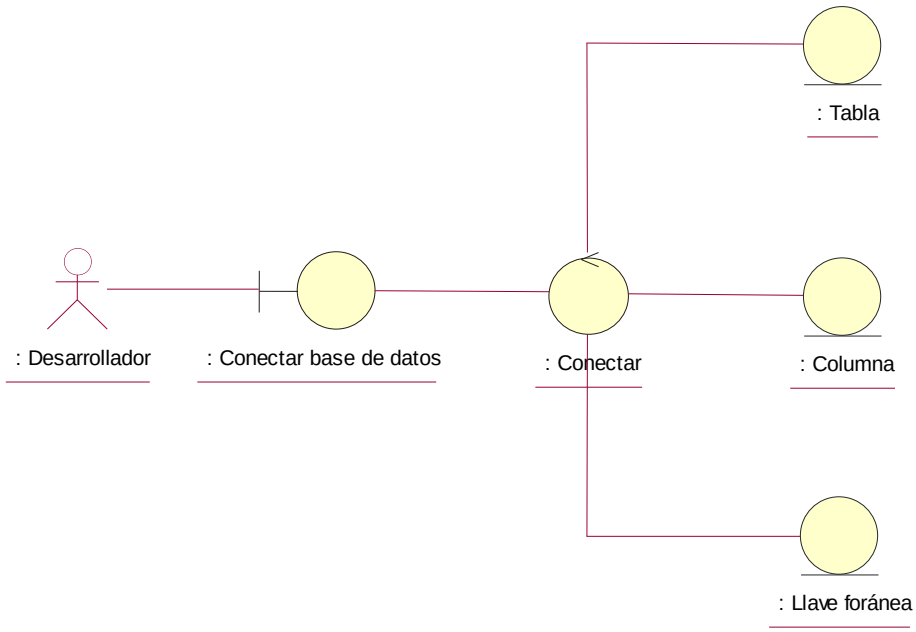


Figura 22. DCU01. Conectar base de datos

Fuente. Elaboración propia

DCU02: Guardar datos

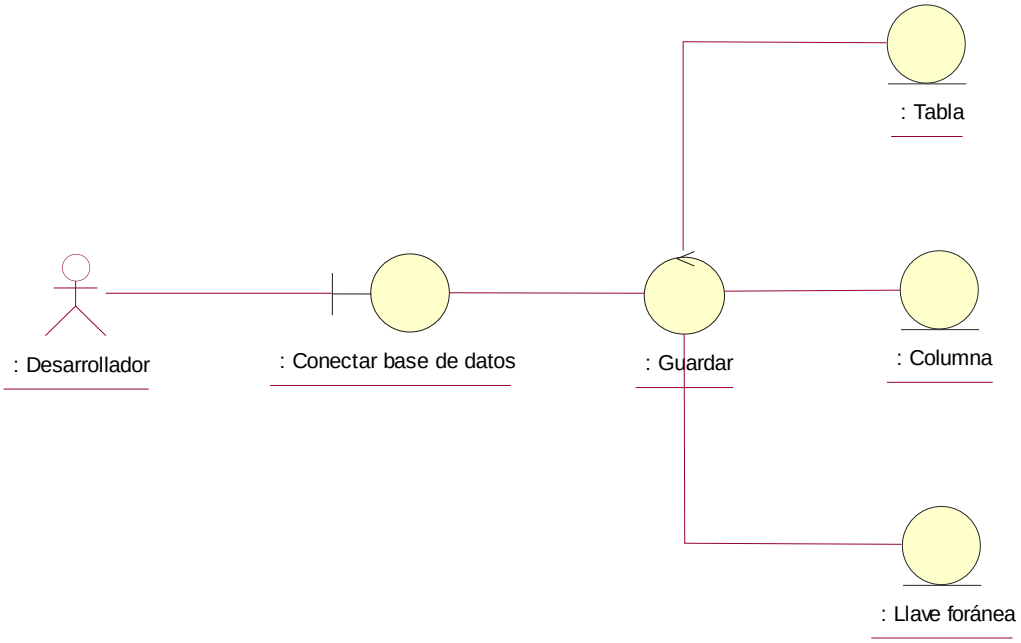


Figura 23. DCU02. Guardar datos

Fuente. Elaboración propia

DCU03: Generar código

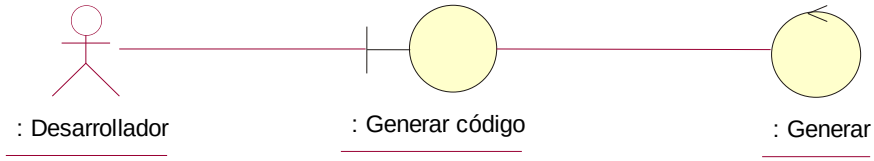


Figura 24. DCU03. Generar código

Fuente. Elaboración propia

DCU04: Descargar proyecto

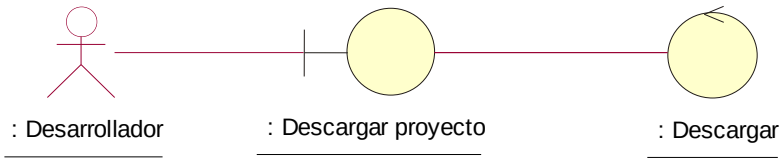


Figura 25. DCU04. Descargar proyecto

Fuente. Elaboración propia

DCU05: Descargar código generado

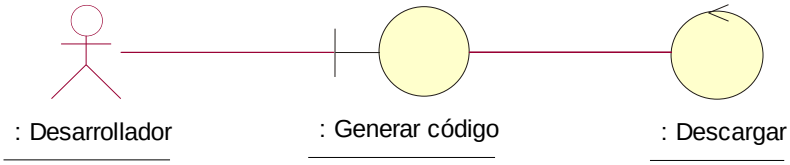


Figura 26. DCU05. Descargar código generado

Fuente. Elaboración propia

- **Descripción de interacciones entre las clases de análisis**

DCU01: Conectar base de datos

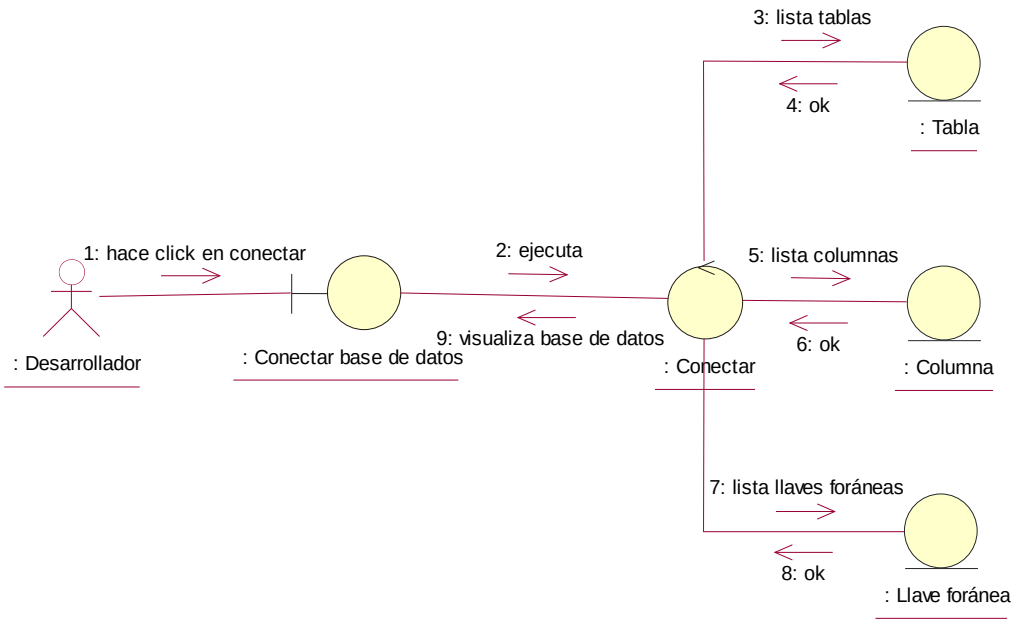


Figura 27. DCU01. Conectar base de datos

Fuente. Elaboración propia

DCU02: Guardar datos

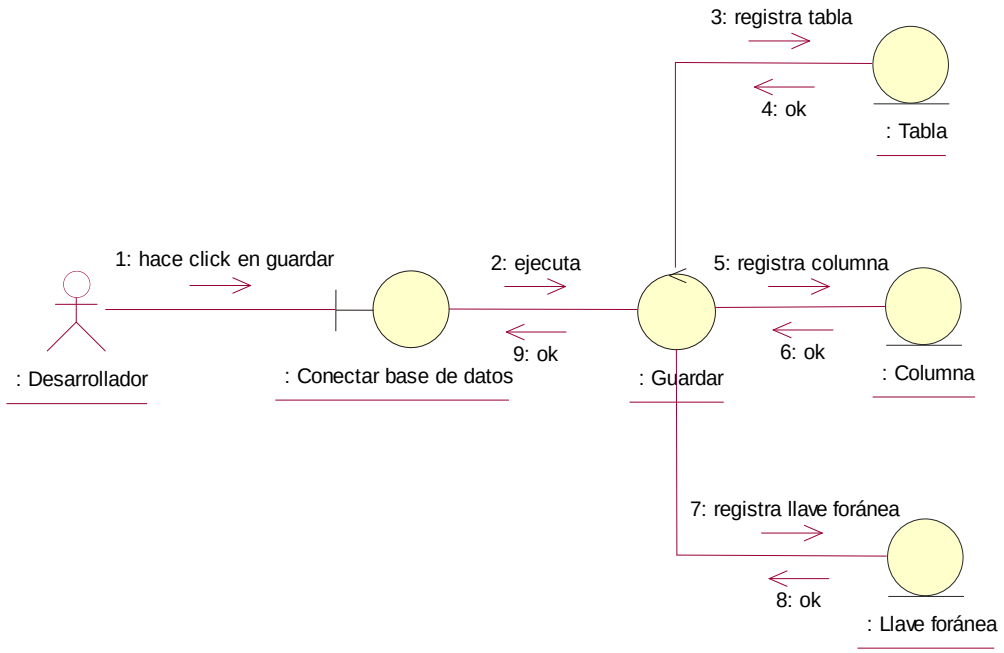


Figura 28. DCU02. Guardar datos

Fuente. Elaboración propia

DCU03: Generar código

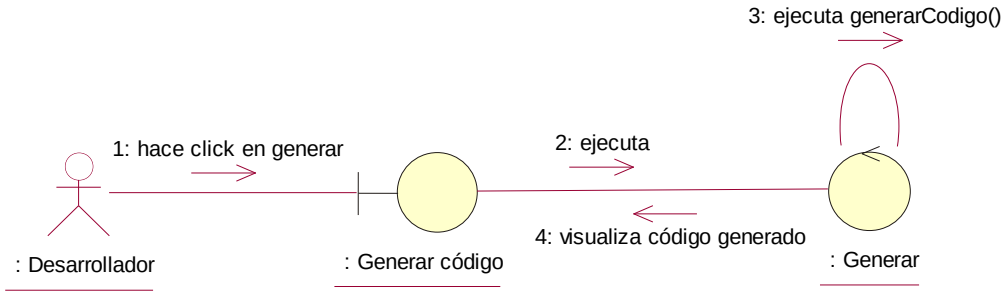


Figura 29. DCU03. Generar código

Fuente. Elaboración propia

DCU04: Descargar proyecto

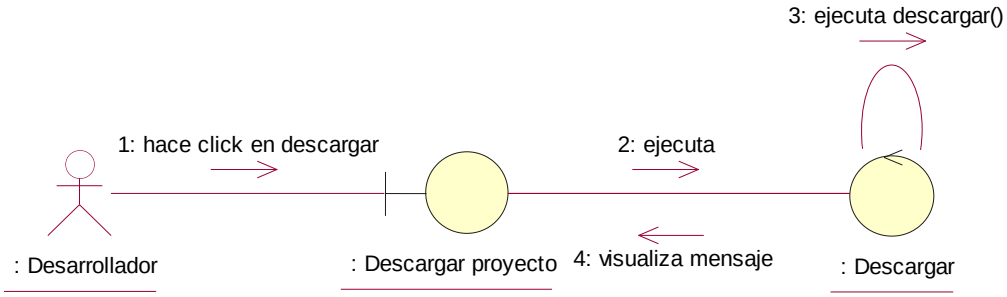


Figura 30. DCU04. Descargar proyecto

Fuente. Elaboración propia

DCU05: Descargar código generado

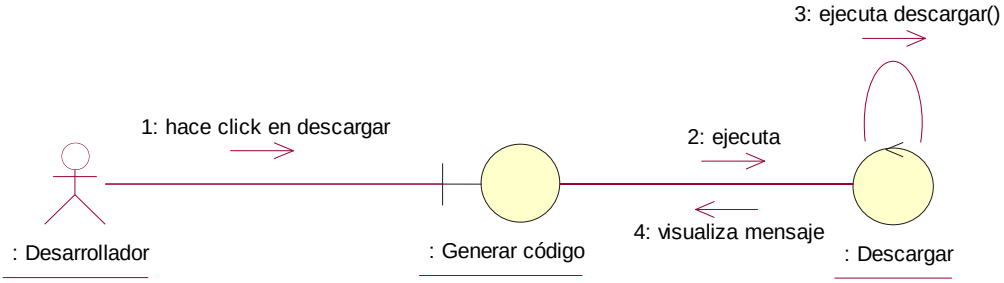


Figura 31. DCU05. Descargar código generado

Fuente. Elaboración propia

- **Identificación de Atributos y Responsabilidades**

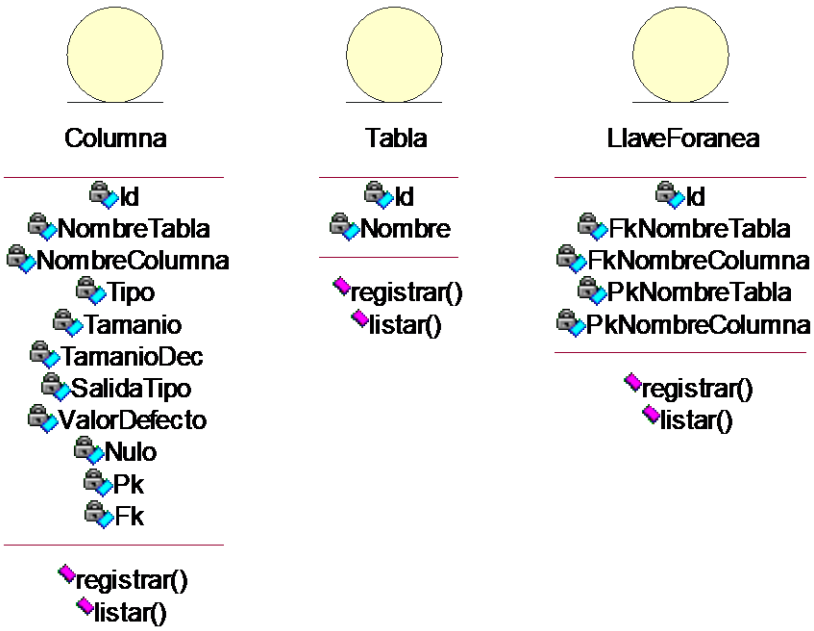


Figura 32. Identificación de Atributos y Responsabilidades

Fuente. Elaboración propia

- **Diagrama de clases (asociaciones, agregaciones, generalizaciones)**

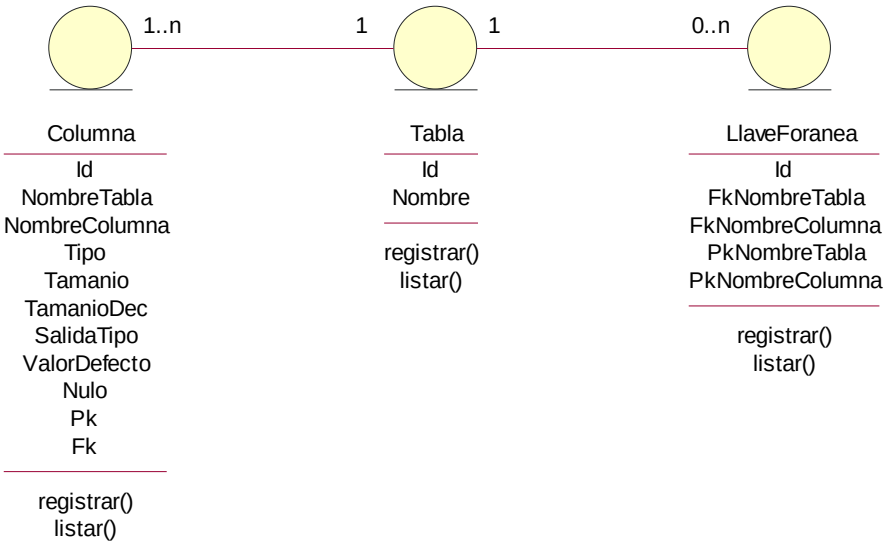


Figura 33. Diagrama de clases (asociaciones, agregaciones, generalizaciones)

Fuente. Elaboración propia

- **Diseño de Interfaces**

  DI01 Conectar base de datos

Figura 34. DI01. Conectar base de datos

Fuente. Elaboración propia

DI02 Guardar datos

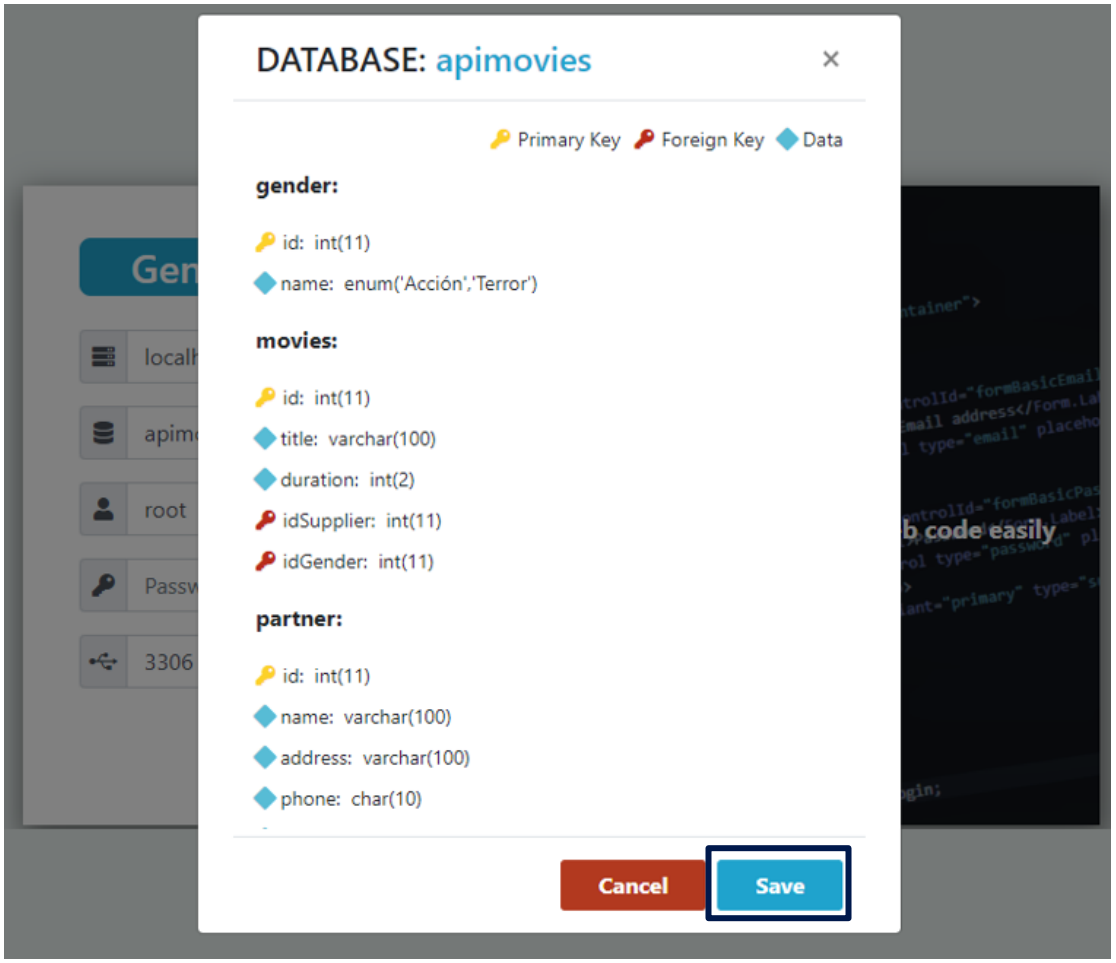


Figura 35. DI02. Guardar datos

Fuente. Elaboración propia

DI Página de inicio

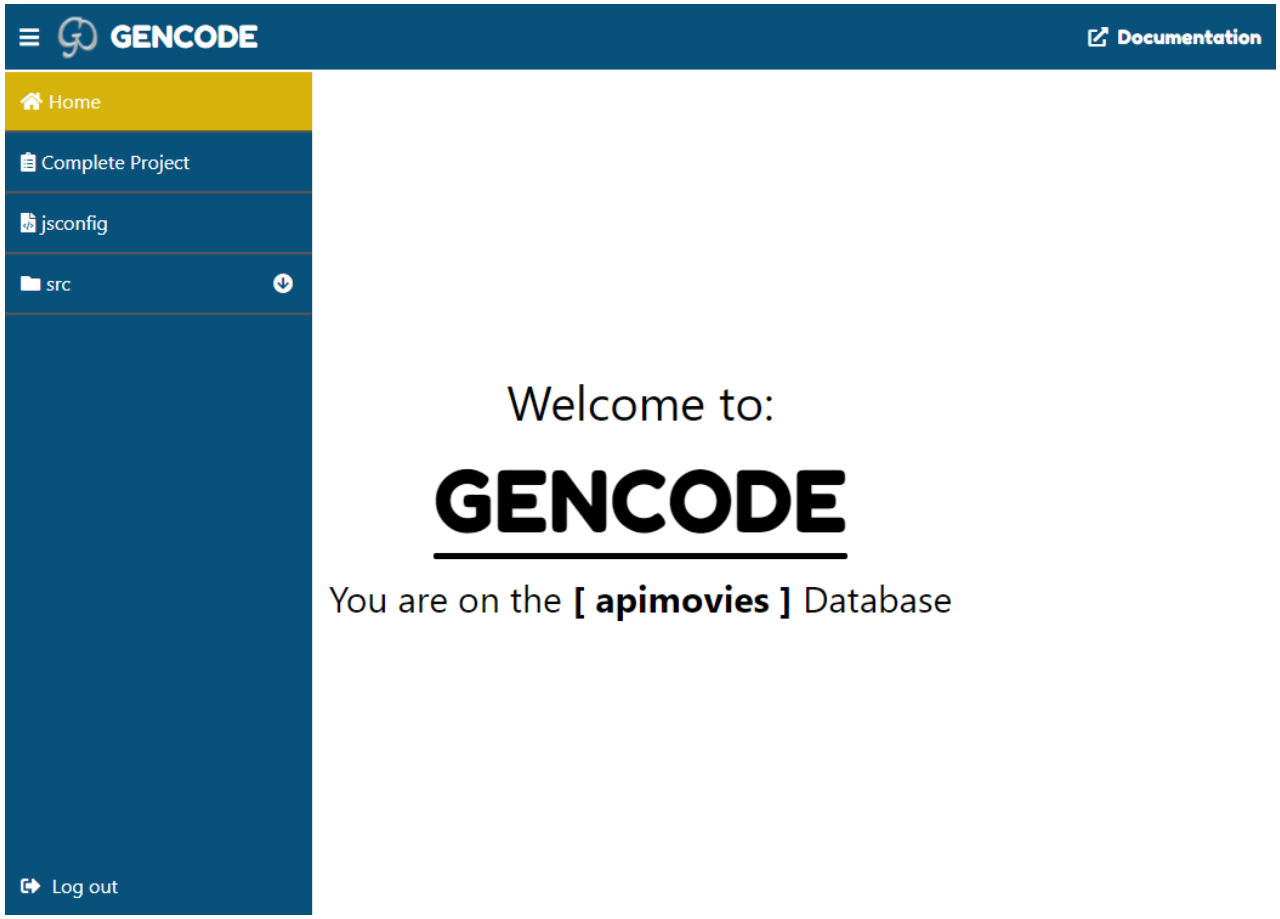


Figura 36. DI. Página de inicio

Fuente. Elaboración propia

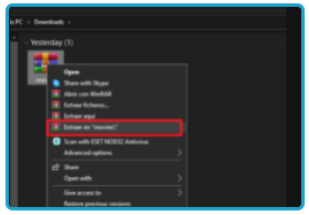
DI Documentación

GENCODE

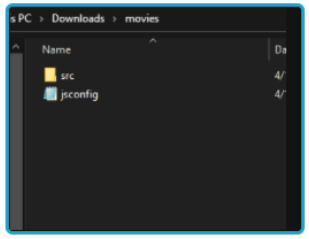
It is a code generator tool to facilitate the creation of web CRUD interfaces.

Steps to follow:

1. Download the complete project from the generator and unzip it.



2. The download contains a "src" folder and a "jsconfig.js" file.



3. In a terminal, run the `npx create-react-app` command to create a React application.

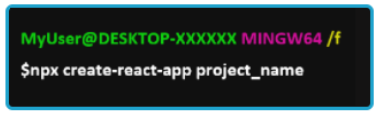


Figura 37. DI. Documentación

Fuente. Elaboración propia

DI03 Generar código

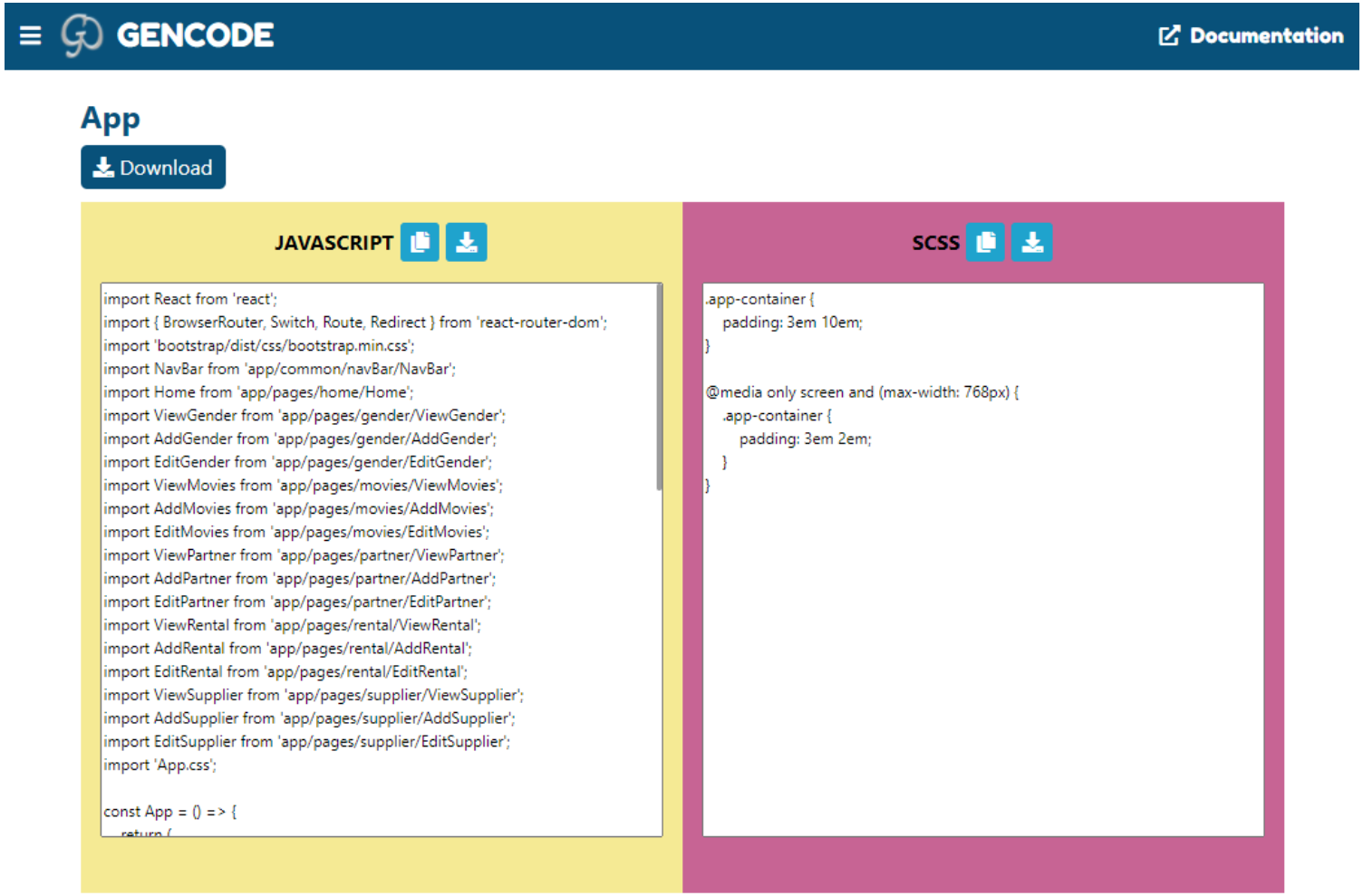


Figura 38. DI03. Generar código

Fuente. Elaboración propia

DI04 Descargar proyecto

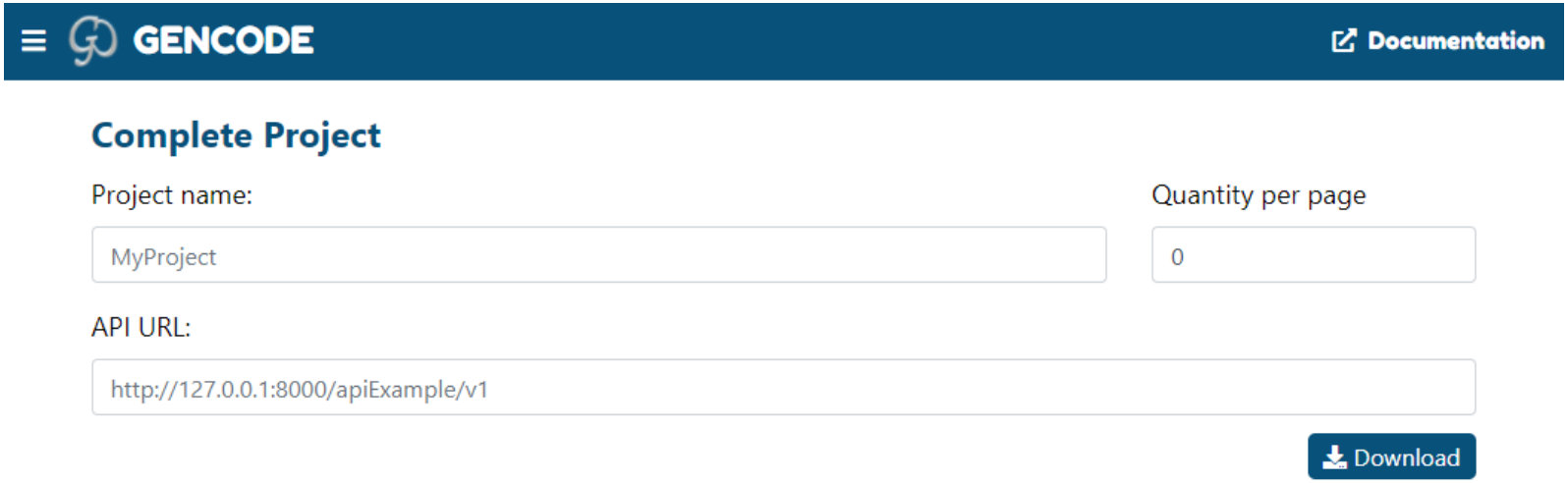


Figura 39. DI04. Descargar proyecto

Fuente. Elaboración propia

DI05 Descargar código generado

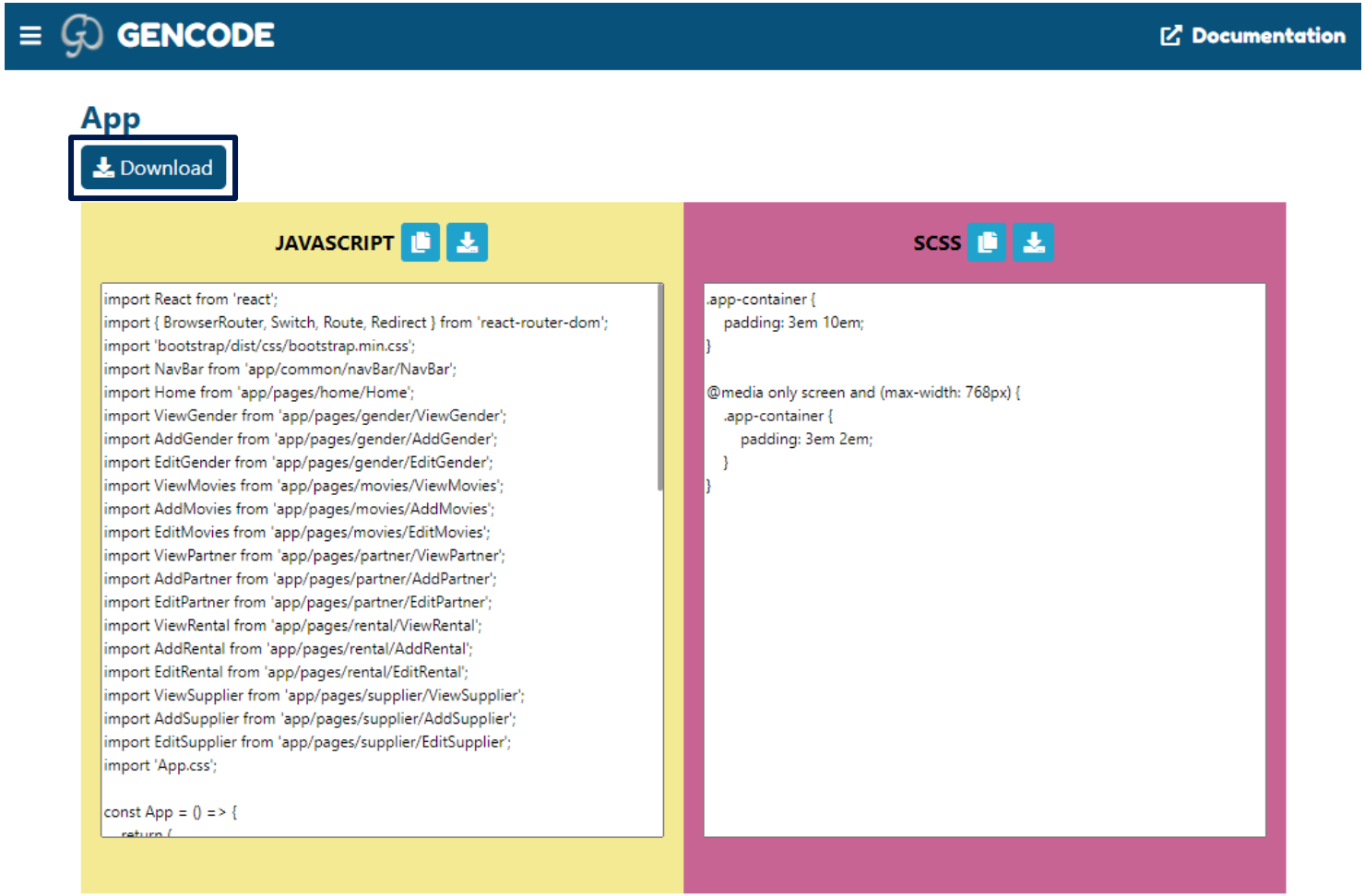


Figura 40. DI05 Descargar código generado

Fuente. Elaboración propia

- **Descripción de los CU de Realización de Diseño**

DRCUD01: Conectar base de datos

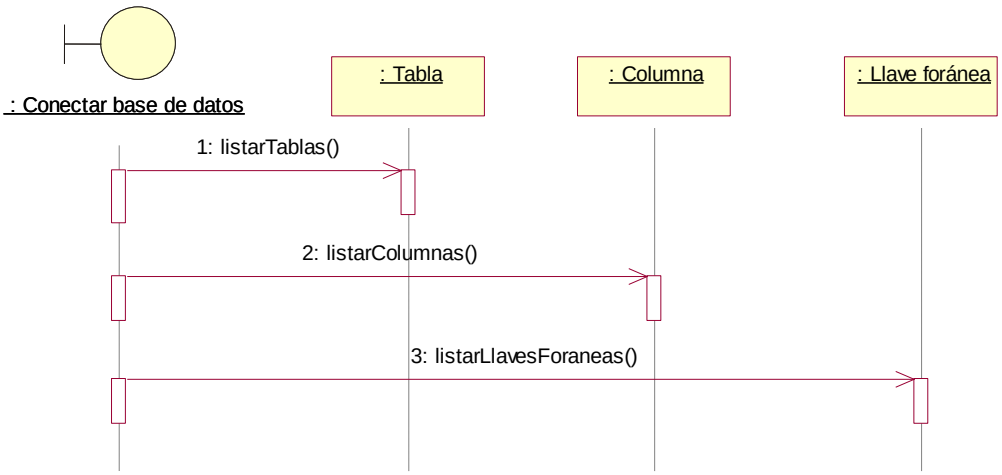


Figura 41. DRCUD01. Conectar base de datos

Fuente. Elaboración propia

DRCUD02: Guardar datos

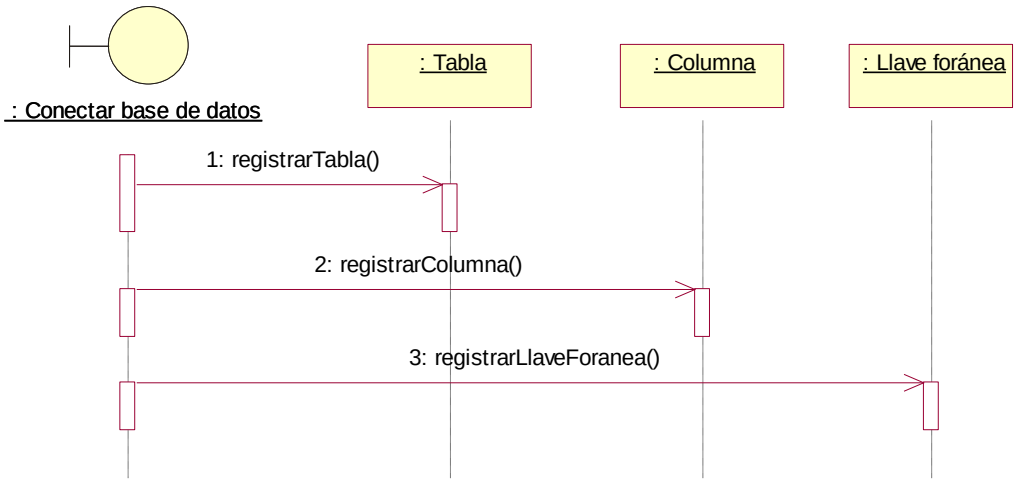


Figura 42. DRCUD02. Guardar datos

Fuente. Elaboración propia

- **Diagrama de Clase General**

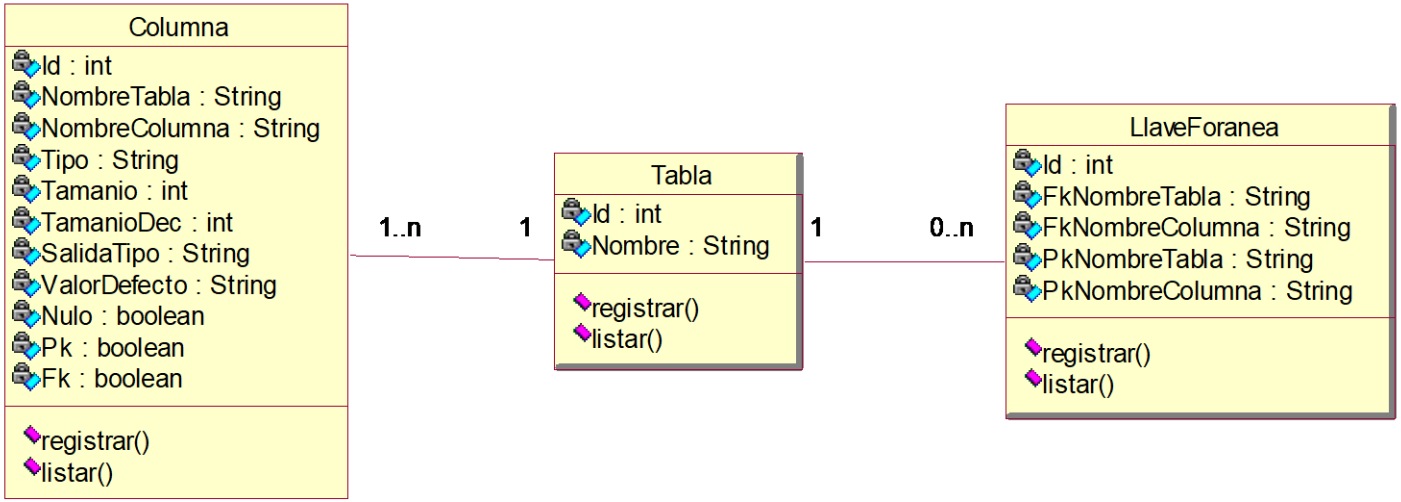


Figura 43. Diagrama de Clase General

Fuente. Elaboración propia

- **Diagrama de Navegabilidad**

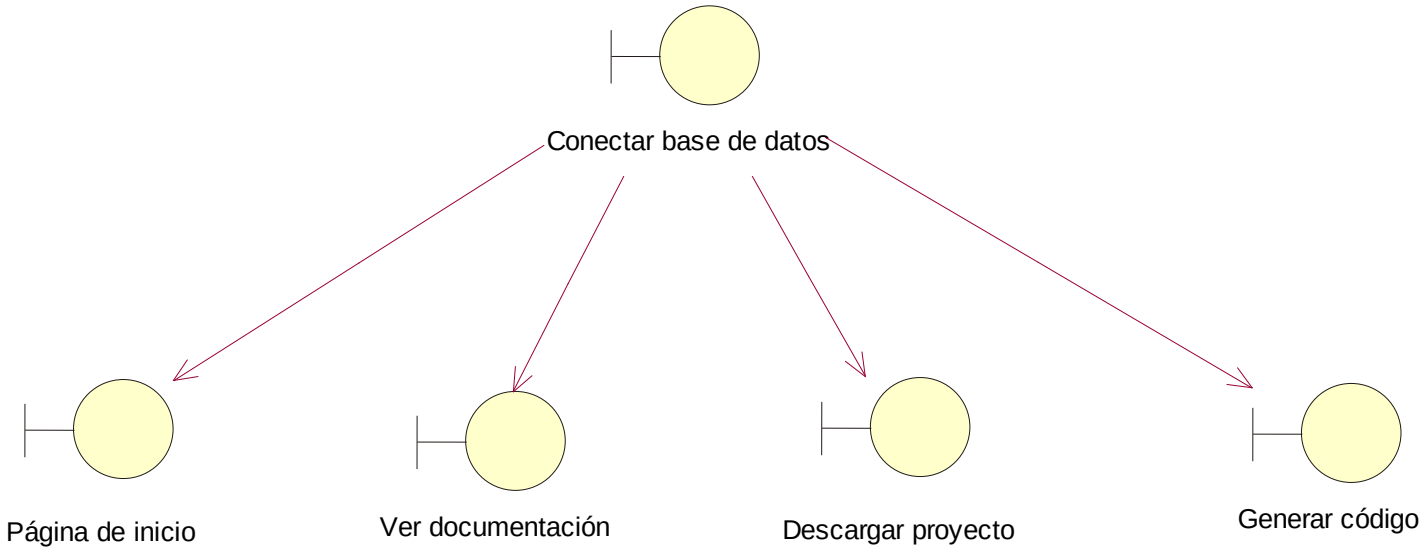


Figura 44. Diagrama de navegabilidad

Fuente. Elaboración propia

- **Diagrama de Implementación**

Diagrama de Componentes

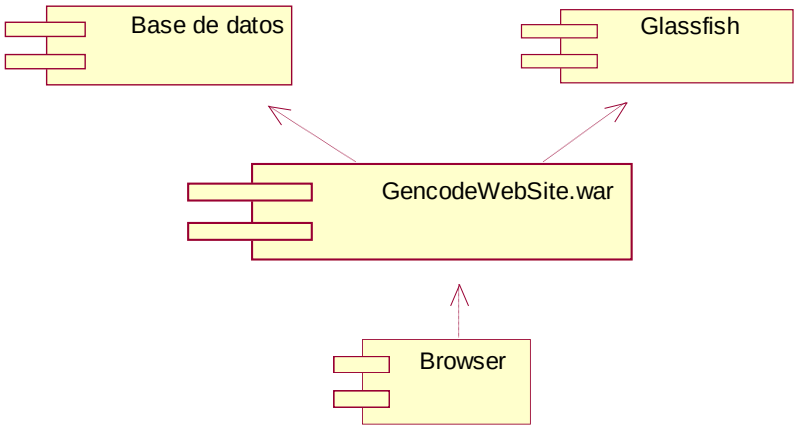


Figura 45. Diagrama de componentes

Fuente. Elaboración propia

Diagrama de despliegue

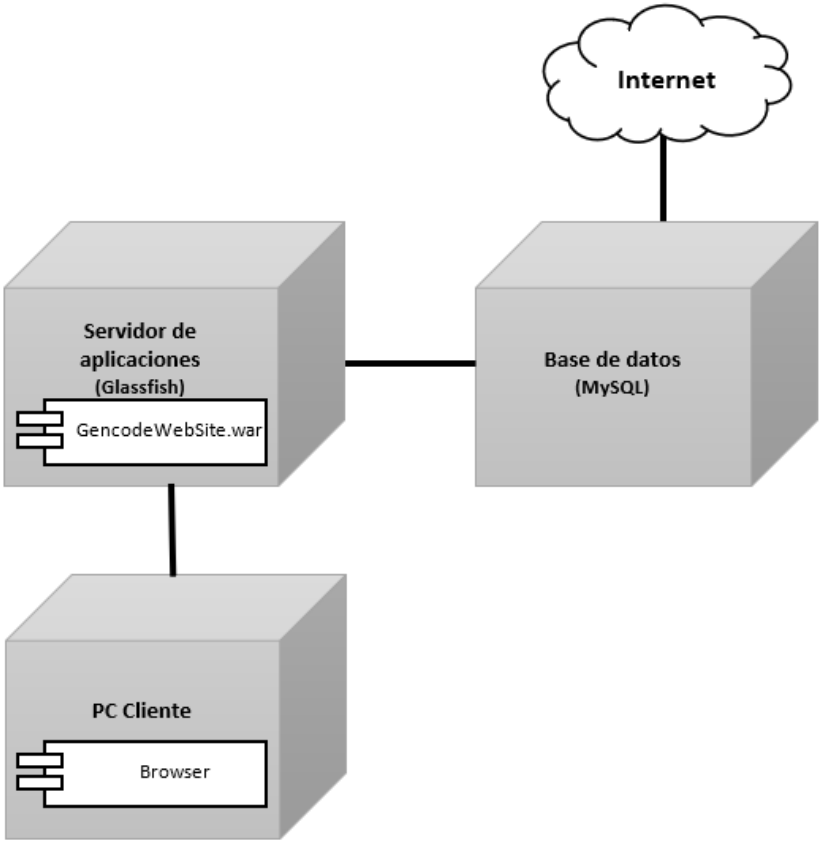


Figura 46. Diagrama de despliegue

Fuente. Elaboración propia

3.2. Desarrollo del código generado

React.js es una librería creada por Facebook que con el paso del tiempo han ido apareciendo versiones donde mejoraban sus características para programar. A partir de la versión 16.8, contamos con los Hooks, el código generado se basa en esta característica de React, donde por ejemplo aparece un hook llamado useState que nos permite inicializar nuestros estados de manera más sencilla, como se observa en la siguiente imagen.

```
const MyComponent = () =>{
    const [ name, setName ] = useState("My Name");

    return (
        <div>My name is {name}</div>
    );
}
```

Figura 47. Código de React Hooks v.16.8

Fuente. Elaboración propia

Arquitectura basados en componentes

Contar con una arquitectura al momento de desarrollar un proyecto de software es importante, ya que de esta manera tendremos una mejor organización al momento de programar y nos servirá como una gran guía para el entendimiento del código.

El código generado se basa en la arquitectura en componentes, característica propia de la librería React.js. Esta arquitectura se diferencia de las demás debido a que el proyecto se divide en partes independientes, uno de otros (como si fuera piezas de lego) para hacer más fácil el desarrollo y cuando sea pertinente realizar un cambio, ubicar el componente y hacer la modificación sin afectar todo el proyecto.

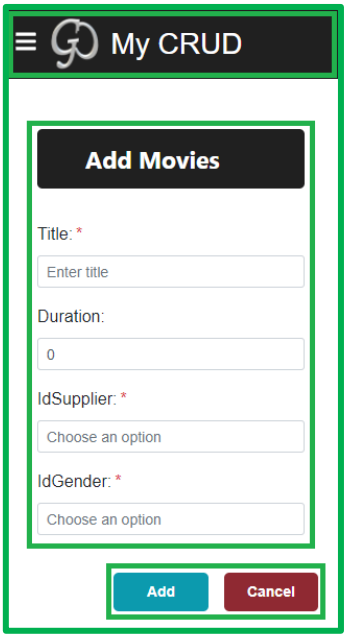


Figura 48. Ejemplo de arquitectura basada en componentes

Fuente. Elaboración propia

Desarrollo declarativo vs imperativo

Cuando uno empieza a desarrollar un proyecto, al escoger un lenguaje, un framework o una librería, nos encontraremos con estilos de programación, entre ellos el desarrollo declarativo e imperativo. El código imperativo, indica en detalle lo que se va a desarrollar ocasionando que el proyecto sea poco mantenible a lo largo del tiempo. En cambio, con React.js, el desarrollo es más de la forma declarativa, ya que se cuenta con un estado de la aplicación en donde si hay un cambio, este se ve reflejado en distintas partes del proyecto.

Análisis de los estilos de codificación

Para el desarrollo del código generado, se ha indagado los diferentes estilos de programación, basados en React.js, para ello se ha tomado en cuenta el libro "React Design Patterns and Best Practices", donde definen un conjunto de técnicas que se debe tener en cuenta al momento de programar para tener un código limpio, reutilizable y la aplicación a largo plazo sea mantenible (Bertoli, 2017).

1. **Propiedades Múltiples**

   Cuando creamos componentes, nos vemos en la necesidad de agregar propiedades múltiples, para poder pasar valores y poder interactuar con ellos, el libro nos plantea 2 estilos las cuales son:

   Estilo 1: Colocar todos los atributos en una sola línea, uno seguido del otro.

   Estilo 2: Colocar cada atributo en una nueva línea y luego alinear la etiqueta de cierre con la etiqueta de apertura.

```
<button
  foo="bar"
  veryLongPropertyName="baz"
  onSomething={this.handleSomething}
/>
```

Figura 49. Propiedades múltiples

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Ventajas y desventajas

Un problema común sucede al escoger el estilo número 1, ya que, si colocamos todas las propiedades en una sola línea, nos arriesgamos a que esta línea de código se extienda demasiado y nuestro código se vuelva ilegible, en cambio con el estilo número 2, tendríamos un código más limpio, fácil de leer, y así en un futuro sería más fácil de mantener.

2. Condicionales

Existen muchas formas diferentes de expresar condiciones en JSX, es importante comprender las ventajas y dificultades de cada una de ellas para poder tener un código más legible y mantenible, a continuación, se mencionan algunas:

- **Condicional if**

Por ejemplo, necesitamos un botón para realizar el cierre de sesión, este debe aparecer solo cuando el usuario esté logueado en la aplicación.

Estilo 1: Tenemos el siguiente extracto de código.

```
let button
if (isLoggedIn) {
  button = <LogoutButton />
}
return <div>{button}</div>
```

Figura 50. Condicional if

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Esto cumple nuestro cometido ya que, si la condición es verdadera, la variable button es igual al botón Logout y nos retornará dicho botón.

Estilo 2: También podemos hacerlo en una sola línea, usando el operador AND (&&). Si la condición es verdadera, nos retornará el botón Logout pero si la condiciona es falsa, quedará sin efecto alguno.

```
<div>
  {isLoggedIn && <LoginButton />}
</div>
```

Figura 51. Condicional if usando el operador AND

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Ventajas y desventajas

En el estilo número 1, usamos la condición if, teniendo el código más extenso, en cambio con el estilo 2 se tiene una mayor facilidad de lectura del código.

- **Condicional if – else**

Ahora si queremos mostrar un botón de cierre de sesión si el usuario está logueado y un botón de inicio de sesión cuando no lo esté, tenemos las siguientes formas de codificar:

Estilo 1: Podemos utilizar la clásica condición if … else.

```
let button
if (isLoggedIn) {
  button = <LogoutButton />
} else {
  button = <LoginButton />
}
return <div>{button}</div>
```

Figura 52. Condicional if - else

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Estilo 2: La segunda forma es usar una condición ternario.

```
<div>
  {isLoggedIn ? <LogoutButton /> : <LoginButton />}
</div>
```

Figura 53. Condicional ternario

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Ventajas y desventajas

Con el estilo 1, si tuviéramos más condiciones, se volvería difícil de entender, mientras con el estilo 2, el código es más legible y mantenible en un futuro.

Veamos ahora la mejor solución para cuando las cosas se pongan más complicadas y, por ejemplo, tenemos que comprobar más de una variable para determinar si renderizar un componente o no.

Estilo 1: Poner la condición en una línea.

```
<div>
 {dataIsReady && (isAdmin || userHasPermissions) &&
   <SecretData />
 }
</div>
```

Figura 54. Condicional en una línea

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Estilo 2: Crear una función auxiliar y retornar los valores de la condición.

```
canShowSecretData() {
  const { dataIsReady, isAdmin, userHasPermissions } = this.props
  return dataIsReady && (isAdmin || userHasPermissions)
}

<div>
  {this.canShowSecretData() && <SecretData />}
</div>
```

Figura 55. Función auxiliar

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Ventajas y desventajas

En este caso, está claro que usar el estilo 1 es una buena solución, pero la legibilidad se ve fuertemente afectada. En cambio, con el estilo 2 podemos crear una función auxiliar dentro de nuestro componente y usarla en JSX para verificar la condición.

3. Sub-renderizado

Al desarrollar una aplicación en React, tenemos 2 formas de codificar:

Estilo 1: Poner toda la lógica en el método render.

Estilo 2: Dividir el método render en funciones más pequeñas de una manera que nos permita mantener toda la lógica en el mismo componente.

```
renderUserMenu() {
  // JSX for user menu
}

renderAdminMenu() {
  // JSX for admin menu
}

render() {
  return (
    <div>
      <h1>Welcome back!</h1>
      {this.userExists && this.renderUserMenu()}
      {this.userIsAdmin && this.renderAdminMenu()}
    </div>
  )
}
```

Figura 56. Sub-renderizado

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Ventajas y desventajas

Con el estilo 1, presentaríamos inconveniencias conforme la aplicación vaya creciendo, el código se volvería ilegible y difícil de mantener mientras con el estilo 2 mantendríamos nuestros componentes muy pequeños, más limpios y fácil de entender.

4. Bucles

Cuando queremos mostrar información en un listado, tenemos distintas maneras de recorrer una lista. A continuación, se describen 2 de ellas:

Estilo 1: Usar forEach.

```
const items=['Item 1','Item 2','Item 3','Item 4','Item 5'];

const itemList=[];

items.forEach((item,index)=>{

   itemList.push( <li key={index}>{item}</li>)

})
```

Figura 57. Función forEach

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Estilo 2: Usar map.

```
<ul>
  {users.map(user =><li>{user.name}</li>)}
</ul>
```

Figura 58. Función Map

Fuente. Bertoli. (2017). React Design Patterns and Best Practices

Ventajas y desventajas

Como se observa, en el estilo 1, claramente se puede aplicar para poder mostrar un listado en nuestra aplicación, pero nos demanda más código, en cambio con el estilo 2, el código es más sencillo, limpio y menos propenso a errores.

Estandarización de proyecto

1. Estructura del proyecto

La estructura del proyecto depende del desarrollador o grupo de desarrollo de cómo dividir la aplicación.

Al crear una aplicación en React.js, nos genera una estructura básica. Para llevar un mejor entendimiento y en el futuro el proyecto sea mantenible se ha dividido de la siguiente forma:

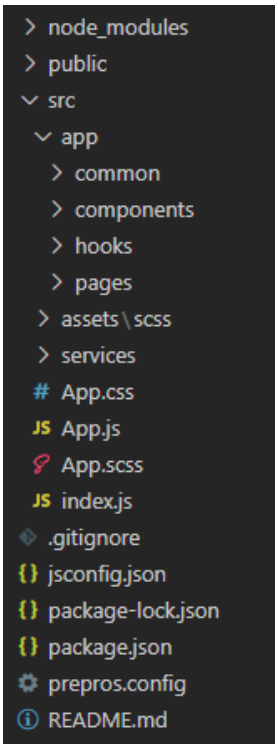


Figura 59. Estructura del proyecto con React.js

Fuente. Elaboración propia

Una carpeta llamada app que contiene toda la aplicación.

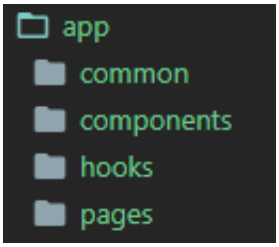


Figura 60. Estructura de la carpeta "app"

Fuente. Elaboración propia

Dentro de ella, se encontrarán 4 subcarpetas:

- **common:** se encuentran los componentes que son comunes en todas la interfaces de usuario, en este caso: alerts(alertas), navBar(Barra de Navegación) y pagination(paginación).

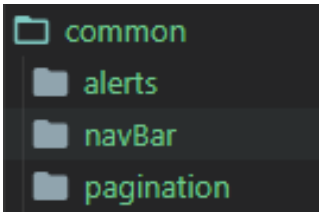


Figura 61. Estructura de la carpeta "common"
Fuente. Elaboración propia

- **components:** encontraremos todos los componentes presentacionales que conforman nuestra aplicación, por ejm: formularios, listados, estilos, aquellos componentes visibles para el usuario.

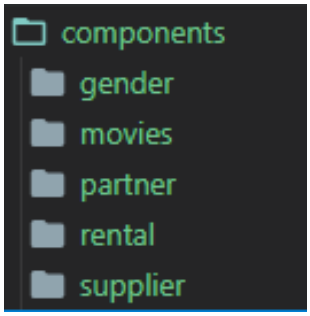


Figura 62. Estructura de la carpeta "components"
Fuente. Elaboración propia

- **hooks:** aquí encontraremos todos los hooks personalizados.

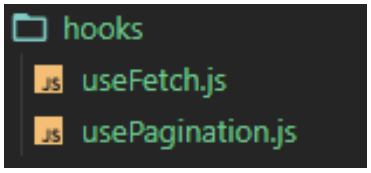


Figura 63. Estructura de la carpeta "hooks"
Fuente. Elaboración propia

- **pages:** contienen la lógica de los componentes presentacionales, por ejm: lógica de formularios, lógica de listados.

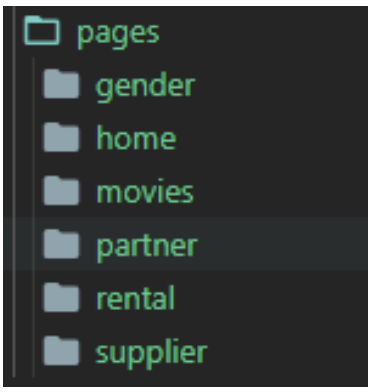


Figura 64. Estructura de la carpeta "pages"

Fuente. Elaboración propia

También tenemos una carpeta llamada assets en donde encontraremos todos los recursos usados en la aplicación, por ejemplos: imágenes, hojas de estilo, etc.

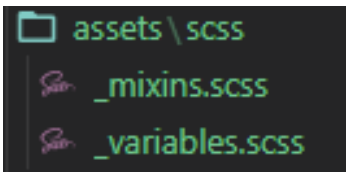


Figura 65. Estructura de la carpeta "assets"

Fuente. Elaboración propia

Finalmente, tenemos la carpeta services, donde están todos los servicios que se necesita para realizar el llamado a la API.

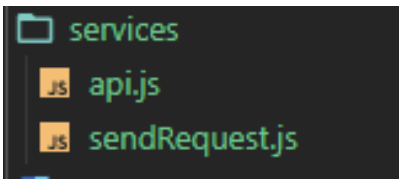


Figura 66. Estructura de la carpeta "services"

Fuente. Elaboración propia

Cabe mencionar que todos los componentes presentacionales cuentan con un archivo JS y SCSS.

2. Estandarización de código

Este proyecto propone estructurar el trabajo de codificación de los desarrolladores, para ello se ha establecido una serie de reglas de código React.js.

Componentes y páginas

- Se ha considerado tres operaciones fundamentales: Registrar, Modificar y Listar. No se recomienda el eliminar, ya que puede causar pérdida de datos de forma definitiva. Teniendo en cuenta estas operaciones se ha definido los siguientes nombres.

  Para los componentes:

  Registrar y editar: <Operación><Nombre de Tabla>Form

  Listar:<Nombre de Tabla>Table

  Donde:

  <Operación>: Determina la operación que se va a realizar. Puede tener los siguientes valores:

  Add: Operación de registrar.

  Edit; Operación de editar.

  <Nombre de Tabla>: Nombre de tabla sobre la cual se realizará la operación.

  Por ejemplo, si se está trabajando con la tabla Empresa los componentes que se crean son los siguientes:

  AddEmpresaForm, EditEmpresaForm y EmpresaTable

Para las páginas:

<Operación><Nombre de Tabla>

Donde:

<Operación>: Determina la operación que se va a realizar. Puede tener los siguientes valores:

Add: Operación de registrar.

Edit; Operación de editar.

View: Operación de listar.

<Nombre de Tabla>: Nombre de tabla sobre la cual se realizará la operación.

Por ejemplo, si se está trabajando con la tabla Empresa las páginas que deberían crearse son las siguientes:

AddEmpresa, EditEmpresa y ViewEmpresa

- Cada página cuenta con variables de estados.

Las variables de estado para almacenar los atributos de cada tabla, tienen la siguiente estructura.

const [<Nombre de Tabla>, set<Nombre de Tabla>] = useState({

<atributo 1>,

<atributo2,

etc

})

```
const [ auto, setAuto ] = useState({
    modelo: '',
    placa: '',
    color: '',
    idMarca: 0
});
```

Figura 67. Estandarización - Hook de estado

Fuente. Elaboración propia

Funciones

Se ha considerado usar funciones anónimas asignándole a una variable (línea 9 y 16), así como para los componentes (líneas 6 y 66), además de poder exportarla al final del fichero (línea 68).

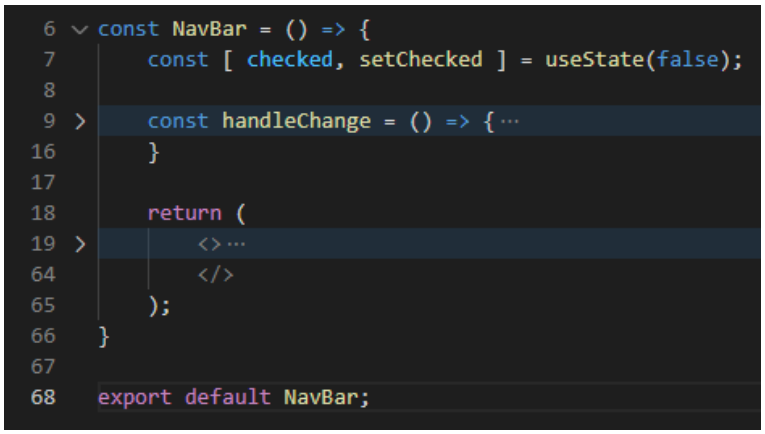


Figura 68. Estandarización - Funciones

Fuente. Elaboración propia

Hooks

Se ha visto conveniente crear dos hooks: useFetch, usePagination, con la finalidad de reutilizar la lógica y tener un código más limpio.

- **useFetch:** Obtiene los datos de la API. Para realizar el llamado a este hook y obtener los valores, se ha definido la siguiente estructura:

  const { data: <Nombre de la tabla>List, loading, error } = useFetch('/<Nombre de la tabla>');

- **usePagination:** Este hook implementa la paginación para filas de tablas en una página. Para llamar a usePagination, se estableció la siguiente estructura:

  const pagination = usePagination(<Nombre la tabla>List);

Rutas

Las rutas tienen la siguiente estructura:

- **Listar:** <Route path="/((Nombre de la tabla))List" exact>
- **Registrar:** <Route path="/new((Nombre de la tabla)) " exact>
- **Modificar:** <Route path="/edit((Nombre de la tabla))/:((Nombre de la llave primaria))" exact>

Propiedades Múltiples

Cuando se trata de crear componentes, y este lleva propiedades, para dar una mejor legibilidad de código, hemos optado por separar las propiedades con un salto de línea, así cuando se va a modificar o pasar más valores, se hace más fácil su entendimiento.

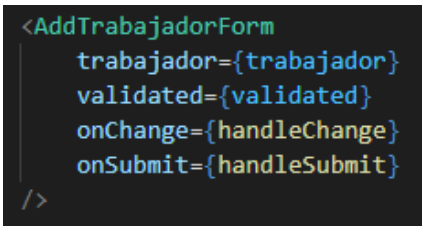


Figura 69. Estandarización - Propiedades Múltiples

Fuente. Elaboración propia

Condicional if

Al momento de realizar una condicional, cuando tengamos solo un valor en la condición para determinar si renderizar o no un componente se ha considerado usar el operador AND, por ejemplo, si hay un error, mostrar un mensaje de alerta.

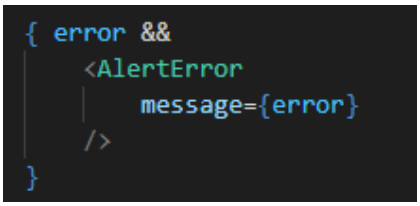


Figura 70. Estandarización - Condicional if

Fuente. Elaboración propia

Cuando tengamos más de dos valores en la condición, se ha optado por crear funciones separadas renderizando el componente que se requiere para no tener una mezcla de código que al final de todo, se complica al leerlo.

```
const backToPage = () => {
    const isBackToPage = pagination.currentPage === pagination.maxPage ||
                         pagination.currentPage > 1;
    if (isBackToPage) {
        return (
            <>
                <Numeration.First onClick={()=>pagination.goJumpPage(1)} />
                <Numeration.Prev onClick={()=>pagination.goNextPrevious()} />
            </>
        );
    }
    return null;
}

const advanceToPage = () => {
    const isAdvanceToPage = pagination.currentPage === 1 ||
                            pagination.currentPage < pagination.maxPage;
    if (isAdvanceToPage) {
        return (
            <>
                <Numeration.Next onClick={()=>pagination.goNextPage()} />
                <Numeration.Last onClick={()=>pagination.goJumpPage(pagination.maxPage)} />
            </>
        );
    }
    return null;
}

return (
    pagination.maxPage > 1 &&
        <Numeration>
            { backToPage() }
            { currentPage() }
            { advanceToPage() }
        </Numeration>
);
```

Figura 71. Estandarización - Función Auxiliar

Fuente. Elaboración propia

Sub-renderizado

Se ha considerado dividir el método render en funciones más pequeñas y así tener el código más legible y fácil de entender.

```
const itemList = () => {···
}

const emptyList = () => {···
}

return (
    <>
        <div className="marcaTable-container">
            <div className="marcaTable-title">Marca List</div>
            <div className="marcaTable-content">
                <div>
                    <Link to="/newMarca" className="marcaTable-btn-add">
                        <FaPlus />Add
                    </Link>
                </div>
                <Table responsive>
                    <thead className="marcaTable-thead">
                        { thead() }
                    </thead>
                    <tbody className="marcaTable-tbody">
                        { itemList() || emptyList() }
                    </tbody>
                </Table>
                <Pagination pagination={pagination} />
            </div>
        </div>
    </>
);
```

Figura 72. Estandarización - Sub-renderizado

Fuente. Elaboración propia

Bucles

Para recorrer un arreglo de objetos se ha utilizado map.

```
return (
    marcaList.map((marca) => (
        <tr key={marca.id}>
            <th className="marcaTable-num">{number++}</th>
            <td>{marca.nombre}</td>
            <td className="marcaTable-action">
                <Link to={"/editMarca/" + marca.id} className="marcaTable-icon-edit">
                    <FaEdit />
                </Link>
            </td>
        </tr>
    ))
);
```

Figura 73. Estandarización - Map

Fuente. Elaboración propia

Nomenclatura

- Para los nombres de constantes y funciones se deben escribir usando el estilo LowerCamelCase.

  Ejemplo: fechaNacimiento, apellidoPaterno, handleSubmit().

- Para los nombres de componentes se usa UpperCamelCase.

  Ejemplo: AddProducto, EditProducto, ViewProducto.

Uso de SCSS

- Para el diseño de las interfaces también hemos usado archivos SCSS, debido a que nos permite manejar el diseño CSS como un lenguaje de programación, se puede crear variables y mixins, donde este último tiene la finalidad de una función.
- Para que el diseño funcione, se debe contar con un preprocesador, en este caso se puede usar PREPOS y así tener como resultado CSS para el diseño de los componentes.

3.3. Pruebas

A fin de contrastar la hipótesis planteada se aplicaron dos pruebas experimentales: GE: conformado por 10 desarrolladores que codificarán la parte front de una aplicación web como primera prueba experimental y como segunda prueba utilizarán la herramienta Gencode.

Primer Experimento

En la tabla se muestra el tiempo requerido por cada desarrollador al implementar interfaces webs que permitan el registro, listado y actualización de datos. Al aplicar la prueba con respecto al indicador Tiempo de codificación, se ha optado separar el desarrollo en dos partes: codificación de tablas sin clave foránea y con claves foráneas.

Tabla 11. Tiempos de codificación por desarrollador – primera prueba experimental

Desarrolladores	Tablas de base de datos					Tiempo Promedio de codificación Tabla sin FK (minutos)	Tiempo Promedio de codificación Tabla con FK (minutos)
	Marca	Auto	Trabajador	Usuario	Tiempo de codificación (minutos)		
Desarrollador 1	27	25	46	57	155	36.50	41.00
Desarrollador 2	25	20	40	52	137	32.50	36.00
Desarrollador 3	30	25	46	52	153	38.00	38.50
Desarrollador 4	27	30	45	50	152	36.00	40.00
Desarrollador 5	20	24	45	53	142	32.50	38.50
Desarrollador 6	26	30	46	48	150	36.00	39.00
Desarrollador 7	30	25	49	48	152	39.50	36.50
Desarrollador 8	25	28	42	46	141	33.50	37.00
Desarrollador 9	24	26	44	51	145	34.00	38.50
Desarrollador 10	27	26	42	53	148	34.50	39.50
	Promedio				147.5	35.30	38.45

Fuente. Elaboración Propia

En la tabla se observa que el tiempo máximo en codificar fue de 155 min y el más corto de 137 minutos.

Segundo Experimento

La segunda prueba experimental consistió en la utilización de la herramienta Gencode con la finalidad de agilizar el proceso de codificación. Cabe mencionar que los integrantes de este grupo fueron capacitados previamente en la herramienta.

Tabla 12. Tiempos de codificación por desarrollador – segunda prueba experimental

Desarrolladores	Tablas de base de datos					Tiempo Promedio de codificación Tabla sin FK (minutos)	Tiempo Promedio de codificación Tabla con FK (minutos)
	Marca	Auto	Trabajador	Usuario	Tiempo de codificación (minutos)		
Desarrollador 1	10	12	11	9	42	10.50	10.50
Desarrollador 2	12	15	13	12	52	12.50	13.50
Desarrollador 3	12	11	11	12	46	11.50	11.50
Desarrollador 4	15	14	14	15	58	14.50	14.50
Desarrollador 5	13	10	12	12	47	12.50	11.00
Desarrollador 6	15	13	11	11	50	13.00	12.00
Desarrollador 7	12	13	12	15	52	12.00	14.00
Desarrollador 8	9	10	10	11	40	9.50	10.50
Desarrollador 9	11	11	11	10	43	11.00	10.50
Desarrollador 10	13	13	10	11	47	11.50	12.00
	Promedio				47.70	11.85	12.00

Fuente. Elaboración Propia

Como se observa en la tabla el tiempo máximo en codificar fue de 58 min y el más corto de 40 minutos.

Tabla 13. Tabla resumen de tiempos promedios de codificación

Resumen	Segunda prueba experimental	Primera prueba experimental	Diferencia	%
Tiempo total promedio (minutos)	47.70	147.5	-99.8	32.34
Tiempo promedio máximo (minutos)	58	155	-97	37.42
Tiempo promedio mínimo (minutos)	40	137	-97	29.20
Tiempo promedio para tablas sin FK (minutos)	11.85	35.3	-23.45	33.57
Tiempo promedio para tablas con FK (minutos)	12	38.45	-26.45	31.21

Fuente. Elaboración Propia

Como se puede observar en la tabla 13, al realizar la primera medición en el desarrollo de una aplicación CRUD con 4 tablas, el tiempo promedio fue de 147.50 min, en comparación con la segunda medición, donde los estudiantes se apoyaron de la herramienta CASE, cuyo tiempo promedio fue de 47.70 min, donde nos demuestra claramente que usar el generador de código es mucho más rápido que construir el sistema desde cero.

La utilización de la herramienta CASE GENCODE nos permite el desarrollo rápido de aplicaciones web (funcionalidad CRUD) basadas en React.js, dejando al desarrollador centrarse en otros requerimientos que conlleven esfuerzo lógico. Cabe mencionar, que el código generado por nuestra herramienta es sencillo de editar y adaptar ya que cuenta con todas las recomendaciones y buenas prácticas indicadas en el punto 3.2.

Capítulo IV. Conclusiones

- Se analizó los estilos de codificación de interfaces en que utilizan React.js, que son empleados por parte de desarrolladores front-end, en base a buenas prácticas.
- Se estandarizó la codificación del diseño de interfaces web usando React.js que facilitó la legibilidad de código y permitió agregar más funcionalidades de manera sencilla.
- Se diseñó la herramienta CASE según la metodología RUP, se capturó los requerimientos para la creación del generador y se realizó un diseño de fácil uso.
- Se codificó la herramienta CASE para la generación de interfaces web, basándonos en el patrón o arquitectura Modelo-Vista-Controlador, haciendo uso de JSF, un framework del lenguaje de programación Java,
- Se probó la herramienta CASE para la verificación de su correcta funcionalidad, además permitió a los desarrolladores construir software en menor tiempo, evitando desarrollar tareas repetitivas.

Capítulo V. Recomendaciones

- Es necesario evolucionar a GENCODE, por lo que se recomienda investigar y analizar nuevos estilos de codificación para la actualización constante del código generado.
- Al momento de añadir nuevas funcionalidades al generador de código, se recomienda utilizar buenas prácticas de codificación, para que sea mantenible a lo largo del tiempo.
- Se recomienda agregar nuevas funcionalidades a GENCODE con el fin de automatizar el desarrollo y concentrarse en nuevos requerimientos que se van presentando.
- En caso de crear una nueva versión se recomienda utilizar las nuevas versiones de la plataforma Java, tanto de Java SE y Java EE.
- Crear nuevas herramientas CASE que ayuden en el desarrollo de software optimizados en los procesos de negocios más demandados del mercado.

Bibliografía referenciada

Ahmed Khan, O. M., & Habib, K. (2020). *Developing Multi-Platform Apps with Visual Studio Code: Get up and running with VS Code by building multi-platform, cloud-native, and microservices-based apps* (First ed.). Birmingham, United Kingdom: Packt Publishing.

Alonso Amo, F., Martínez Normand, L., & Segovia Pérez, J. (2005). *Introducción a la Ingeniería del Software* (Primera ed.). Madrid, España: Delta Publicaciones.

Arizmendi, P. (2018). *AngularJS: Conviértete en el mejor profesional que las compañías de software necesitan* (Primera ed.). Paiminix.

Balaji Varanasi, & Sudha Belida. (2015). *Spring Rest* (First ed.). Apress.

Banks, A., & Porcello, E. (2020). *Learning React: Modern Patterns for Developing React Apps* (Second ed.). O'Reilly Media.

Becerra Urbina, J. (2019). *Generador de código de funcionalidades tipo CRUD en la mantenibilidad de software aplicado a sistemas de información empresariales.* Trujillo. Recuperado el 06 de Mayo de 2021

Bertoli, M. (2017). *React Design Patterns and Best Practices* (Primera ed.). Birmingham, United Kingdom: Packt Publishing.

Bugl, D. (2019). *Learn React Hooks: Build and refactor modern React.js applications using Hooks* (First ed.). Birmingham, United Kingdom: Packt Publishing.

Caballé, S., & Xhafa, F. (2007). *Aplicaciones Distribuidas con Java* (Primera ed.). Delta Publicaciones.

Carrión Bou, R., Noriega, A., & Del Castillo, D. (2019). *Usando XAMPP con Bootstrap y WordPress* (Primera ed.). (M. G. Alcalá, Ed.)

Castledine, E., & Sharkie, C. (2012). *JQuery: Novice to Ninja* (Second ed.). SitePoint.

Ceballos Sierra, J. (2015). *JAVA. Interfaces gráficas y aplicaciones para Internet* (Cuarta ed.). Madrid, España: RA-MA.

Durango, A. (2015). *Diseño Wen con CSS* (Segunda ed.). Createspace Independent Publishing Platform.

Egea García, C. (2008). *Diseño Web para todos I* (Primera ed.). Barcelona, España: Icaria.

Fedosejev, A. (2015). *React.js Essentials* (First ed.). Birmingham, United Kingdom: Packt Publishing.

García Córdoba, F. (2004). *El cuestionario: recomendaciones metodológicas para el diseño de cuestionarios* (Primera ed.). México D.F., México: Editorial Limusa S.A.

García Mariscal, A. B. (2015). *Diseño de bases de datos relacionales* (Quinta ed.). España: Editorial Elearning S.L.

Garza Mercado, A. (2007). *Manual de técnicas de investigación para estudiantes de ciencias sociales y humanidades* (Séptima ed.). (E. C. México, Ed.) México D.F., México.

Gilfillan, I. (2003). *La Biblia de MySQL* (Primera ed.). Anaya Multimedia.

Gómez Salazar, L. E., Gonzáles Téllez, J., & López Gonzáles, E. S. (2004). *Metodología de las Ciencias* (Primera ed.). Jalisco: Umbral.

Groussard, T. (2012). *Java 7. Los fundamentos del lenguaje Java* (Primera ed.). Barcelona, España: ENI EDICIONES.

Hlavats, I. (2009). *Jsf 1.2 Components* (First ed.). Birmingham, United Kingdom: Packt Publishing.

Huamán Valencia, H. (2005). *Manual de Técnicas de Investigación* (Segunda ed.). Lima, Perú: IPLADEES S.A.C.

Huércano Ruíz, F., & Villar Cueli , J. (2015). *UF1290: Implementación e integración de elementos software con tecnologías basadas en componentes* (Primera ed.). IC Editorial.

Izquierdo Herrera, L., Quiñonez Ku, X., & Casierra Cavada, J. (Enero de 2018). Generador automático de aplicaciones web e interfaces de usuarios con funcionalidad responsiva en el lenguaje Python. 6. Recuperado el 12 de Octubre de 2019

Jamsa, K. (2014). *Introduction to Web Development Using HTML 5* (First ed.). Jones & Bartlett Learning.

Jiménez Capel, M. Y. (2014). *Bases de datos relacionales y modelado de datos* (Primera ed.). Málaga, España: IC Editorial.

Kendall, K., & Kendall, J. (2005). *Análisis y diseño de sistemas* (Sexta ed.). México: Pearson.

Kiessling, M. (2011). *The Node Beginner Book* (First ed.). Kindle.

Kruchten, P. (2003). *The Rational Unified Process An Introduction* (Third ed.). United States: Addison-Wesley.

Mamani Rodrigo, W. (2015). *Sistema Generador de Aplicaciones Web a partir de Modelos Físicos de Datos.* Juliaca. Recuperado el 12 de Octubre de 2019

Martín Sierra, A. (2011). *Ajax en J2EE* (Segunda ed.). España: RA-MA S.A.

Martínez Unaucho, J. S., & Rodríguez Chalá, J. L. (2014). *Desarrollo de un prototipo de un generador de código para aplicaciones JEE6 para la empresa Clear Minds Consultores CIA LTDA.* Quito. Recuperado el 12 de Octubre de 2019

Microsoft. (2020). Obtenido de Visual Studio Code: https://code.visualstudio.com/docs

Noriega Martínez, R. (2017). *El proceso de desarrollo de software* (Segunda ed.). IT Campus Academy.

Oracle. (2017). Obtenido de Java Platform, Enterprise Edition (Java EE) 8: https://javaee.github.io/tutorial/jsf-configure002.html

Oracle. (2020). Obtenido de MySQL: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

Oracle. (2021). Obtenido de Oracle: https://www.oracle.com/technical-resources/articles/java/facelets.html

Peña, C. (2019). *Programador Web Full Stack - Desarrollo fontend y backend: Ecosistema Web* (Primera ed.). Buenos Aires, Argentina: Creative Andina Corp.

Pérez Martínez, E. (2015). *Desarrollo de aplicaciones mediante framework de spring* (Primera ed.). Madrid, España: RA-MA Editorial.

Pollock, J. (2014). *jQuery: A Beginner's Guide* (First ed.). McGraw-Hill Education.

Puciarelli, L. (2020). *Node JS - Vol. 1: Instalación - Arquitectura - node y npm* (Primera ed.). Buenos Aires, Argentina: Creative Andina Corp.

Quatrani, T. (2003). *Visual Modeling with Rational Rose 2002 and UML* (Third ed.). Boston, United States: Pearson Education.

Rajput, D. (2019). *Designing Applications with Spring Boot 2.2 and React JS* (First ed.). India: BPB Publications.

Saleh, H., Christensen, A., & Wadia, Z. (2014). *Pro JSF and HTML5: Building Rich Internet Components* (Second ed.). New York, United States: Apress.

Sánchez Maza, M. Á. (2001). *JavaScript* (Primera ed.). Málaga, España: INNOVACIÓN Y CUALIFICACIÓN S.L.

Sass. (2021). Obtenido de Sass: https://sass-lang.com/

Sayago Heredia, J. P. (2018). Generador de Código Utilizando el Paradigma de Líneas de Producto Software. *Hallazgos21*, 23. Recuperado el 12 de Octubre de 2019

Singh, H., & Bhatt, M. (2016). *Learning Web Development with React and Bootstrap* (First ed.). Birmingham, United Kingdom: Packt Publishing.

Sommerville, I. (2005). *Ingeniería del software* (Séptima ed.). Madrid, España: Pearson Educación.

The Apache Software Foundation. (2020). Obtenido de Apache NetBeans: https://netbeans.apache.org/

Torres Remon, M. A. (2012). *Programación orientada a objetos con Visual Basic 2012* (Primera ed.). Lima, Perú: Macro E.I.R.L.

Trottier, A. (2003). *Java 2 Developer, Volume 2* (First ed.). United States: Que Publishing.

Turatti, M., & Pillitu, M. (2013). *Instant Apache Maven Starter* (First ed.). Birmingham, United Kingdom: Packt Publishing.

Vipul , A. M., & Prathamesh, S. (2016). *ReactJS by Example - Building Modern Web Applications with React* (First ed.). Packt Publishing.

Wanyoike, M., M., M., Franklin, J., Teller, S., & Bouchefra, A. (2017). *React: Tools & Resources* (First ed.). Collingwood, Australia: SitePoint.

Zambon, G., & Sekler, M. (2007). *Beginning JSP, JSF, and Tomcat Web Development* (First ed.). New York, United States: Apress.

ANEXOS

ANEXO 1: Glosario de Términos

- **Librería**

  Es un proyecto con métodos o funciones puntuales en el cual se puede anexar a otros proyectos y complementarlo usando métodos específicos para una determinada solución.

- **Interfaz Gráfica**

  Conocida también como GUI (Graphical User Interface), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

- **Sistema Empresarial**

  Es un sistema que ofrece alta calidad de servicio, lidia con grandes volúmenes de datos, capaz de soportar cualquier organización grande.

- **GENCODE**

  Generador de código para la creación de interfaces web.

ANEXO 2: Encuesta para identificar los problemas existentes en el desarrollo de software

1. ¿Cuánto tiempo emplea usted en desarrollar las funcionalidades básicas de un sistema?
    a) Menos de 90 min
    b) 90 min – 120 min
    c) 120 min – 150 min
    d) 150 min – 180 min
    e) 180 min a más

2. ¿Usted considera que el tiempo es un factor importante en el desarrollo del software?
    a) Sin Importancia
    b) De Poca Importancia
    c) Moderadamente Importante
    d) Importante
    e) Muy Importante

3. ¿Ha trabajado con código heredado?
    a) Nunca
    b) Raramente
    c) Ocasionalmente
    d) Frecuentemente
    e) Muy frecuentemente

4. ¿Qué factores afectan la actualización de código fuente? Marcar 1 o más.
    a) Política de negocio desconocida.
    b) Lógica de negocio complicada.
    c) No utilización de modelo de programación por capas.
    d) Falta de estándares de codificación.
    e) Otros: _____

5. ¿Considera usted que la estandarización de código influye positivamente en el mantenimiento de código?

    a) Totalmente en desacuerdo
    b) En desacuerdo
    c) Indeciso
    d) De acuerdo
    e) Totalmente de acuerdo

6. ¿Ha realizado adaptaciones de código existente para la creación de nuevas funcionalidades?

    a) Nunca
    b) Raramente
    c) Ocasionalmente
    d) Frecuentemente
    e) Muy frecuentemente

7. ¿Usted cree que podrá centrarse en otros requerimientos funcionales que necesitan capacidades analíticas si se tiene como base las funcionalidades básicas (Registrar, Modificar, Listar, Eliminar)?

    a) Nunca
    b) Raramente
    c) Ocasionalmente
    d) Frecuentemente
    e) Muy Frecuentemente

8. ¿Usted considera que sea posible generar código automáticamente para el desarrollo eficaz y eficiente de un sistema?

    a) Totalmente en desacuerdo
    b) En desacuerdo
    c) Indeciso
    d) De acuerdo
    e) Totalmente de acuerdo

9. ¿Estaría dispuesto a usar una herramienta CASE que genere las funcionalidades básicas para facilitar el proceso de codificación?

    a) Totalmente en desacuerdo
    b) En desacuerdo
    c) Indeciso
    d) De acuerdo
    e) Totalmente de acuerdo

Tabla 14. Respuestas de las personas encuestadas.

Marca temporal	Pregunta 1	Pregunta 2	Pregunta 3	Pregunta 4	Pregunta 5	Pregunta 6	Pregunta 7	Pregunta 8	Pregunta 9
7/3/2020 22:33:54	180 min a más	Importante	Muy frecuentemente	Política de negocio desconocida., Lógica de negocio complicada., Falta de estándares de codificación.	De acuerdo	Muy frecuentemente	Frecuentemente	Indeciso	De acuerdo
7/3/2020 22:49:58	90 min – 120 min	Muy Importante	Frecuentemente	Falta de estándares de codificación.	Totalmente de acuerdo	Frecuentemente	Ocasionalmente	De acuerdo	Totalmente de acuerdo
7/5/2020 11:53:23	120 min – 150 min	Muy Importante	Ocasionalmente	Falta de estándares de codificación.	De acuerdo	Frecuentemente	Frecuentemente	De acuerdo	De acuerdo
7/5/2020 18:33:51	90 min – 120 min	Moderadamente Importante	Raramente	Lógica de negocio complicada., Falta de estándares de codificación.	Totalmente en desacuerdo	Ocasionalmente	Ocasionalmente	De acuerdo	Totalmente en desacuerdo
7/5/2020 19:34:12	90 min – 120 min	Importante	Ocasionalmente	Lógica de negocio complicada., No utilización de modelo de programación por capas., Falta de estándares de codificación.	De acuerdo	Ocasionalmente	Frecuentemente	De acuerdo	De acuerdo
7/6/2020 13:41:37	180 min a más	Muy Importante	Muy frecuentemente	Lógica de negocio complicada., Falta de estándares de codificación.	De acuerdo	Muy frecuentemente	Frecuentemente	Totalmente de acuerdo	De acuerdo
7/6/2020 18:17:26	150 min – 180 min	Importante	Muy frecuentemente	Lógica de negocio complicada., Falta de estándares de codificación.	Totalmente de acuerdo	Muy frecuentemente	Muy Frecuentemente	Totalmente de acuerdo	En desacuerdo
7/6/2020 19:40:56	150 min – 180 min	Muy Importante	Muy frecuentemente	Falta de estándares de codificación., malas prácticas de código, deuda técnica, sobre ingeniería	De acuerdo	Frecuentemente	Ocasionalmente	De acuerdo	De acuerdo
7/6/2020 19:43:41	180 min a más	Muy Importante	Frecuentemente	Política de negocio desconocida., Lógica de negocio complicada., Falta de estándares de codificación.	Totalmente de acuerdo	Frecuentemente	Muy Frecuentemente	En desacuerdo	En desacuerdo
7/6/2020 22:03:46	180 min a más	Muy Importante	Ocasionalmente	No utilización de modelo de programación por capas., Falta de estándares de codificación.	Totalmente de acuerdo	Ocasionalmente	Ocasionalmente	De acuerdo	Indeciso
7/6/2020 22:14:59	180 min a más	Muy Importante	Muy frecuentemente	Lógica de negocio complicada., Falta de estándares de codificación.	De acuerdo	Muy frecuentemente	Ocasionalmente	Totalmente de acuerdo	Totalmente de acuerdo
7/6/2020 22:33:00	180 min a más	Muy Importante	Muy frecuentemente	No utilización de modelo de programación por capas., Falta de estándares de codificación.	Totalmente de acuerdo	Muy frecuentemente	Ocasionalmente	De acuerdo	Indeciso
7/7/2020 17:04:57	90 min – 120 min	Importante	Raramente	Falta de estándares de codificación.	De acuerdo	Ocasionalmente	Frecuentemente	De acuerdo	Totalmente de acuerdo
7/9/2020 15:55:23	90 min – 120 min	Muy Importante	Frecuentemente	Falta de estándares de codificación.	Totalmente de acuerdo	Muy frecuentemente	Muy Frecuentemente	Indeciso	Indeciso
7/9/2020 16:04:59	120 min – 150 min	Muy Importante	Frecuentemente	Política de negocio desconocida., Lógica de negocio complicada., No utilización de modelo de programación por capas., Falta de estándares de codificación.	Totalmente de acuerdo	Muy frecuentemente	Muy Frecuentemente	De acuerdo	Totalmente de acuerdo
7/9/2020 16:14:28	150 min – 180 min	Muy Importante	Frecuentemente	No utilización de modelo de programación por capas., Falta de estándares de codificación.	De acuerdo	Muy frecuentemente	Ocasionalmente	De acuerdo	De acuerdo
7/9/2020 16:19:01	90 min – 120 min	Moderadamente Importante	Raramente	Política de negocio desconocida., Lógica de negocio complicada.	Totalmente de acuerdo	Raramente	Frecuentemente	Indeciso	Indeciso
7/9/2020 16:55:37	150 min – 180 min	Muy Importante	Frecuentemente	Política de negocio desconocida., Lógica de negocio complicada., Falta de estándares de codificación.	Totalmente de acuerdo	Muy frecuentemente	Frecuentemente	De acuerdo	De acuerdo
7/9/2020 20:57:12	150 min – 180 min	Importante	Frecuentemente	Lógica de negocio complicada., Falta de estándares de codificación.	Totalmente de acuerdo	Muy frecuentemente	Frecuentemente	De acuerdo	De acuerdo
7/9/2020 21:05:30	180 min a más	Importante	Muy frecuentemente	Falta de estándares de codificación.	De acuerdo	Muy frecuentemente	Frecuentemente	De acuerdo	Totalmente de acuerdo
7/10/2020 12:04:17	150 min – 180 min	Importante	Frecuentemente	Falta de estándares de codificación.	De acuerdo	Frecuentemente	Frecuentemente	De acuerdo	De acuerdo
7/10/2020 15:23:02	150 min – 180 min	Muy Importante	Ocasionalmente	Falta de estándares de codificación.	De acuerdo	Frecuentemente	Frecuentemente	De acuerdo	De acuerdo
7/10/2020 16:01:25	150 min – 180 min	Importante	Muy frecuentemente	Lógica de negocio complicada., Falta de estándares de codificación.	De acuerdo	Muy frecuentemente	Ocasionalmente	De acuerdo	De acuerdo
7/10/2020 16:03:39	150 min – 180 min	Muy Importante	Frecuentemente	Política de negocio desconocida., Lógica de negocio complicada., Falta de estándares de codificación.	De acuerdo	Muy frecuentemente	Frecuentemente	De acuerdo	Totalmente de acuerdo

Fuente. Elaboración propia

ANEXO 3: Código fuente

```
public void fn_connect() throws Exception {
    GlobalDAO globalDAO = new GlobalDAO();
    message = "";

    try {
        if(fn_validateForm()){
            if(password == null){
                setPassword("");
            }
            message = globalDAO.fn_validateData(server, port, database, user, password);
            if (message.equals("")) {
                fn_toList();
            }
        }
    } catch (Exception e) {
        message = e.getMessage();
    }
}
```

Figura 74. Código conectar base de datos

Fuente. Elaboración propia

```
public void fn_download() throws IOException {
    try {
        if(fn_validateForm()){
            HttpServletResponse response = (HttpServletResponse) FacesContext.getCurrentInstance()
                    .getExternalContext().getResponse();
            response.setContentType("application/zip");
            response.setHeader("Content-Disposition", "attachment;filename=\"" + projectName + ".zip\"");
            ServletOutputStream sOutStream = response.getOutputStream();
            ZipOutputStream zos = new ZipOutputStream(sOutStream);
            fn_createFiles(zos);
            zos.closeEntry();
            zos.finish();
            sOutStream.close();
            FacesContext.getCurrentInstance().responseComplete();
        }
    } catch (IOException e) {
        throw e;
    }
}
```

Figura 75. Código descargar proyecto

Fuente. Elaboración propia