Serpost
El Correo del Perú

NORMAS PARA EL

DESARROLLO

DE APLICACIONES

SD-N-005.04

I. OBJETIVO

La presente Especificación de Estándares – Desarrollo de Aplicaciones, tiene como objetivo establecer un marco general bajo el cual se pueda crear interfaces, objetos y variables, y nomenclaturas (codificación) de cualquier sistema que deba crearse por solicitud de cualquier área usuaria de la Empresa.

II. ALCANCE

El documento alcanza al Departamento de Sistemas de la Información de la Gerencia de Desarrollo Corporativo y todos los especialistas informáticos que participen directa o indirectamente, a tiempo parcial o completo en el desarrollo y/o mantenimiento de cualquiera de los aplicativos que forman parte de SERPOST S.A.

III. BASE LEGAL

3.1. Decreto Legislativo N° 685 Ley de Creación de SERPOST S.A.

3.2. Estatuto Social de SERPOST S.A.

3.3. Resolución de Contraloría N° 320-2006-CG Normas de Control Interno

3.4. Norma Técnica Peruana "NTP-ISO/IEC 17799:2007 EDI. TECNOLOGÍA DE LA INFORMACIÓN. CÓDIGO DE BUENA PRÁCTICAS PARA LA GESTIÓN DE SEGURIDAD DE LA INFORMACIÓN"

3.5. Normativa interna.

IV. DEFINICIONES Y ABREVIACIONES

Se definen los siguientes términos para mejor interpretación del contenido de este documento:

- **API:** Interfaz de Programación de Aplicaciones.
- **CI/CD:** Integración Continua / Entrega Continua.
- **CRON:** Programador de Tareas.
- **CRUD:** Crear, Leer, Actualizar y Borrar (Create, Read, Update, Delete).
- **CSS:** Hojas de Estilo en Cascada.
- **DB:** Base de Datos.
- **DDoS:** Ataque de Denegación de Servicio Distribuido.
- **DNS:** Sistema de Nombres de Dominio.
- **HTML:** Lenguaje de Marcado de Hipertexto.
- **HTTP:** Protocolo de Transferencia de Hipertexto.
- **HTTPS:** Protocolo de Transferencia de Hipertexto Seguro.
- **IaaS:** Infraestructura como Servicio.
- **IDE:** Entorno de Desarrollo Integrado.
- **JS:** JavaScript.
- **JSON:** Notación de Objetos de JavaScript.
- **JWT:** Token de Seguridad JSON (JSON Web Token).

- **MVC:** Modelo Vista Controlador.
- **ORM:** Mapeo Objeto-Relacional.
- **OWASP:** Proyecto de Seguridad de Aplicaciones Web Abiertas.
- **PaaS:** Plataforma como Servicio.
- **QA:** Aseguramiento de la Calidad.
- **REST:** Transferencia de Estado Representacional.
- **SaaS:** Software como Servicio.
- **SOAP:** Protocolo Simple de Acceso a Objetos.
- **SQL:** Lenguaje de Consulta Estructurada.
- **SSH:** Protocolo Seguro de Shell.
- **UI:** Interfaz de Usuario.
- **UI/UX:** Diseño de Interfaz de Usuario y Experiencia de Usuario.
- **URL:** Localizador Uniforme de Recursos.
- **UX:** Experiencia del Usuario.
- **VCS:** Sistema de Control de Versiones.
- **XML:** Lenguaje de Marcado Extensible.

V. NORMAS

ESTRUCTURA

A. Stack tecnológico
B. Seguridad
C. Lineamientos Frontend
D. Lineamientos Backend
E. Lineamientos Base de Datos
F. Buenas prácticas

A. STACK TECNOLÓGICO

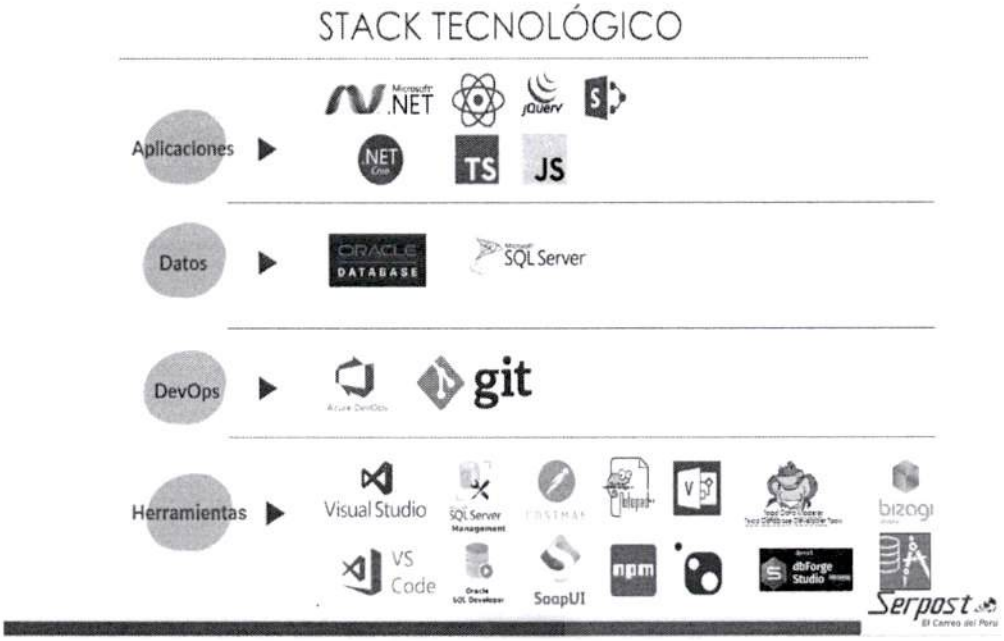
Se refiere a las tecnologías, lenguajes de programación, frameworks y herramientas que se utilizarán en el desarrollo de aplicaciones. Es importante definir un stack coherente y actualizado para garantizar la eficiencia y la calidad del desarrollo. Algunos aspectos a considerar incluyen:

A.1. Lenguaje de programación: Especificar el lenguaje principal a utilizar, como Visual Studio .NET, Java, Python, JavaScript, etc.

A.2. Frameworks y bibliotecas: Enumerar los frameworks y bibliotecas recomendados para agilizar el desarrollo, como React, Angular, Django, Spring, etc.

A.3. Herramientas de desarrollo: Indicar las herramientas de desarrollo, como IDEs (por ejemplo, Visual Studio Code, IntelliJ IDEA) y sistemas de control de versiones (como Git).

A.4. Entornos de ejecución: Definir los entornos de ejecución, como servidores web, bases de datos, contenedores Docker, etc.



B. SEGURIDAD

La seguridad es esencial en el desarrollo de aplicaciones para proteger los datos y la privacidad de los usuarios. Los aspectos de seguridad incluyen:

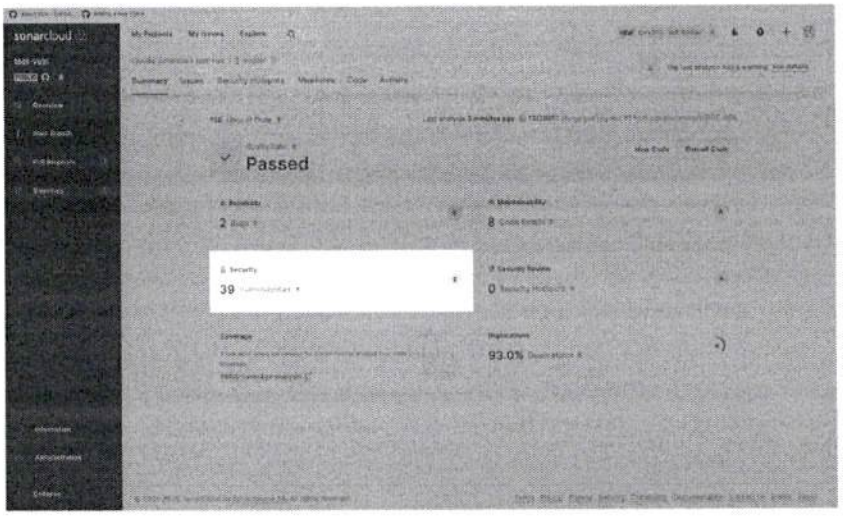
B.1. Autorización para Todo por Defecto: Por defecto, deberá requerirse autorización para acceder los recursos de la aplicación.

En caso de que se decidiera publicar ciertos recursos, debe fundamentarse la decisión en base a reglas de la organización y deberá hacerse un análisis de los riesgos asociados a la decisión.

B.2. Principio de Mínimo Privilegio: Se deberá asignar sólo los permisos necesarios y suficientes a un componente o usuario para realizar una acción específica. Se asignará los permisos lo más tarde posible y se deberá revocar inmediatamente en cuanto no sean necesarios.

B.3. Separación de Responsabilidades: Cuando un componente falle, que no afecte el resto del sistema. Este principio consiste en aislar los privilegios de los componentes para evitar que interfieran entre sí en caso de que alguno sea vulnerable.

USO DE SONAR (CLOUD o QUBE)



Serpost
El Correo del Perú

C. LINEAMIENTOS FRONTEND

Los lineamientos frontend se refieren a las prácticas recomendadas para el desarrollo de la interfaz de usuario de la aplicación.

Consideraciones generales

C.1. Todo desarrollo deberá seguir el stack tecnológico establecido.

C.2. Todo desarrollo deberá considerar la estructura de capas recomendada.

C.3. El código generado deberá ser totalmente estático (CSS, JS, Imágenes, html, etc.).

C.4. Deberá cumplir con las reglas de desarrollo seguro y no tener vulnerabilidades que se encuentren en el TOP10 de OWASP.

C.5. Las fuentes deberán ser versionadas en el repositorio de Azure DevOps de SERPOST S.A., no se realizan despliegues a DEV, QA y PROD de artefactos no versionados y custodiados por SERPOST S.A.

C.6. Diseño responsivo: Asegurar que la interfaz de usuario sea adaptable a diferentes tamaños de pantalla y dispositivos.

C.7. Accesibilidad: Garantizar que la aplicación sea accesible para personas con discapacidades.

C.8. Optimización de rendimiento: Establecer pautas para la optimización de carga de la página y rendimiento del lado del cliente.

C.9. Arquitectura de componentes: Definir una arquitectura de componentes reutilizables y mantenibles.

D. LINEAMIENTOS BACKEND

Los lineamientos backend se refieren a las prácticas recomendadas para el desarrollo de la lógica y la infraestructura de servidor de la aplicación.

D.1. Todo desarrollo deberá seguir el stack tecnológico establecido.

D.2. Todo desarrollo deberá considerar la estructura de capas recomendada.

D.3. Todo desarrollo deberá capturar y escalar las excepciones, desde la específica a la general.

D.4. Considerar la construcción de log detallado para el buen registro y monitorio de eventos.

D.5. La comunicación entre servicios y sistemas deberá ser vía servicios Rest y deben ser API Restfull.

D.6. Todos los servicios Rest deberán contar con su documentación en formato Swagger, cumpliendo los principios de Open API 3.0 (https://swagger.io/specification/ https://www.openapis.org/).

D.7. No se deberá persistir la sesión, ni la validación de token para autenticar un usuario dentro del backend.

D.8. No deberá persistir imágenes, archivos, ficheros planos de más de 5 MB y de una duración mayor de 10 minutos.

D.9. Deberá cumplir con las reglas de desarrollo seguro y no tener vulnerabilidades que se encuentren en el TOP10 de OWASP.

D.10. Las fuentes deberán ser versionadas en el repositorio Azure DevOps de SERPOST, no se realizarán despliegues a DEV, QA y PROD de artefactos no versionados y custodiados por SERPOST S.A.

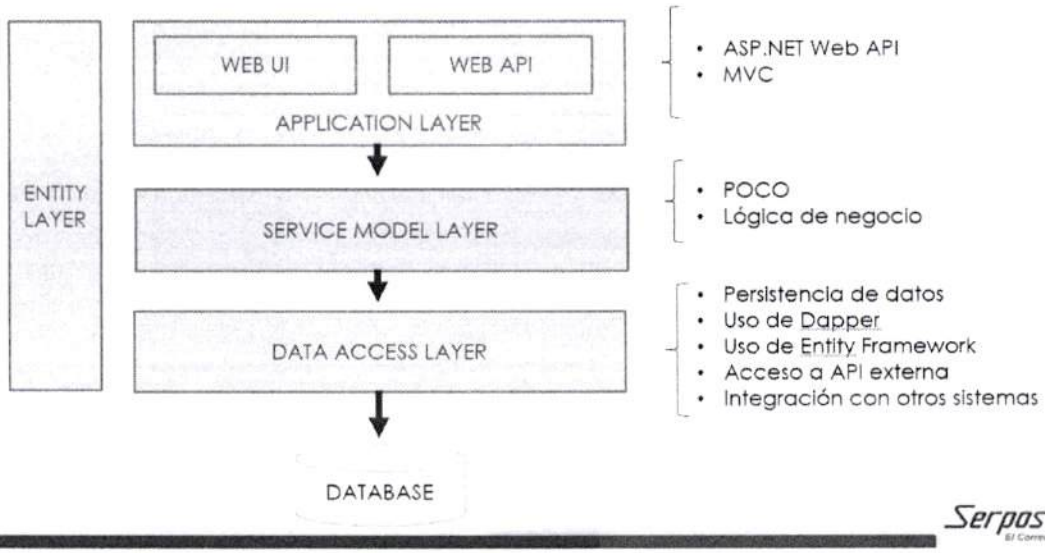
D.11. Deberá cumplir con un code coverage de pruebas unitarias de al menos 80% (Sugerencia SonarQube).

D.12. No se podrá tener deficiencias en la calidad de código Critical, Block, code smell con (Sugerencia SonarQube).

Pueden incluir:

- **APIs y servicios web:** Establecer estándares para la creación de APIs y servicios web consistentes y bien documentados.
- **Escalabilidad:** Definir estrategias para la escalabilidad horizontal y vertical de la infraestructura backend.
- **Seguridad de la API:** Especificar prácticas para asegurar las APIs contra ataques y abusos.
- **Gestión de sesiones y autenticación:** Definir cómo se gestionarán las sesiones y la autenticación en el backend.

ARQUITECTURA BACKEND



Serpost
El Correo del Perú

E. LINEAMIENTOS BASE DE DATOS

Los lineamientos de base de datos se refieren a las mejores prácticas para la gestión de datos.

E.1. Todo desarrollo debe seguir el stack tecnológico establecido.

E.2. Servicios de BD relacionales:

- Oracle
- SQL Server

E.3. No se realizará el pase a los entornos de DEV, QA y PROD si no se tienen los scripts correctamente versionados.

Pueden incluir:

- Modelado de datos: Define cómo se diseñarán y estructurarán las bases de datos.
- Índices y consultas eficientes: Establece pautas para el rendimiento de consultas y optimización de índices.
- Respaldo y recuperación: Describe procedimientos para el respaldo regular de datos y la recuperación en caso de fallos.
- Consistencia de datos: Define estrategias para mantener la integridad y la consistencia de los datos.

F. BUENAS PRÁCTICAS

Las buenas prácticas son fundamentales para mantener la calidad y la coherencia en el desarrollo. Estas incluyen:

F.1. Control de versiones: Establecer la obligación de utilizar control de versiones para rastrear los cambios en el código.

F.2. Documentación: Elaborar una documentación completa y actualizada del código y de la arquitectura de la aplicación.

F.3. Revisión de código: Implementar revisiones regulares de código para identificar problemas y mejoras potenciales.

F.4. Pruebas unitarias y de integración: Elaborar la escritura de pruebas unitarias y de integración para garantizar la funcionalidad y la calidad del código.

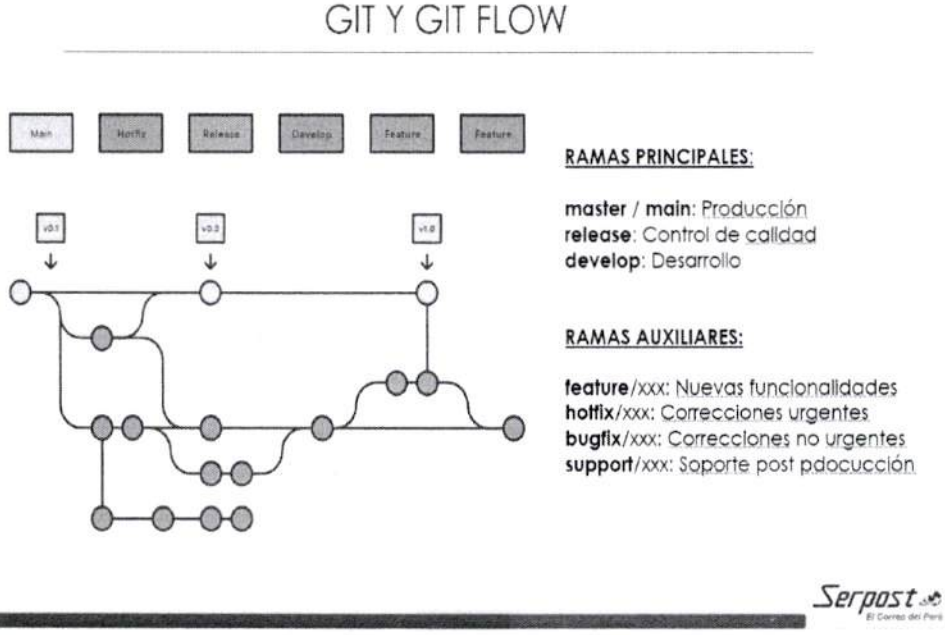
F.5. Ciclo de vida del desarrollo: Definir un ciclo de vida de desarrollo que incluya planificación, diseño, desarrollo, pruebas y despliegue.

GIT Y GIT FLOW

Git: Es un sistema de control de versiones que se utiliza para rastrear cambios en el código fuente de proyectos de software y colaborar en equipos de desarrollo.

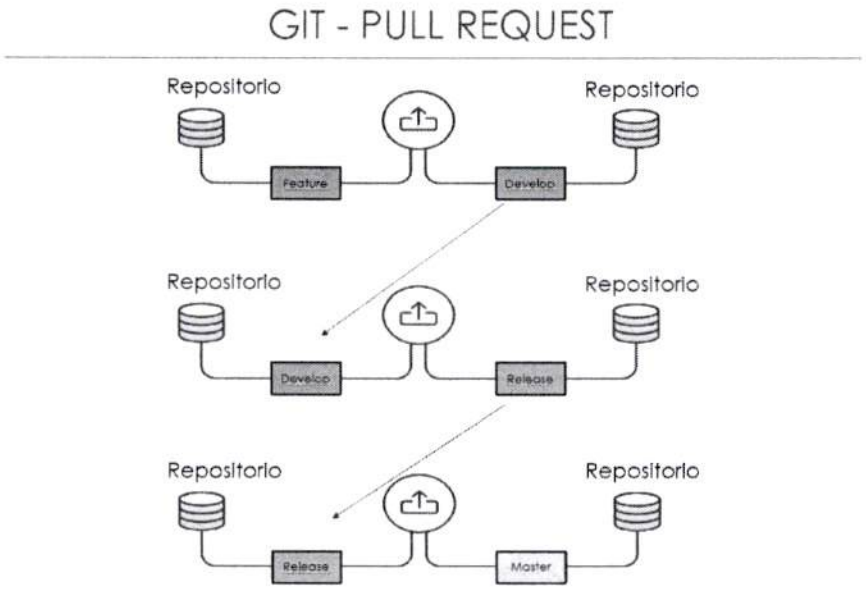
Permite a los desarrolladores llevar un registro de las modificaciones en el código, colaborar con otros programadores de manera efectiva y mantener un historial completo de las revisiones del proyecto. Algunos de los conceptos clave de Git incluyen repositorios, commits, ramas (branches), fusiones (merges), conflictos, entre otros.

Git Flow: Es una convención o conjunto de reglas y prácticas recomendadas para el uso de Git en proyectos de desarrollo de software. Proporciona una estructura y flujo de trabajo para simplificar la gestión de ramas y versiones en un proyecto.



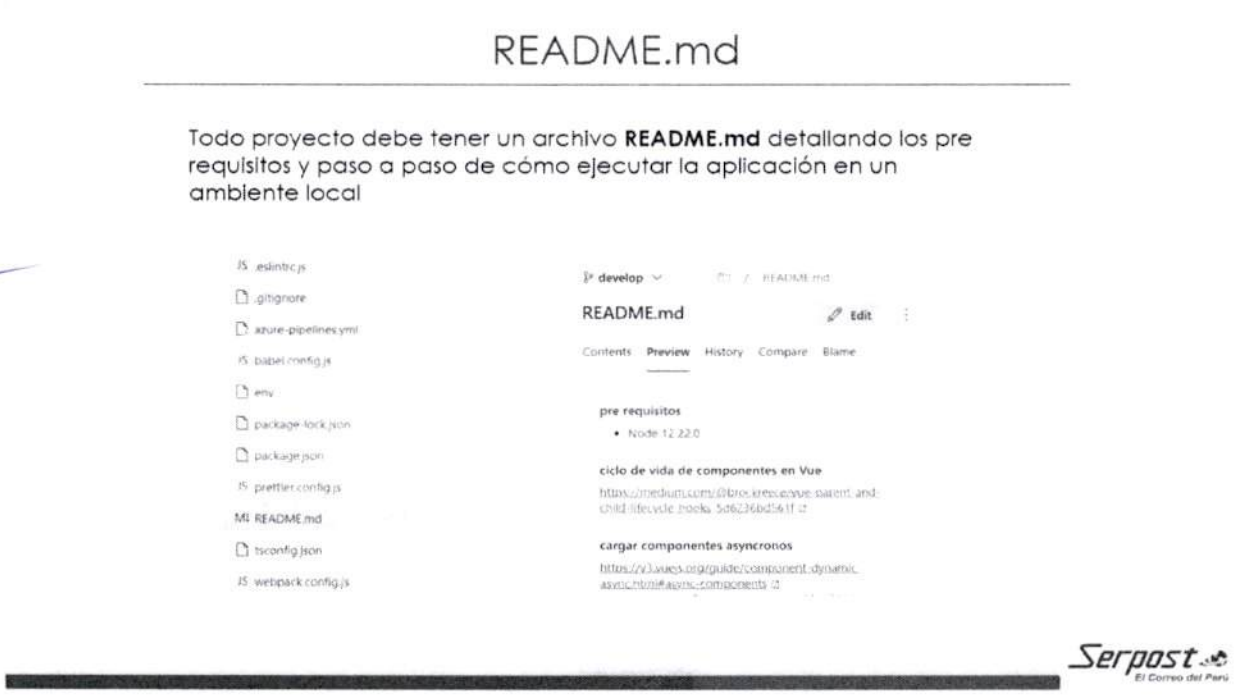
GIT – PULL REQUEST

Es una forma de solicitar que otro desarrollador incorpore los cambios que se ha hecho en una rama del repositorio a su rama principal. Es una manera de colaborar en proyectos de código abierto o grupales, y de recibir feedback sobre tu trabajo. Para hacer un GIT - PULL REQUEST, se necesita tener una copia del repositorio original (haciendo un fork) y una rama local donde se hayan efectuado las modificaciones.



READMY.md

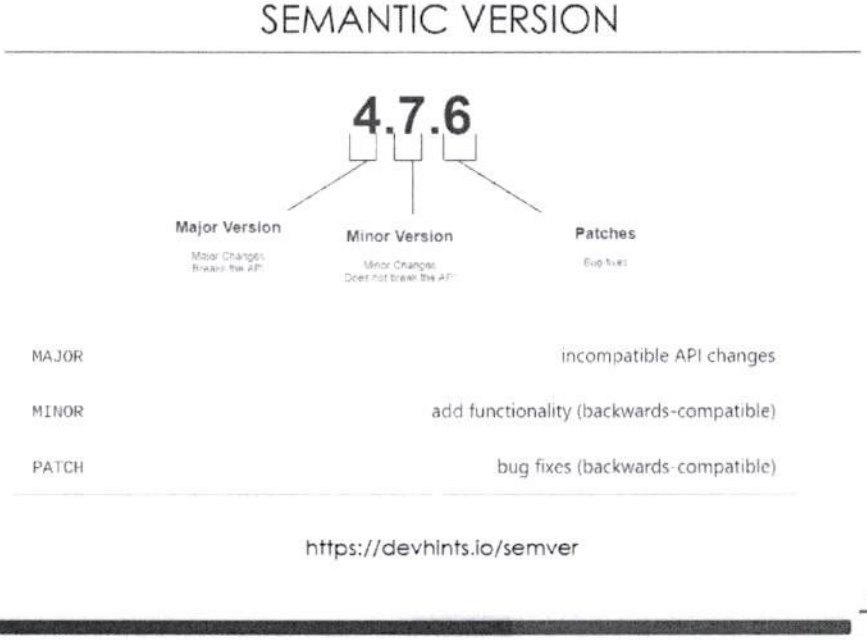
Un archivo README.md es un archivo de texto que contiene información sobre un proyecto, sistema o software. El nombre README es el primer archivo que un usuario o desarrollador debe leer antes de usar o contribuir al proyecto.



SEMANTIC VERSION

El versionado semántico (o SemVer) es un esquema de versiones ampliamente adoptado que codifica una versión mediante un número de versión de tres partes (Mayor.Menor.Parche), una etiqueta opcional de prelanzamiento y una etiqueta opcional de metadatos de compilación.

El versionado semántico se basa en un conjunto de reglas y requerimientos que dictan cómo asignar e incrementar los números de versión, permitiendo comunicar los cambios realizados en el API público de un software con incrementos específicos al número de versión.



Serpost
El Correo del Perú

CONVENTIONAL COMMIT

Es una forma de escribir los mensajes de los commits siguiendo una convención simple que facilita la lectura y el análisis del historial del repositorio. Esta convención se basa en el uso de un tipo, un ámbito opcional, una descripción, un cuerpo opcional y un pie de página opcional.

CONVENTIONAL COMMIT

1. **fix:** un commit de *tipo* `fix` corrige un error en la base del código (se correlaciona con `PATCH` en el Versionado Semántico).
2. **feat:** un commit de *tipo* `feat` introduce una nueva funcionalidad en la base del código (se correlaciona con `MINOR` en el Versionado Semántico).
3. **BREAKING CHANGE:** un commit que contiene la nota al pie `BREAKING CHANGE:` , o que agrega un `!` después del tipo/ámbito, introduce un cambio de ruptura de API (se correlaciona con `MAJOR` en el Versionado Semántico). Un BREAKING CHANGE puede ser parte de commits de cualquier *tipo*.
4. *tipos* distintos a `fix:` y `feat:` están permitidos, por ejemplo @commitlint/config-conventional (basados en la convención de Angular) que recomienda `build:` , `chore:` , `ci:` , `docs:` , `style:` , `refactor:` , `perf:` , `test:` , y otros.
5. *notas al pie* distintas de `BREAKING CHANGE: <descripción>` pueden ser añadidas y siguen una convención similar al formato git trailer.

https://www.conventionalcommits.org/en/v1.0.0/

Serpost
El Correo del Perú

PRINCIPIOS S.O.L.I.D.

S.O.L.I.D. es el acrónimo de los 5 principios para el diseño de un programa orientado a objetos.

En ello mencionan la importancia de que cada clase debe tener una determinada función o trabajo a realizar, evitando sobrecargar cada clase.

Del mismo modo cada objeto debería estar abierto a una extensión pero cerrado a modificaciones, esto lo logramos usando interfaces. Evitando hacer futuros cambios sobre la misma clases y rompiendo posibles funcionalidades ya implementadas.

Finalmente, al momento de proveer dependencias a una clase, la creación de los mismos deben realizarse por un objeto o clase externa, fuera de la clase en donde se requiera las dependencias. Aplicando ello, podemos proveer distintos tipos de dependencias o los mismos(Singleton) según se requiera.

PRINCIPIOS S.O.L.I.D.

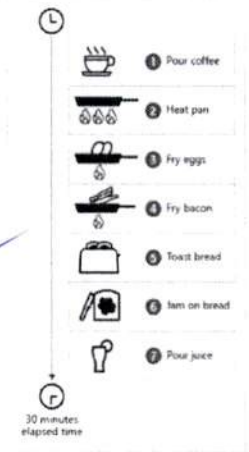
Serpost
El Correo del Perú

PROGRAMACIÓN ASINCRÓNICA

Permite ejecutar varias tareas al mismo tiempo sin bloquear el flujo principal de la aplicación. Esto mejora el rendimiento y la experiencia de usuario, especialmente cuando se trabaja con operaciones que dependen de recursos externos, como la red, el disco duro o la base de datos.

PROGRAMACIÓN ASÍNCRONA

- La programación síncrona bloquea la ejecución del hilo actual
- La programación asíncrona realiza la invocación de métodos y puede continuar con la ejecución de las siguientes instrucciones



Programación síncrona

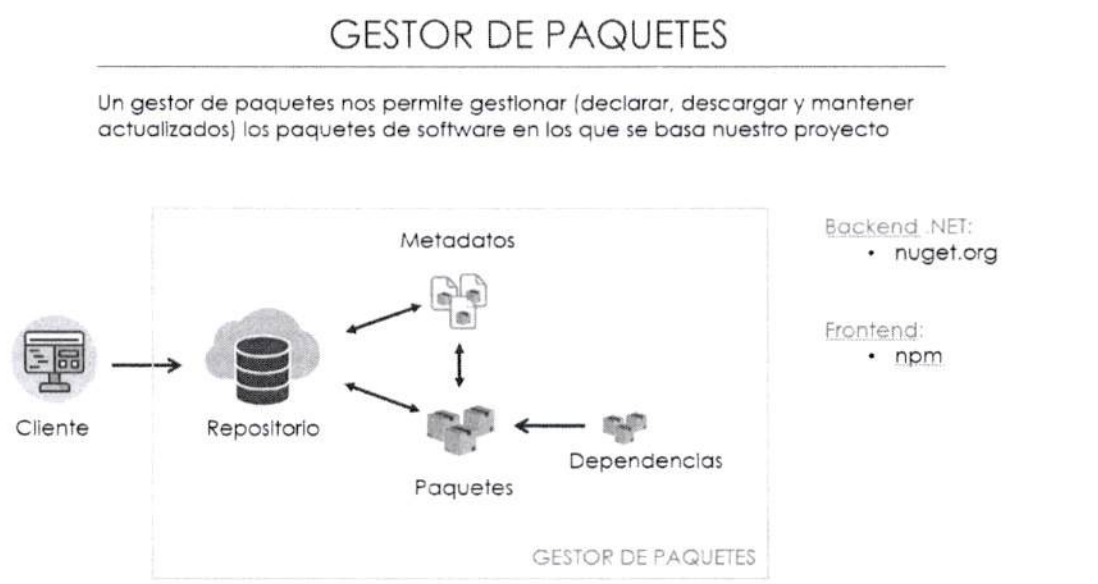
Programación asíncrona con **async** y **await**

Serpost
El Correo del Perú

GESTOR DE PAQUETES

Es una herramienta que ayuda a instalar, actualizar, configurar y eliminar paquetes de software de una manera fácil y automatizada. Un paquete de software es un conjunto de archivos y metadatos que contiene el código fuente, los recursos y la información necesaria para ejecutar una funcionalidad específica en tu ordenador.

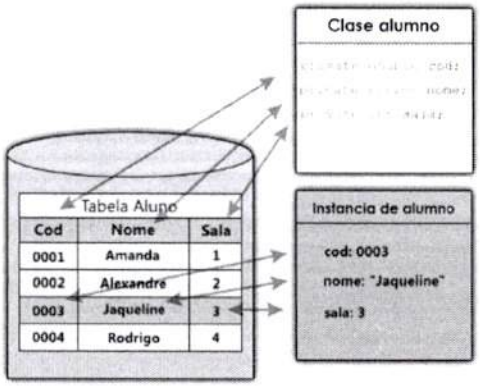
Los gestores de paquetes se encargan de mantener un registro de todos los paquetes instalados en el sistema, y de resolver las posibles incompatibilidades o conflictos entre ellos.



ORM EN CAPA DE DATOS

El mapeo de objeto-relacional (ORM: Object-Relational mapping) , es un modelo de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador.

ORM EN CAPA DE DATOS



Un ORM (Object Relational Mapping) es un mapeo a un objeto relacional en nuestra aplicación. Es una técnica empleada para convertir los datos entre los distintos sistemas de base de datos y los lenguajes de programación.

Dapper:
Micro-ORM, que ofrece los servicios principales de parametrización y materialización.

Lo podemos ver como un conjunto de clases que nos permiten mapear nuestras clases POCO con la base de datos.

CLEAN/HEXAGONAL ARCHITECTURE

Se basa en la idea de aislar la lógica de negocio de las preocupaciones externas, separando la aplicación en componentes poco acoplados e intercambiables, como el núcleo de la aplicación, la base de datos, la interfaz de usuario, los scripts de prueba y las interfaces con otros sistemas.

Se compone de los siguientes elementos:

- **Núcleo de la aplicación:** contiene la lógica de negocio, las entidades y los casos de uso de la aplicación.
- **Puertos:** son las interfaces que expone el núcleo de la aplicación para comunicarse con el mundo exterior.
- **Adaptadores:** son los componentes que se encargan de adaptar los datos y las llamadas entre los puertos y los componentes externos.

Gerente General V°B° Serpost ALEXANDER INFANTES POMAR

Gerente de Desarrollo Corporativo V°B° Serpost CARLOS OBERTI CORONADO

Jefa Subgerente de Tecnologías de la Información V°B° Serpost Ing. SHIRLEY MUÑOZ AZCARRAGA

Jefe del Dpto. Organización y Procesos V°B° Serpost JUAN PABLO FERNANDEZ HART

Departamento de Sistemas de Información V°B° Serpost Sr. JOSÉ PINTO VILLAN

Gerente Legal (e) V°B° Serpost LILIANA SOLDEVILLA NARVAEZ

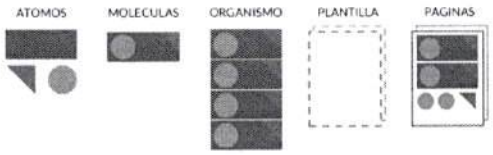
CSS ARCHITECTURE

CSS ARCHITECTURE es el término que se usa para referirse al diseño, la organización y la estructura del código CSS de una aplicación web. La arquitectura CSS tiene como objetivo mejorar la calidad, la legibilidad, la reutilización y el mantenimiento del código CSS, así como facilitar la colaboración entre los desarrolladores.

CSS ARCHITECTURE

Estamos usando Atomic Design como arquitectura CSS.

El diseño atómico nos permite tomar sistemas de diseño complejos y descomponerlos en componentes más pequeños llamados átomos. Entonces, es mucho más fácil para todos los involucrados en el proyecto ver qué partes del sitio se pueden reutilizar.



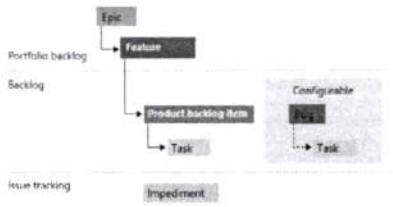
SCRUM: BACKLOG

BACKLOG es una de las herramientas que se emplean en SCRUM para organizar y priorizar el trabajo que se debe realizar en el proyecto. Un BACKLOG es una lista ordenada de los requisitos, características, mejoras y correcciones que se necesitan para mejorar el producto.

SCRUM: BACKLOG

Las **Epics** y **Features** deben tener un enfoque comercial.

Los **Task** deben ir sumando las **Users Story** y éstas se convierten en **Features**, y las **Features** se convierten en **Epics**



Epics:

- Incrementar el compromiso del cliente
- Mejorar y simplificar la experiencia del usuario
- Implementar una nueva arquitectura para mejorar el rendimiento
- Diseñar la aplicación para respaldar el crecimiento futuro
- Apoyar la integración con servicios externos
- Admitir aplicaciones móviles

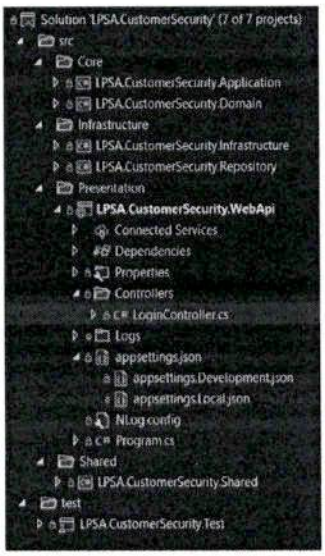
Features:

- Agregar opciones de vista al nuevo centro de trabajo
- Agregar carrito de compras móvil
- Alertas de texto de soporte
- Actualice el portal web con una nueva apariencia

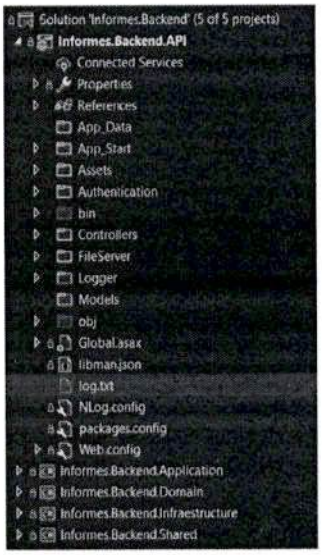
Serpost
El Correo del Perú

ESTRUCTURA.NET

ESTRUCTURA .NET



CQRS



Clean Architecture

N - Tiers

ESTRUCTURA REACT

dist – Resultado de compilación
src – Código fuente
.env – Variable de entorno
readme.md – Requisitos e Instrucciones para ejecutar la aplicación en un nueva computador

ESTRUCTURA ANDROID

AndroidManifest – nodos descriptivos sobre las características de la aplicación
java – Código fuente de la aplicación
res – Se encuentra layouts, traducciones, archivos multimedia, otros.
readme.md – Requisitos e instrucciones para ejecutar la aplicación en un nueva computador

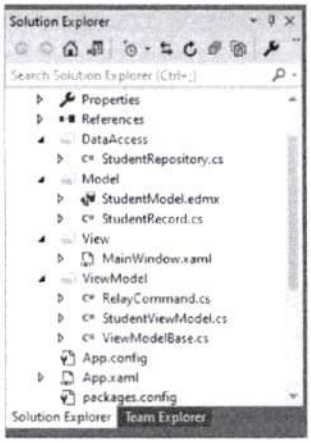
Serpost
El Correo del Perú

ESTRUCTURA iOS

Project-Info.plist – nodos descriptivos sobre las características de la aplicación
Resources – Se encuentra traducciones, archivos multimedia, otros.
readme.md – Requisitos e instrucciones para ejecutar la aplicación en un nueva computador

Serpost
El Correo del Perú

Xamarin



DataAccess – Acceso a servicios web externos o base de datos local
Model – Mapeo de entidades
View – Capa de pesentación
ViewModel – Capa de lógica de negocio, el nexo para DataAccess y View

Serpost
El Correo del Perú

Flutter

repository – Acceso a servicios web externos o base de datos local
models – Mapeo de entidades
view – Capa de pesentación
view_model – Capa de lógica de negocio, el nexo para DataAccess y View

Serpost
El Correo del Perú

Equipo de trabajo

Herramientas:

- Visual Studio 2019 +
- Visual Studio Code
- Oracle SQL Developer
- SQL Server Management 2022 +
- Notepad++
- SOAP UI
- Postman
- Code compare
- Git
- Toad Data Modeler
- Toad database developer tools
- Microsoft visio
- Bizagi Modeler
- DbForge Studio
- DbModellet.net
- Android Studio
- XCode

Serpost
El Correo del Perú

VI. DISPOSICIONES COMPLEMENTARIAS

1. El presente documento deja sin efecto a las Normas para el Desarrollo de Aplicaciones (SD-N-005.03) aprobadas con fecha 17 de agosto de 2020 y al Procedimiento Estándar para el Ciclo de Desarrollo de Sistemas (SD-P-006.02) aprobado con fecha 7 de abril de 2016.

2. HISTORIAL DE VERSIONES

CÓDIGO	DOCUMENTO	FECHA
SD-N-005.00	Normas para el desarrollo de aplicaciones	21/07/06
SD-N-005.01	Normas para el desarrollo de aplicaciones	10/12/10
SD-N-005.02	Normas para el desarrollo de aplicaciones	27/01/15
SD-N-005.03	Normas para el desarrollo de aplicaciones	17/08/20
SD-P-006.00	Procedimiento estándar para el ciclo de desarrollo de sistemas	21/07/06
SD-P-006.01	Procedimiento estándar para el ciclo de desarrollo de sistemas	27/01/15
SD-P-006.02	Procedimiento estándar para el ciclo de desarrollo de sistemas	07/04/16

VII. APROBACIÓN

El presente procedimiento queda aprobado por Gerencia General y entra en vigencia a partir de la fecha de su suscripción.

Lima, 27 DIC. 2023

ALEXANDER INFANTES POMAR
Gerente General
Serpost