MEJORES PRÁCTICAS

|  |  |
| --- | --- |
| **TEMA** | **Tema** |
| Procesos | Tipos de documentos |
| Nombres de los documentos |
| Tipos de liberación |
| Flujo de liberación |
| Autorizadores |

|  |  |
| --- | --- |
| **TEMA** | **Tema** |
| SCRUM | Roles |
| Historias de Usuario |
| Fases de SCRUM |
| Daily |
| Ceremonias |
| Épicas |
| Graficas de SCRUM |
| Artefactos |
| Sprints |
| Características de Historias de Usuarios |
| Procesos SCRUM |

TÉCNICOS

|  |  |  |
| --- | --- | --- |
| **TEMA** | **Plataforma** | **Tema** |
| Programación Java | Fundamentos | Sintaxis. |
| Tipos de datos y variables. |
| Estructuras de control (bucles, condicionales, etc.). |
| Métodos y funciones. |
| Clases y objetos. |
| Excepciones y manejo de errores. |
| Programación orientada a objetos (POO) | Principios de la POO. |
| Encapsulamiento, herencia y polimorfismo. |
| Abstracción y composición. |
| Relaciones entre clases (asociación, agregación, composición, etc.). |
| Uso de interfaces y clases abstractas. |
| Colecciones y estructuras de datos | Uso de listas, conjuntos y mapas. |
| Iteración y manipulación de colecciones. |
| Uso de genéricos. |
| Implementación de algoritmos de ordenamiento y búsqueda. |
| Programación concurrente | Hilos (threads) y concurrencia. |
| Sincronización y manejo de condiciones. |
| Uso de bloqueos y semáforos. |
| Uso de la API de concurrencia de Java (java.util.concurrent). |
| Persistencia de datos | Uso de JDBC |
| ORM (Object-Relational Mapping) con JPA (Java Persistence API) o Hibernate. |
| Bases de datos relacionales y SQL. |
| Desarrollo de aplicaciones web con Java | Servlets y JSP (JavaServer Pages). |
| Frameworks de desarrollo web como Spring MVC |
| Manejo de sesiones y autenticación. |
| Comunicación cliente-servidor (HTTP, REST). |
| Herramientas y entorno de desarrollo | IDEs (Entornos de Desarrollo Integrados) como Eclipse, IntelliJ IDEA o NetBeans. |
| Sistemas de control de versiones como Bitbucket. |
| Gestión de dependencias con Maven. |
| Buenas prácticas y patrones de diseño | Principios SOLID. |
| Patrones de diseño como Singleton, Factory, Observer, Strategy, entre otros. |
| Organización y estructura del código. |
| Mantenibilidad y legibilidad del código. |

|  |  |  |
| --- | --- | --- |
| **TEMA** | **Plataforma** | **Tema** |
| NET CORE | Fundamentos | Arquitectura y conceptos básicos. |
| Sintaxis básica de C#. |
| Tipos de datos y variables en C#. |
| Estructuras de control (bucles, condicionales, etc.) en C#. |
| Métodos y funciones en C#. |
| Clases, objetos y programación orientada a objetos en C#. |
| Desarrollo de aplicaciones en .NET Core | Uso de la CLI (Command-Line Interface) de .NET Core. |
| Creación y gestión de proyectos. |
| Configuración y opciones de desarrollo. |
| Gestión de dependencias con NuGet. |
| Desarrollo de aplicaciones web con .NET Core | Uso del framework ASP.NET Core. |
| Routing y manejo de peticiones HTTP. |
| Middleware y pipeline de solicitud/respuesta. |
| Modelos de enrutamiento y controladores. |
| Trabajo con vistas y plantillas (Razor). |
| Autenticación y autorización. |
| Uso de Web API y servicios RESTful. |
| Acceso a datos | Uso de Entity Framework Core (ORM). |
| Trabajo con bases de datos relacionales (SQL Server, MySQL, etc.). |
| Uso de LINQ (Language Integrated Query). |
| Implementación de consultas y operaciones CRUD. |
| Desarrollo de servicios y microservicios | Diseño y desarrollo de servicios web. |
| Uso de la arquitectura de microservicios. |
| Comunicación entre microservicios (REST, mensajería, etc.). |
| Implementación de seguridad y tolerancia a fallos. |

|  |  |  |
| --- | --- | --- |
| **TEMA** | **Plataforma** | **Tema** |
| **Base de datos Oracle (Oracle PL/SQL)** | Fundamentos | Sintaxis básica de PL/SQL. |
| Tipos de datos y variables en PL/SQL. |
| Estructuras de control (bucles, condicionales, etc.) en PL/SQL. |
| Cursores. |
| Subprogramas (procedimientos y funciones) en PL/SQL. |
| Manipulación de datos | Consultas SQL en PL/SQL. |
| Manipulación de tablas y registros. |
| Uso de cláusulas como JOIN, GROUP BY, HAVING, etc. |
| Transacciones y control de concurrencia. |
| Programación modular | Creación de paquetes PL/SQL. |
| Uso de paquetes, procedimientos y funciones. |
| Reutilización de código. |
| Cursores y cursores implícitos | Uso de cursores explícitos para procesar conjuntos de datos. |
| Cursores implícitos y bucles FOR. |
| Triggers (disparadores) | Creación y uso de triggers en PL/SQL. |
| Tipos de triggers (before, after, instead of). |
| Manejo de eventos en la base de datos. |
| Manejo de excepciones | Declaración y manejo de excepciones. |
| Propagación de excepciones. |
| Uso de SQL dinámico | Construcción de sentencias SQL dinámicas en PL/SQL. |
| Uso de cursores dinámicos. |
| Mejores prácticas y optimización | Buenas prácticas de codificación en PL/SQL. |
| Optimización de consultas y rendimiento |
| Uso de índices y vistas materializadas. |

|  |  |  |
| --- | --- | --- |
| **TEMA** | **Plataforma** | **Tema** |
| **Base da datos SQL Transact SQL** | Fundamentos | Sintaxis básica. |
| Tipos de datos y variables. |
| Estructuras de control (bucles, condicionales, etc.). |
| Cursores. |
| Manipulación de datos | Consultas SQL. |
| Manipulación de tablas y registros. |
| Uso de cláusulas como JOIN, GROUP BY, HAVING, etc. |
| Transacciones y control de concurrencia. |
| Programación | Creación y uso de procedimientos y funciones. |
| Reutilización de código. |
| Cursores | Uso de cursores para procesar conjuntos de datos. |
| Triggers (disparadores) | Creación y uso de triggers. |
| Tipos de triggers. |
| Manejo de excepciones | Declaración y manejo de excepciones. |
| Propagación de excepciones. |
| Uso de SQL dinámico | Construcción de sentencias SQL dinámicas |
| Uso de cursores dinámicos. |
| Mejores prácticas y optimización | Buenas prácticas de codificación. |
| Optimización de consultas y rendimiento de código. |
| Uso de índices. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TEMA** | **Plataforma** | **Tema** |
| **Angular** | **TypeScript** | Fundamentos | Sintaxis básica de TypeScript. |
| Tipos de datos y variables. |
| Anotaciones de tipos. |
| Funciones y parámetros tipados. |
| Interfaces y tipos personalizados. |
| Clases y herencia. |
| Modificadores de acceso. |
| Características avanzadas | Tipos genéricos. |
| Union types e intersection types. |
| Discriminated unions. |
| Tipos literales. |
| Módulos y namespaces. |
| Decoradores. |
| Mixins. |
| Trabajo con JavaScript y librerías externas | Integración con JavaScript existente. |
| Uso de librerías y frameworks JavaScript en proyectos TypeScript. |
| Declaraciones de tipos para librerías externas. |
| Programación orientada a objetos en TypeScript | Encapsulamiento, herencia y polimorfismo. |
| Interfaces y clases abstractas. |
| Mixins y composición. |
| Uso de accessors (getters y setters). |
| Desarrollo de aplicaciones web con TypeScript | Uso de TypeScript en el lado del cliente (frontend). |
| Uso de frameworks como Angular, React o Vue.js con TypeScript. |
| Manipulación del DOM con TypeScript. |
| Comunicación con servicios RESTful. |
| Manejo de módulos y dependencias | Uso de módulos y sistema de módulos de TypeScript. |
| Gestión de dependencias con herramientas como npm o Yarn. |
| Configuración de proyectos y tsconfig.json. |
| Generación de código y compilación | Configuración del proceso de compilación de TypeScript. |
| Transpilación de TypeScript a JavaScript. |
| Configuración de opciones de compilación (target, module, etc.). |
| **Angular** | Fundamentos | Arquitectura y conceptos básicos. |
| Uso del CLI. |
| Creación y gestión de proyectos. |
| Estructura de un proyecto. |
| Componentes y directivas. |
| Enlace de datos (data binding). |
| Uso de pipes para transformar datos. |
| Eventos y manejo de eventos. |
| Routing y navegación | Configuración de rutas. |
| Navegación entre vistas. |
| Paso de parámetros y datos entre rutas. |
| Uso de rutas anidadas (nested routes). |
| Protección de rutas con guards. |
| Servicios y comunicación con API | Creación y uso de servicios. |
| Comunicación con una API RESTful. |
| Uso de Observables y RxJS. |
| Manejo de peticiones HTTP. |
| Interceptor HTTP y manejo de errores. |
| Formularios y validación | Creación y manejo de formularios. |
| Validación de formularios utilizando validadores integrados y personalizados. |
| Control de estado y retroalimentación visual. |
| Uso de formularios reactivos. |
| Componentes reutilizables | Creación de componentes reutilizables. |
| Uso de componentes de terceros. |
| Directivas estructurales (ngIf, ngFor, etc.). |
| Interacción con el DOM y eventos | Manipulación del DOM utilizando ElementRef y Renderer2. |
| Uso de eventos y directivas de eventos. |
| Integración con librerías externas | Uso de librerías y frameworks externos. |
| Integración con librerías de gráficos, mapas, etc. |

**MICROSERVICIOS JAVA**

|  |  |  |
| --- | --- | --- |
| **TEMA** | **Plataforma** | **Tema** |
| **Spring Boot** | Fundamentos | Arquitectura y conceptos básicos. |
| Creación y configuración de proyectos. |
| Uso de la anotación @SpringBootApplication. |
| Configuración de propiedades y perfiles. |
| Desarrollo de aplicaciones web | Uso de Spring MVC para la creación de controladores y rutas. |
| Gestión de vistas con plantillas (Thymeleaf, JSP, etc.) o RESTful API. |
| Validación de datos en formularios. |
| Manejo de sesiones y autenticación. |
| Uso de Spring Security para la seguridad de la aplicación. |
| Acceso a datos | Uso de Spring Data JPA para el acceso a bases de datos. |
| Creación de entidades y relaciones. |
| Uso de transacciones y control de concurrencia. |
| Configuración de bases de datos (MySQL, PostgreSQL, etc.). |
| Desarrollo de APIs REST | Uso de anotaciones como @RestController y @RequestMapping. |
| Mapeo de rutas y parámetros en las peticiones. |
| Serialización y deserialización de datos en JSON. |
| Manejo de dependencias y configuración | Gestión de dependencias con Maven |
| Configuración del proyecto con el archivo application.properties o application.yml. |
| Uso de perfiles y propiedades. |
| **JPA** | Fundamentos | Conceptos básicos de mapeo objeto-relacional. |
| Configuración de JPA en un proyecto Java. |
| Uso de anotaciones para el mapeo de entidades. |
| Configuración de conexiones a bases de datos. |
| Mapeo de entidades | Creación de clases de entidades. |
| Mapeo de atributos y relaciones (uno a uno, uno a muchos, muchos a muchos). |
| Uso de anotaciones como @Entity, @Table, @Column, @JoinColumn, etc. |
| Generación de claves primarias (ID) y estrategias de generación. |
| Consultas y operaciones CRUD | Uso de consultas JPQL (Java Persistence Query Language). |
| Creación de consultas utilizando el lenguaje de consultas de JPA. |
| Uso de la API Criteria para consultas dinámicas. |
| Realización de operaciones CRUD (crear, leer, actualizar, eliminar). |
| Relaciones y asociaciones | Mapeo de relaciones uno a uno, uno a muchos y muchos a muchos. |
| Uso de anotaciones como @OneToOne, @OneToMany, @ManyToOne, @ManyToMany. |
| Uso de fetch types y lazy loading para optimizar el rendimiento. |
| Transacciones y control de concurrencia | Uso de anotaciones como @Transactional. |
| Control de transacciones en operaciones de lectura y escritura. |
| Optimización de rendimiento y manejo de bloqueos. |
| Herencia y polimorfismo | Mapeo de jerarquías de clases (herencia). |
| Uso de anotaciones como @Inheritance, @DiscriminatorColumn, etc. |
| Polimorfismo en consultas y operaciones CRUD. |
| Ciclo de vida de entidades | Uso de callbacks de ciclo de vida (eventos) en entidades. |
| Manejo de eventos como @PrePersist, @PostLoad, @PreUpdate, etc. |
| Personalización de comportamiento antes y después de las operaciones. |
| Cache de segundo nivel | Configuración y uso de cache de segundo nivel en JPA. |
| Uso de proveedores de cache como Ehcache. |

**Microservicios Net Y .NET**

|  |  |  |
| --- | --- | --- |
| **TEMA** | **Plataforma** | **Tema** |
| **Entity Framework** | Fundamentos | Conceptos básicos de mapeo objeto-relacional. |
| Uso de Entity Framework en un proyecto .NET. |
| Configuración de conexiones a bases de datos. |
| Uso de la clase DbContext y DbSet. |
| Mapeo de entidades | Creación de clases de entidades. |
| Mapeo de atributos y relaciones (uno a uno, uno a muchos, muchos a muchos). |
| Uso de atributos como [Table], [Column], [Key], [ForeignKey], etc. |
| Configuración de claves primarias y estrategias de generación. |
| Consultas y operaciones CRUD | Uso de LINQ (Language Integrated Query) para consultas y operaciones CRUD. |
| Consultas LINQ con operadores como Where, Select, Join, etc. |
| Uso de métodos de extensión como First, Single, Include, etc. |
| Uso de métodos asincrónicos para operaciones asíncronas. |
| Relaciones y asociaciones | Mapeo de relaciones uno a uno, uno a muchos y muchos a muchos. |
| Uso de atributos como [OneToOne], [OneToMany], [ManyToOne], [ManyToMany]. |
| Configuración de carga ansiosa y diferida (eager loading y lazy loading). |
| Herencia y polimorfismo | Mapeo de jerarquías de clases (herencia). |
| Uso de atributos como [Inheritance], [DiscriminatorColumn], etc. |
| Polimorfismo en consultas y operaciones CRUD. |
| Optimización y rendimiento | Optimización de consultas con el uso de índices y joins. |
| Configuración de carga ansiosa y diferida para mejorar el rendimiento. |
| Uso de caché de consultas y caché de entidades. |

|  |  |  |
| --- | --- | --- |
| **TEMA** | **Plataforma** | **Tema** |
| JAVASCRIPT | Fundamentos | Sintaxis básica de JavaScript. |
| Tipos de datos y variables. |
| Operadores y expresiones. |
| Estructuras de control (bucles, condicionales, etc.). |
| Funciones y ámbito de variables. |
| Trabajo con objetos y prototipos. |
| Eventos y manejo del DOM (Document Object Model). |
| Programación asíncrona | Callbacks. |
| Promesas. |
| Async/await. |
| Trabajo con AJAX y peticiones HTTP. |
| Manipulación del DOM | Selección y modificación de elementos. |
| Manipulación de estilos y clases. |
| Creación y eliminación de elementos. |
| Manejo de eventos. |
| Frameworks y librerías populares | Uso de bibliotecas como jQuery |
| Patrones de diseño en JavaScript | Patrón módulo. |
| Patrón observador. |
| Patrón constructor. |
| Patrón fachada. |
| Otros patrones comunes. |
| Programación orientada a objetos en JavaScript | Prototipos y herencia prototípica. |
| Clases y objetos en ECMAScript 6+. |
| Encapsulamiento y abstracción. |
| Uso de Mixins y composición. |
| APIs y servicios web | Uso de APIs nativas del navegador. |
| Consumo de servicios RESTful. |
| Trabajo con JSON y XML. |