UNIVERSIDAD LAICA ELOY ALFARO DE MANABÍ

Facultad de Ciencias de la Vida y Tecnologías
Carrera de Software

Trabajo Autónomo - Primer Parcial

Docente: John Cevallos
Asignatura: Aplicación para el Servidor Web

Objetivo General

Desarrollar un sistema completo utilizando múltiples lenguajes de programación y tecnologías, implementando una arquitectura distribuida que integre servicios REST, GraphQL, WebSockets y un frontend interactivo. El proyecto debe demostrar competencias técnicas en desarrollo full-stack, trabajo colaborativo y gestión de proyectos de software.

Condiciones y Reglas

1. Generales

- Las arquitecturas aplicadas deben sustentarse en documentación verificable (artículos, libros o páginas oficiales).
- Cuidar detalles en cuanto al tema definido por el grupo y revisado por el docente.
- El código debe ser subido a un repositorio grupal en GitHub (de preferencia privado con acceso al correo del docente: joancemac@gmail.com).
- No se requiere publicación de la solución en ningún hosting.
- Debe aplicar los conceptos revisados en la asignatura en el primer parcial.

2. Composición de Grupos

- Grupos de 3 integrantes obligatoriamente.
- Cada grupo debe registrarse con el docente antes de iniciar el desarrollo.
- Los grupos permanecerán fijos durante todo el desarrollo del proyecto.

3. Distribución de Lenguajes de Programación

Cada integrante debe elegir UN lenguaje de programación para desarrollar su componente:

- Integrante 1: Python - Desarrolla el componente que elija usando Python.
- Integrante 2: TypeScript - Desarrolla el componente que elija usando TypeScript.
- Integrante 3: Lenguaje Especializado - Elige uno de: Golang, Rust, Ruby o Kotlin.

Nota: Los integrantes tienen libertad total para decidir qué componente desarrollar con su lenguaje asignado, siempre que se cubran todos los requisitos arquitectónicos del sistema.

4. Arquitectura del Sistema Requerida

El sistema debe implementar obligatoriamente los siguientes componentes:

1. **Servicio REST**
   - Funcionalidad básica de la aplicación
   - Operaciones CRUD principales
   - Gestión de autenticación y autorización
   - API REST bien documentada
   - Manejo de validaciones y errores

2. Servicio GraphQL
- Capa de reportes y consultas complejas
- Schema bien definido para consultas de datos
- Resolvers eficientes para agregación de información
- Consultas optimizadas para reportes analíticos

3. WebSocket Server
- Dashboard en tiempo real
- Gestión de conexiones de clientes
- Emisión de eventos y notificaciones en tiempo real
- Manejo de salas/canales para diferentes tipos de datos

4. Frontend
- Interfaz de usuario que integre todas las capas
- Consumo del servicio REST para operaciones básicas
- Integración con GraphQL para mostrar reportes
- Conexión WebSocket para datos en tiempo real
- Dashboard interactivo y responsivo

5. Integración de Tecnologías

Requisito crítico: La forma como se integran todas estas tecnologías es un parámetro fundamental de evaluación. El sistema debe demostrar comunicación fluida entre componentes, consistencia de datos, experiencia unificada en el frontend, notificaciones en tiempo real y reportes dinámicos.

6. Funcionalidades Mínimas Requeridas

• CRUD completo para entidades del emprendimiento.
• Sistema de autenticación funcional.
• Reportes vía GraphQL.
• Notificaciones en tiempo real.
• Dashboard interactivo.
• Validación de datos en todos los componentes.
• Manejo de errores estructurado.

7. Documentación

• **README.md** completo con descripción del proyecto y arquitectura.
• Instrucciones de instalación y configuración.
• Guía de uso de cada componente.
• Endpoints de APIs documentados.
• Explicación de la integración entre tecnologías.

Rúbrica Técnica y de Soft Skills (100% de la nota)

Categoría	Peso	Detalle de Evaluación
Integración de Tecnologías	30%	Comunicación entre Componentes (15%), Consistencia de Datos (8%), Experiencia de Usuario Unificada (7%)
Implementación Técnica	25%	Funcionalidad REST (8%), Capa GraphQL (8%), WebSocket en Tiempo Real (9%)
Frontend y UX	15%	Interfaz de Usuario (8%), Dashboard Interactivo (7%)
Arquitectura y Diseño	10%	Diseño de Sistema (5%), Gestión de Estado (3%), Seguridad (2%)
Trabajo Colaborativo	10%	Distribución Equitativa (3%), Liderazgo y Coordinación (3%), Comunicación y Colaboración (2%), Adaptabilidad (2%)
Gestión de Proyecto	5%	Planificación y Organización (2%), Gestión de Recursos (2%), Proactividad y Seguimiento (1%)
Presentación y Demo	5%	Demostración Técnica (3%), Explicación de Arquitectura (2%)

Factor de Participación Individual

Se evaluará el aporte individual mediante commits semanales al repositorio Git durante las 5 semanas de desarrollo. Estos commits representan un 25% del factor que se aplicará sobre la nota obtenida.

Criterios:

- Commits significativos.
- Consistencia semanal.
- Calidad y mensajes descriptivos.
- Participación en la integración de componentes.

Ejemplo de cálculo: Si la nota base es 8.5 y el estudiante solo aporta en 4 semanas, su factor es 80%. Nota final: 8.5 × 0.8 = 6.8

Cronograma de Desarrollo

Semana	Actividades Principales
Semana 3	Inicio del Proyecto: Definición del tema, diseño de arquitectura, setup inicial de repositorios, distribución de componentes.
Semana 4	Desarrollo Individual: Implementación de componentes REST, GraphQL, WebSockets y bases de datos.
Semana 5	Integración Inicial: Comunicación entre servicios, desarrollo del frontend, primeras integraciones.
Semana 6	Integración Completa: Finalización del frontend, dashboard en tiempo real, testing de integración, documentación.
Semana 7	Presentación: Refinamiento final, preparación de demo y exposición oral del proyecto.

Recomendaciones Técnicas

- Tecnologías sugeridas para REST, GraphQL, WebSockets y Frontend.
- Uso de PostgreSQL, MySQL o MongoDB como bases de datos.
- Documentación clara en README.md.
- Gestión con GitHub Projects, issues y milestones.
- Frontend recomendado: React, Vue o Angular (con Material-UI o Tailwind).

Entrega Final - Semana 7

Componentes de la entrega:

1. Repositorio GitHub con código completo.
2. README.md con documentación técnica.
3. Presentación oral de 20-25 minutos con demo en vivo.
4. Reflexión grupal sobre el proceso de desarrollo.

Criterios de aceptación:

- Sistema completo y funcional.
- Frontend que consuma REST, GraphQL y WebSockets.
- Dashboard en tiempo real operativo.
- Reportes dinámicos.
- Código bien estructurado y documentado.
- Participación equitativa de los integrantes.
- Demostración exitosa de la integración.