CV-ENG

joel.teodoro.software@gmail.com · (+34) 640 31 80 80 · /in/joel-teodoro-gomez/ ·
github.com/JoelTeoGom

JOEL TEODORO GÓMEZ

JUNIOR SOFTWARE ENGINEER

+4 years of experience in programming, focused on backend development and microservices architecture. Experienced in working with Java, Go, C, JavaScript, and Python. Skilled in building high-performance servers, designing custom protocols, and working with key Go packages such as net, tcp/ip, and context. Well-versed in messaging systems and queues like Kafka, RabbitMQ, Redis, and gRPC, ensuring efficient and scalable communication between services. Skilled in Java frameworks such as Spring Boot, with hands-on experience using sockets and Kafka to develop real-time, distributed applications. Comfortable working with Docker and Git for continuous integration and deployment. Additionally, experienced in using Linux as a development environment and capable with Bash scripting.

MOST RECENT PROJECTS

HTMX E-commerce Application

Technologies: HTMX, Go (Golang), PostgreSQL, Docker, JWT, Tailwind CSS

https://github.com/JoelTeoGom/htmx-golang-ecommerce

RESPONSIBILITIES

- **Full Stack Development:** Designed and developed an e-commerce web application using **HTMX** for the frontend, **Go** for the backend, and **PostgreSQL** for the database.
- **Backend Implementation:** Used Go's **net/http** and **html/template** libraries to create an efficient backend capable of serving dynamic content and handling requests.
- **User Authentication:** Created secure authentication using JSON Web Tokens (**JWT**), managing protected routes with **middleware**.
- **Product Management:** Implemented features for user registration, login, product listing, and cart management.
- **Security and Middleware:** Integrated robust security measures against **XSS**, **CSRF**, and **SQL injection**, using middleware for route protection.
- **Database Design:** Structured a relational database using **PostgreSQL**, including tables for users, products, carts, and orders.
- **Docker Containerization:** Deployed the entire application using **Docker**, allowing easy management of dependencies across environments.
- **Frontend Optimization:** Developed dynamic web pages without relying on heavy JavaScript libraries, utilizing HTMX to simplify the creation of dynamic content.

KEY ACCOMPLISHMENTS

- **Efficiency with HTMX:** Simplified the creation of dynamic web pages by reducing JavaScript overhead, utilizing **HTML** attributes to manage **AJAX** requests and update content.
- **Security Implementation:** Applied strong authentication mechanisms with **JWT**, ensuring secure access to user data and preventing common vulnerabilities.
- **Lightweight Frontend:** Built an interactive and responsive UI using **HTMX** and **Tailwind CSS**, improving performance and user experience on various devices.
- **Scalable Backend:** The **Go** backend provided fast and efficient request handling, ensuring scalability for potential future traffic spikes.
- **PostgreSQL Management:** Ensured data consistency and integrity through well-structured relational models, with efficient querying to handle product and order data.

- **Docker Integration:** Streamlined the development and deployment process by packaging the frontend, backend, and database within **Docker** containers.

EasyCab Autonomous Taxi System

Technologies: Spring Boot, Sockets, Kafka, PostgreSQL, Docker, Distributed Systems

https://github.com/JoelTeoGom/easycab

RESPONSIBILITIES

- **Distributed Backend Development (Spring Boot):** Developed a backend system simulating autonomous taxis, managing system logic and communication between components.
- **Custom Socket Communication Protocol:** Designed a **socket**-based **protocol** for secure, real-time data exchange between taxi sensors and the Digital Engine.
- **Real-Time Sensor Communication:** Built continuous communication where sensors report obstacles, enabling the Digital Engine to control taxi movement in real-time.
- **Taxi-Central ServerSocket Connection:** Implemented a ServerSocket on the Central server to handle connections, track taxis, and manage authentication through a **HashMap**
- **Kafka Integration for Fault Tolerance:** Integrated Kafka for data streaming between Central, taxis, and customers, ensuring **fault-tolerant** and continuous operations.
- **Customer Request and Taxi Assignment via Kafka:** Handled customer requests and taxi assignments through Kafka, using **dynamic topics** for efficient, scalable handling of **concurrent** requests.
- **PostgreSQL Database Management:** Designed a database with tables for taxis, customers, and locations, optimizing data storage and retrieval for fleet and customer management.

KEY ACCOMPLISHMENTS

- **Custom Socket Protocol:** Developed a secure, real-time communication **protocol** between taxi sensors and the Digital Engine.
- **Fault-Tolerant Taxi Management:** Built a Kafka-based system ensuring continuous taxi operations, even during microservice failures. Additionally, implemented logic for sockets to automatically attempt reconnection if the connection is lost, ensuring uninterrupted communication between taxis and the central server.
- **Scalable Kafka Topics:** Implemented **dynamic** Kafka **topics** to efficiently manage multiple customer and taxi requests in real-time.
- **Centralized Logic Management:** Centralized the system's core functionalities in Spring Boot for efficient fleet management and customer service.

E-commerce Full-Stack

Technologies: React.js, Node.js, Express, PostgreSQL, Stripe, JWT, TypeORM, Docker, Material-UI

https://github.com/JoelTeoGom/FullStack-app

RESPONSIBILITIES

- **Full Stack Development:** Created a full-stack web application utilizing **React** for the frontend, **Node.js** and **Express** for the **backend**, and **PostgreSQL** for data management.
- **Backend Development:** Designed and implemented the backend application, ensuring proper functionality of server logic, user management, and database interactions.
- **User Management:** Developed an authentication and authorization system using **JWT**, managing security and permissions on both the frontend and backend.
- **Payment Gateway Integration:** Integrated payment gateways like **Stripe** and PayPal Checkout using **webhooks** for secure and efficient transactions.
- **Database Maintenance:** Designed and managed tables (users, products/services, orders) using **TypeORM** with migrations, entity schemas, and repositories.
- **Docker Containerization:** Containerized the entire application with **Docker** to maintain a consistent development and deployment

environment.

- **Frontend and Backend Configuration:** Configured and managed the environments for both backend and frontend, ensuring smooth communication between different application components.

KEY ACCOMPLISHMENTS

- **Implementation of Best Practices:** Applied backend best practices, including the use of **TypeORM**, **middleware**, and separation of responsibilities for maintainability.
- **Security Enhancement:** Strengthened application security by securely managing confidential data, persisting **JWTs**, and implementing **CORS** and cookie management.
- **Clean Code:** Ensured code readability and maintainability through proper variable naming and clear code structures, adhering to clean code principles.
- **Effective Containerization:** Successfully created and managed Docker containers for PostgreSQL, backend, and frontend using one **docker-compose** file and two **Dockerfiles**, streamlining the development process.
- **Stripe Integration:** Seamlessly integrated **Stripe** for payment processing, including webhook handling and payment state management.
- **Database Management:** Implemented migrations and seed scripts to set up and populate the database, ensuring data consistency and integrity.
- **Comprehensive React Usage:** Utilized **React hooks** and components to build a well-structured frontend application, enhancing user experience.
- **Interactive Navbar:** Developed a responsive navigation bar using React and **Material-UI**, providing users with easy access to different sections of the application.

SKILLS

- Fast learner | Team player | Adaptability | Problem solving | Active listening
- Java (+4 years) | Go (Golang) | C | Python | JavaScript | Node.js
- ReactJS | Spring Boot | RabbitMQ | Kafka | gRPC | Sockets
- PostgreSQL | Redis | Stripe | Microservices architecture | RESTful APIs | HTTP/HTTPS | WebSockets
  Linux | Bash | Git | Docker
- Object-oriented programming (OOP) | HTML/CSS | JSON | Unit Testing

EDUCATION

Software engineering degree at Universitat Rovira i Virgili

Sept. 2020 - June 2024 [Tarragona, Spain]

LANGUAGES

Spanish, Catalan, English.