**Кыргызская Республика**

**Министерство труда, социального обеспечения и миграции**

**Проект Всемирного банка «Модернизация и устойчивость электроэнергетического сектора»**

**Компонент 3 «Укрепление систем социальной защиты»**

**Технического задание на позицию: Архитектор ПО**

(Software Architect)

## **I. Общая информация**

Архитектор программного обеспечения (Software Architect) отвечает за проектирование и развитие архитектуры программного решения, обеспечивающего высокую масштабируемость, отказоустойчивость, безопасность и соответствие лучшим практикам разработки ПО.

Данный проект строится на основе **Java** для серверной части (backend) и **Vue 3 (Composition API)** для клиентской части (frontend). Основной задачей архитектора является разработка архитектурных решений, документирование технических стандартов, определение стратегий интеграции модулей и внешних сервисов, а также контроль внедрения этих решений в ходе разработки.

Архитектор ПО взаимодействует с Team Lead, Backend и Frontend разработчиками, DevOps-инженерами, QA-специалистами и UI/UX-дизайнерами для обеспечения согласованности всей системы.

## **II. Цели и задачи**

### **Основная цель:**

Разработка и документирование архитектурного решения программного продукта, обеспечивающего высокую производительность, безопасность, масштабируемость и отказоустойчивость.

### **Задачи:**

* Разработка архитектуры приложения на основе Java (Spring Boot) и Vue 3.
* Определение структуры микросервисов и методов их взаимодействия.
* Проектирование API-интерфейсов и схем потоков данных.
* Оптимизация архитектуры для масштабируемости и отказоустойчивости.
* Выбор и настройка механизмов кэширования, логирования, мониторинга.
* Определение требований к DevOps (CI/CD, контейнеризация, оркестрация).
* Разработка стратегии безопасности, включая защиту API и данных.
* Контроль соответствия разрабатываемого кода архитектурным требованиям.
* Подготовка документации и передача знаний команде разработки.

## **III. Объем работ**

Архитектор ПО обязан выполнить следующие конкретные задачи:

### **1. Разработка архитектуры системы**

* Проектирование общей архитектуры приложения с учетом микросервисного подхода.
* Определение структуры взаимодействия компонентов (backend, frontend, API, БД).
* Разработка модели данных, распределения нагрузки и хранения информации.
* Разработка стратегий интеграции сервисов и внешних API.
* Определение методологии асинхронного взаимодействия (Kafka, RabbitMQ).

### **2. Определение технологического стека**

* **Backend:** Java 17, Spring Boot, Spring Cloud, Spring Security.
* **Frontend:** Vue 3 (Composition API), TypeScript, Vite.
* **API:** RESTful (OpenAPI, Swagger), gRPC (при необходимости).
* **База данных:** PostgreSQL, Redis (кэширование).
* **Очереди сообщений:** Kafka, RabbitMQ.
* **DevOps:** Docker, Kubernetes, GitLab CI/CD, Prometheus, Grafana.
* **Аутентификация:** OAuth2, OpenID Connect, JWT.

### **3. Проектирование API и интеграций**

* Определение форматов взаимодействия (REST, WebSockets, GraphQL).
* Разработка API Gateway для маршрутизации запросов.
* Определение механизмов кэширования и rate limiting.
* Обеспечение версионности API и backward compatibility.

### **4. Оптимизация производительности**

* Настройка индексации и репликации БД.
* Оптимизация работы с высоконагруженными запросами.
* Разработка стратегии горизонтального масштабирования сервисов.
* Использование стратегий кэширования (Redis, CDN, HTTP Cache-Control).

### **5. Контроль безопасности**

* Реализация механизма RBAC (Role-Based Access Control).
* Шифрование данных (TLS, AES, Hashing).
* Защита API от атак (CORS, CSRF, SQL Injection, XSS, DDoS).
* Логирование и аудит действий пользователей.

### **6. Инфраструктура и DevOps**

* Проектирование схемы развертывания (Kubernetes, Helm Charts).
* Взаимодействие с DevOps-инженерами для автоматизации CI/CD.
* Определение мониторинга производительности и логирования.
* Настройка механизма Health Check и Circuit Breakers.

### **7. Документирование архитектурных решений**

* Подготовка архитектурных диаграмм (C4 Model, UML).
* Документация API в Swagger/OpenAPI.
* Описание сценариев взаимодействия сервисов.
* Проведение презентаций и обучающих сессий для команды.

## **IV. Требования к кандидату**

### **Технические компетенции:**

* Опыт работы архитектором ПО от 5 лет.
* Глубокие знания **Java (Spring Boot, Spring Cloud)**.
* Опыт работы с **Vue 3 (Composition API), TypeScript**.
* Знание принципов **DDD (Domain-Driven Design)**, **Event-driven архитектуры**.
* Опыт проектирования API (**REST, gRPC**).
* Опыт работы с базами данных **PostgreSQL, Redis**.
* Опыт DevOps-практик (**Docker, Kubernetes, CI/CD**).
* Навыки работы с брокерами сообщений **Kafka, RabbitMQ**.
* Опыт обеспечения безопасности (**JWT, OAuth2, OpenID Connect**).

## **V. Механизмы отчетности и контроля**

Архитектор ПО обязан предоставлять:

* **Еженедельные отчеты** по архитектурным решениям.
* **Документацию архитектуры**, диаграммы и схемы взаимодействия.
* **Результаты код-ревью**, выявленные проблемы и рекомендации.
* **Финальный отчет** с детальным описанием реализованных решений.
* **Основные архитектурные и другие** технические требования к итерациям — перед началом новой итерации будет подготовлен и согласован с заказчиком и группами специалистов документ, включающий требования и предполагаемую продолжительность

Отчеты направляются **Team Lead и заказчику**.

## **VI. Сроки выполнения работы**

1. Срок контракта – 3 месяцев (с возможностью продления).

## **VII. Институциональные механизмы**

Архитектор ПО взаимодействует с:

* **Team Lead** – для согласования архитектурных решений.
* **Backend и Full-stack разработчиками** – для контроля реализации API и бизнес-логики.
* **Frontend разработчиком** – для согласования API и интеграции клиентской части.
* **DevOps-инженерами** – для настройки развертывания и CI/CD.
* **QA-инженерами** – для организации тестирования.

Разработка ведется с использованием **Git, Jira, Confluence**.

## **VIII. Конечные результаты**

Результатом работы Архитектора ПО является:

* Разработанная и документированная архитектура системы.
* Оптимизированная структура данных, API и сервисов.
* Готовая документация архитектурных решений.
* Внедренные механизмы отказоустойчивости, безопасности, масштабируемости.
* Передача знаний и обучение команды.

## **IX. Конфиденциальность и интеллектуальная собственность**

* Все результаты работы и права интеллектуальной собственности принадлежат заказчику.
* Запрещено разглашение конфиденциальной информации.
* Все проектные материалы передаются заказчику после завершения работы.