Мироненко Даниил

Junior Golang Developer

Бэкенд-разработчик с опытом продуктовой разработки на Go, интересующийся highload-оптимизациями. Занимался инфраструктурной разработкой на C++. Особое внимание уделяю командной работе, а также инфраструктуре проекта, применяя современные DevOps-подходы и методы Infrastructure as Code (IaC) для повышения эффективности процессов разработки.

Опыт

NovaMusic golang rest api grpc microservices postgresql clean arch minio

Музыкальный стриминговый сервис, реализованный в рамках семестрового командного проекта в VK Education. Проект включает ключевые функции воспроизведения треков, управления плейлистами, полнотекстовый поиск и многое другое.

- реализовал полнотекстовый поиск в PostgreSQL с поддержкой поисковых запросов на русском языке, работал с GIN индексами, весами и кастомными словарями
- добавил сбор метрик и настроил дашборд с метриками состояния сервиса, используя Prometheus, Node Exporter и Grafana
- активно занимался разработкой RESTful и gRPC микросервисов: реализовал авторизацию на jwt-токенах, внедрил gRPC, добавил загрузку файлов в S3 хранилище, защитил сервис от sql-инъекций, реализовал различные middleware для защиты от CSRF-атак, CORS и многое другое
- занимался переходом от модульного монолита к микросервисной архитектуре: добавил gRPC-интерфейсы, почистил зависимости
- занимался настройкой инфраструктуры проекта: настроил контейнеризацию в Docker Compose, написал Makefile'ы, добавил алертинг в Telegram
- написал Github Actions пайплайны для автоматической проверки кода, непрерывной доставки и развертывания микросервисов на сервере
- написал конфигурацию для nginx сервера: проксирование на микросервисы, ssl сертификаты, кэширование, http2, сжатие http-ответов
- писал интеграционные и юнит тесты, генерировал моки зависимостей
- проводил code review участников команды

Fiber c++ concurrency async fiber stackfull futex cmake google test

Библиотека многопоточных кооперативных stackfull файберов для C++, в которой реализованы основные концепции асинхронного программирования для повышения эффективности высоконагруженных систем.

- занимался проектированием модульной архитектуры, продумал read-only интерфейс для файберов
- написал обертку для системного вызова futex
- проводил анализ и оптимизацию работы файберов, включая внедрение интрузивных очередей и уменьшение количества синхронизаций кэшей процессорных ядер
- написал Github Actions пайплайн для сборки и тестирования библиотеки, настроил ccache-кэширование для ускорения компиляции и сборки проекта
- занимался настройкой инфраструктуры для разработки библиотеки: настроил контейнеризацию в Docker с эмуляцией x86/64, добавил поддержку выполнения таргетов Makefile как локально, так и внутри контейнера, добавил раздельную debug и release сборку
- работал с различными санитайзерами: *ASAN*, *UBSAN*, *TSAN*
- написал документацию по установке через FetchContent API CMake'а и по работе с компонентами библиотеки
- написал и замерил benchmark'и для анализа производительности библиотечных файберов и стандартных потоков *std::thread*
- написал unit-тесты, используя фреймворк GoogleTest

AskMe python django memcached centrifugo web socket nginx gunicorn

Сервис, поддерживаемый сообществом, где пользователи могут задавать вопросы и находить на них ответы.

- реализовал real-time сообщения, работающие на web socket'ах через сервер Centrifugo
- настроил Gunicorn в качестве application-сервера для улучшения производительности сервиса
- добавил кэширование в Memcached для оптимизации работы сервиса
- провел нагрузочные тестирования для выявления проблем с производительностью и узких мест
- написал конфигурацию для nginx сервера: раздача и кэширование статики, сжатие http-ответов, проксирование запросов
- работал с Django Rest Framework для внедрения в сервис AJAX
- оптимизировал запросы к ORM, чтобы они вычислялись лениво

> Подробнее о каждом проекте на GitHub.

Soft Skills

- Организация времени
- Работа в команде (Agile, Gantt)
- Приоритизация задач (MoSCoW, RICE, Kano)
- Постановка задач (SMART)
- Формулировка функциональных и нефункциональных требований
- Предложение идей по улучшению проекта
- Управление рисками

Контакты

Telegram: @daronenko
Почта: dan.mironenko@gmail.com
Телефон: +7 (916) 708-00-06
GitHub: @daronenko
LeetCode: @daronenko
Языки: русский, английский
Город: Москва

Навыки

Основные языки: golang, c++, python, c, sql
Библиотеки & фреймворки: fiber, uber fx, easyjson, django, c api (python)
Инструменты: linux, rest, grpc, web socket, pprof, cgo, nvim, makefile, cmake, swagger
Инфраструктура: docker, docker compose, nginx, ci/cd, github actions, prometheus, grafana, centrifugo, alertmanager, gunicorn
Базы данных & хранилища: postgresql, memcached, minio

Образование

МГТУ им. Н. Э. Баумана
Москва, 09/22 - н.в.
Информатика и вычислительная техника (Компьютерные системы и сети, ИУ6). 4.86/5.00 GPA.
Дисциплины: ООП, сети, ОС, машинно-зависимые языки программирования (NASM, x86/64) и основы компиляции.

Образовательный центр VK в МГТУ (ex. Технопарк)
Москва, 02/24 - н.в.
Программа по веб-разработке.
Дисциплины: разработка микросервисов на Go, алгоритмы и структуры данных, системный дизайн, СУБД.

Последние курсы

- Углубленный Python от VK Education
- Тренировка по алгоритмам 5.0 от Яндекса
- Тренировка по алгоритмам 4.0 от Яндекса
- Открытый лекторий Школы Бэкенд Разработки 2023 от Яндекса
- Продвинутое программирование на C++ от Ильи Мещерина

Хакатоны

CSAT Survey csat survey microservice http rest api

Микросервис CSAT опросов для сервиса NovaMusic, предоставляющий REST API бэкенд для проведения опросов и оценки уровня удовлетворенности пользователей.

- разработал архитектуру микросервиса на основе принципов Clean Architecture
- реализовал HTTP ручку для получения CSAT вопросов по заданному топику, а также ручку для сохранения ответа пользователя

Web Framework python framework middleware gunicorn

Полноценный web framework, поддерживающий создание REST API, раздачу HTML-страниц, server side rendering, механизм middleware, URL-паттерны, query-параметры, конфигурирование web-сервера, редиректы и переопределение логики обработки ошибок (например, 404 Not Found).

- разработал структуру и инфраструктуру проекта: контейнеризацию в Docker, написал Makefile, настроил сборку фреймворка в модуль через *setuptools*, добавил независимый playground с примером использования
- написал Github Actions пайплайн для проверки исходного кода линтерами и запуска unit-тестов
- спроектировал архитектуру фреймворка и разделил его на независимые компоненты
- интегрировал Gunicorn в качестве основного WSGI-сервера и вынес интерфейс для его конфигурирования на пользовательский уровень
- реализовал два вида ответа сервера: json и редирект
- разработал механизм middleware и определил интерфейс взаимодействия с ним
- реализовал основу для роутера