

## Прокопович Антон

E-mail: 96anton96@mail.ru

Телефон: +7 (995) 656-04-32

Телеграм: @a\_prokopovich

Дата рождения: 25.09.1996



### Опыт работы:

#### *Bitsgap: Golang разработчик (2022 – н.в.)*

- Проекты:

1. Бот автоматизации биржевой торговли “DCA Bot”.

- Статический анализ кода. Построение графов вызовов с помощью утилиты rprof) с целью изучения кодовой базы и документирование существующей архитектуры.
- Локализация и исправление багов системы. Для оптимизации процесса отладки настроил возможность локального запуска системы - Dockerfile для сборки образа с инстансом базы данных Tarantool и всего приложения на Go. А также скрипт для автоматической пересборки образа с новыми исходниками базы данных (.lua файлы) в случае обновления схемы данных.
- Рефакторинг и переписывание компонентов, дополнение документации в коде.
- Покрытие кода юнит и интеграционными тестами. Преимущественно использовались библиотеки тестирования Testify и GoMock.
- Реализация нового функционала системы (валидация формата и значений приходящих с фронтэнда структур; модификация настроек уже запущенных на бирже ботов; новая логика расчета “сетки” ордеров на покупку; новая логика выставления ордеров с учетом заданных пользователем лимитов; расчет комиссии со сделок).

Использовались следующие технологии и фреймворки:

gRPC, Websockets, Docker, СУБД Tarantool, ClickHouse, Prometheus

#### *Chatex: Golang разработчик (2019 – 2021)*

- Проекты:

1. Сервис интеграции с блокчейном Waves.

- Исследование библиотеки от сервиса на Go в силу отсутствия полной документации (например, на предмет возможности генерации публичных и приватных ключей локально, без gRPC);
- Реализация логики постоянного сканирования цепочки блоков и вывода депозитов из этого блокчейна на общий счет (сложность - нужно было

выводить с минимальной задержкой, но максимальной уверенностью, что блок однозначно подтвердился);

- Написание юнит, интеграционных тестов, небольшое перепроектирование с целью перевести остальные сервисы-интеграторы с другими блокчейнами на схожую общую архитектуру;

Использовались следующие технологии и фреймворки: gRPC,

веб-фреймворк Echo для реализации интерфейса взаимодействия с нодами блокчейна и внутренними сервисами компании, Testify (как библиотека для unit тестирования, GoMock – для генерации моков для тестов;

## 2. Рефакторинг и локальное перепроектирование 3 проектов-интеграторов с другими блокчейнами (TRX, ETH, XMR).

- Выделение общих частей для всех проектов, проектирование интерфейсов core компонентов сервисов, согласование, брейншторм с командой по поводу новой архитектуры.
- Рефакторинг и переписывание компонентов всех трех сервисов под новые интерфейсы и правка написанных юнит тестов, написание новых, дополнение документации в коде.
- Добавление Prometheus метрик и дифференцированного логирования во все сервисы.

Использовались следующие технологии и фреймворки: PUML (для визуализации всех диаграмм в документации), веб-фреймворк Echo для реализации интерфейса взаимодействия с нодами блокчейна и внутренними сервисами компании, Testify (как библиотека для unitтестирования, GoMock – для генерации моков для тестов; Pprof и gops + Prometheus для профилирования и оптимизации.

## 3. Интеграция с сервисом KYC ChainAnalysis.

Интеграция нужна была на предмет обнаружения и реагирования в автоматическом или полуручном режиме на подозрительные транзакции (как депозиты, так и выводы).

- Взаимодействие с сервисом происходило через HTTP API – написал для него внутреннего клиента и часть реализации пула клиентов (с возможностью настраивать rate limit'ы, количество параллельно запущенных, удобно масштабировать программно в пиках).
- Нужно было много общаться, планировать, координировать и проектировать. Так как, по причине асинхронности работы сервиса (оставляется заявка, начинается long-polling'ов статуса конкретной заявки, затем можно получить некий статус - метаданные об alert'ах и тд), а также по причине сложной внутренней логики (какие-то транзакции

должны были уходить на рассмотрение дежурным аналитикам, затем попадать обратно).

- Архитектура проекта была выбрана на основе event-driven подхода - были независимые обработчики, которые выполняли свою часть логики над транзакциями в определенных состояниях, пересылали их друг другу на этапы, во внутренние сервисы, общались с ChainAnalysis, получали на вход новые транзакции от ядра системы.

Использовались следующие технологии и фреймворки: объединения их взаимодействия между собой и с внешними сервисами использовалась библиотека Watermill. Для HTTP клиентов, их пула, настроек лимитирования использовались библиотеки tunny, ratelimiter от Uber, Resty (как HTTP client, т.к. удобнее было в нем сохранять данные сессий, изолированные данные лимитеров, контекстов и пр).

В качестве шины данных сервиса был выбран RabbitMQ (т.к. он уже использовался в инфраструктуре многих проектов). В качестве БД для сохранения промежуточных состояний проверки транзакций использовалась MongoDB (т.к. сообщения о транзакциях на тот момент очень варьировались по схеме).

Для кэш-хранилища и хранилища состояний (с TTL) использовался Redis.

В качестве интерфейса для взаимодействия с фронтендом и внутренними сервисами был выбран gRPC + gRPC Gateway для REST API фронтенду.

Также из-за слишком выросшей сложности и асинхронности логики сервиса пришлось начать писать интеграционные тесты (при помощи Ginko).

## *Zyfra Robotics: Python разработчик (2017 – 2019)*

### ◦ Проекты:

#### 1. Роботизированный беспилотный буровой станок.

- Написание всего “бортового” ПО – алгоритмов управления роботом.
- Написание консольных симуляторов органов управления робота для тестирования бортового ПО.
- Написание автотестов (юнит и интеграционных).

Использовались следующие технологии и фреймворки:

ROS; ZeroMQ; Docker; OpenCV; pytest

#### 2. Сервер управления беспилотным транспортом на роботизированном карьере.

- Написание бэкенд-сервера, через который диспетчер взаимодействует с роботами работающими на карьере.

- Создание панелей визуализации статистических данных о работе роботов.
- Полное перепроектирование и переписывание сервера на новую архитектуру и новый стэк технологий.
- Подробная документация терминологии проекта, API бэкэнда, архитектуры проекта на уровне сервисов и архитектуры сервисов на уровне объектов.
- Написание телеграм-бота для получение статистики о работе робота и отчетов о текущем статусе.

Использовались следующие технологии и фреймворки:

Flask (REST API); MongoDB; MySQL; Prometheus; Graphana; OAuth; Telegram-Bot-API; PURL; Pytest; Docker.

3. Десктоп-приложение “Рабочее место” оператора дистанционного управления.

- Графический интерфейс и протокол взаимодействия для дистанционного управления роботами.
- Локализация интерфейса для англо- и испано-говорящих пользователей.

Использовались следующие технологии и фреймворки:

QT; Docker.

4. Симулятор роботизированного бурового станка на Unity3D.

- Написание логики и физики для симуляции всех органов управления робота, датчиков и сенсоров.
- Интергация и тестирование с системой бортового ПО.

Использовались следующие технологии и фреймворки:

Unity3D; RTSP; Docker.

Языковая школа *British Federation*: Преподаватель английского языка (2016 – 2017)

#### **Ключевые технологии и навыки:**

- Golang, Python, C#, Javascript
- MongoDB, MySQL, Redis, MinIO, Prometheus + Graphana
- ZMQ, RabbitMQ, ROS, QT, gRPC
- Linux (Ubuntu), Git, Docker
- Web-фреймворки:
  - *Golang*: Echo
  - *Python*: Flask (flask-restful), FastAPI
- ООП, паттерны проектирования объектов, архитектурные паттерны.

- Теория алгоритмов и структур данных.
- Машинное обучение и анализа данных. Лежащая в основе математика: теория вероятностей и статистика.

**Образование:**

- РГГУ, Управление персоналом (2014 – 2018);
- Presbyterian College, Applied Mathematics (2017);
- Финалист стипендиальной программы студенческого обмена Global UGRAD (2017).

**Дополнительное образование:**

Специализация «Машинное обучение и анализ данных – Яндекс, МФТИ (2018);  
Deep learning school – МФТИ (2019).

**Дополнительные навыки:**

- Английский язык (свободный – устный/письменный), Испанский язык (начальный);