100 Задач для Senior Rust Backend Разработчика (PDF SaaS)

Этот список задач разработан для программиста с опытом в Python/Go и знанием CS, желающего стать Senior Rust Backend разработчиком с фокусом на API/SaaS для сложной обработки PDF. Задачи идут от основ к продвинутым темам, постепенно вводя веб-разработку, работу с PDF и SaaS концепции.

Этап 1: Основы Rust и Инструменты (Задачи 1-15)

1. **Настройка окружения:** Установите Rust (rustup), настройте IDE (VS Code с rust-analyzer).
2. **Cargo Основы:** Создайте новый проект с `cargo new`, изучите `Cargo.toml`, команды `build`, `run`, `check`, `test`.
3. **Базовый синтаксис:** Переменные, типы данных (скалярные, составные), функции, комментарии. Напишите простую программу "Hello, world!".
4. **Управляющие конструкции:** `if/else`, циклы (`loop`, `while`, `for`). Решите несколько базовых алгоритмических задач (FizzBuzz, факториал).
5. **Владение (Ownership):** Изучите концепцию владения, правила владения. Сравните с GC в Python/Go.
6. **Заимствование (Borrowing):** Поймите изменяемые и неизменяемые ссылки. Напишите функции, принимающие ссылки.
7. **Времена жизни (Lifetimes):** Изучите базовые концепции времен жизни, аннотации времен жизни в функциях и структурах.
8. **Структуры (Structs):** Определите и используйте структуры, методы структур (`impl`).
9. **Перечисления (Enums) и `match`:** Определите енумы, используйте `match` для исчерпывающего сопоставления с образцом. Изучите `Option` и `Result`.
10. **Обработка ошибок:** Используйте `Result` для обработки возможных ошибок, `panic!` для невосстановимых ошибок. Изучите оператор `?`.
11. **Коллекции:** Изучите стандартные коллекции: `Vec`, `String`, `HashMap`. Выполните операции над ними.
12. **Модули и Crates:** Организуйте код с помощью модулей. Импортируйте и используйте внешние крейты (библиотеки) из crates.io.
13. **Тестирование:** Напишите юнит-тесты и интеграционные тесты с помощью `#[test]`.

14. **Работа с файлами:** Чтение и запись текстовых и бинарных файлов с использованием `std::fs`.
15. **Простое CLI приложение:** Напишите CLI утилиту, которая принимает аргументы командной строки (используя `std::env::args` или крейт `clap`) и выполняет простую операцию с файлом (например, подсчет строк).

Этап 2: Продвинутый Rust и Конкурентность (Задачи 16-30)

1. **Трейты (Traits):** Изучите трейты как аналог интерфейсов. Определите и реализуйте свои трейты. Используйте `dyn Trait` и `impl Trait`.
2. **Дженерики (Generics):** Напишите обобщенные функции и структуры для уменьшения дублирования кода.
3. **Замыкания (Closures):** Изучите анонимные функции, их синтаксис и способы захвата переменных из окружения.
4. **Итераторы (Iterators):** Используйте и создавайте итераторы. Изучите методы адаптеров итераторов (`map`, `filter`, `fold` и т.д.).
5. **Умные указатели:** Изучите `Box`, `Rc`, `Arc`, `RefCell`, `Mutex`, `RwLock` и их применение.
6. **Потоки (Threads):** Создавайте и управляйте нативными потоками (`std::thread`).
7. **Каналы (Channels):** Используйте каналы (`std::sync::mpsc`) для обмена сообщениями между потоками.
8. **Разделяемое состояние:** Используйте `Arc<Mutex<T>>` или `Arc<RwLock<T>>` для безопасного разделения данных между потоками.
9. **Основы `async/await`:** Изучите асинхронное программирование в Rust.
10. **Асинхронный рантайм:** Выберите и используйте асинхронный рантайм (например, `tokio` или `async-std`). Напишите простую асинхронную программу.
11. **Асинхронные задачи и `join`:** Запускайте несколько асинхронных задач параллельно.
12. **Асинхронные Mutex:** Используйте асинхронные примитивы синхронизации (например, `tokio::sync::Mutex`).
13. **Макросы:** Изучите декларативные макросы (`macro_rules!`). Попробуйте написать простой макрос.
14. **Процедурные макросы (Основы):** Поймите концепцию процедурных макросов (`derive`, атрибутные, функциональные). Попробуйте использовать `#[derive(Debug)]` и другие.

15. **FFI (Foreign Function Interface):** Вызовите простую функцию из C библиотеки в Rust. (Опционально: попробуйте вызвать Rust из Python/Go).

Этап 3: Веб-Бэкенд Основы и API (Задачи 31-45)

1. **Обзор HTTP:** Вспомните основы HTTP (методы, статусы, заголовки).
2. **Выбор веб-фреймворка:** Изучите популярные Rust веб-фреймворки (Actix-web, Axum, Rocket). Выберите один (например, Axum) для следующих задач.
3. **Базовый сервер:** Создайте простейший HTTP сервер, отвечающий "Hello, World!".
4. **Роутинг:** Определите несколько маршрутов (GET, POST) для разных URL.
5. **Обработка запросов:** Извлеките параметры из пути URL, строки запроса и тела запроса.
6. **Генерация ответов:** Формируйте ответы с разными статус-кодами, заголовками и телами.
7. **Сериализация/Десериализация JSON:** Используйте `serde` и `serde_json` для работы с JSON в запросах и ответах.
8. **Простой REST API:** Создайте CRUD API для простого ресурса (например, заметки), хранящегося в памяти (например, в `Arc<Mutex<Vec<T>>>`).
9. **Обработка ошибок в API:** Преобразуйте ошибки `Result` в соответствующие HTTP ответы.
10. **Middleware (Промежуточное ПО):** Реализуйте простое middleware для логирования запросов.
11. **Конфигурация:** Управление конфигурацией приложения (например, порт сервера) через переменные окружения или файлы (используя крейты вроде `config` или `dotenv`).
12. **Статические файлы:** Настройте раздачу статических файлов (CSS, JS, изображения).
13. **Шаблонизаторы (Опционально):** Интегрируйте шаблонизатор (например, `askama` или `tera`) для генерации HTML.
14. **CORS:** Настройте Cross-Origin Resource Sharing для вашего API.
15. **Тестирование API:** Напишите интеграционные тесты для вашего API эндпоинтов.

Этап 4: Работа с Данными и Базами Данных (Задачи 46-55)

1. **Обзор баз данных:** Вспомните основы SQL и NoSQL баз данных.

2. **Выбор ORM/Query Builder:** Изучите `sqlx` или `diesel`. Выберите один для работы с реляционной БД (например, PostgreSQL).
3. **Подключение к БД:** Настройте подключение к базе данных из вашего Rust приложения.
4. **Миграции БД:** Настройте и используйте инструмент для миграций схемы БД (например, `sqlx-cli` или `diesel_cli`).
5. **CRUD операции с БД:** Перепишите ваш API из Этапа 3 для сохранения данных в PostgreSQL вместо памяти.
6. **Асинхронная работа с БД:** Используйте асинхронные возможности `sqlx` (если выбрали его) для неблокирующей работы с БД.
7. **Транзакции:** Реализуйте операции, требующие атомарности, с использованием транзакций БД.
8. **Пулинг соединений:** Настройте и используйте пул соединений к БД (обычно встроено в `sqlx` или `diesel`).
9. **Работа с Redis (Опционально):** Интегрируйте Redis для кэширования или других задач, используя крейт `redis`.
10. **Основы индексации:** Проанализируйте запросы к БД и добавьте необходимые индексы для оптимизации.

Этап 5: Обработка PDF - База и Средний Уровень (Задачи 56-70)

1. **Структура PDF:** Изучите базовую структуру PDF документа (объекты, потоки, страницы, перекрестные ссылки).
2. **Выбор библиотек:** Исследуйте Rust библиотеки для работы с PDF (`lopdf`, `printpdf`, `pdf-extract`, `genpdf` и др.). Выберите подходящие для следующих задач.
3. **Извлечение метаданных:** Напишите программу для извлечения метаданных (автор, заголовок, количество страниц) из PDF файла.
4. **Извлечение текста (Простое):** Извлеките весь текст из простого PDF (без сложного форматирования).
5. **Извлечение текста постранично:** Извлеките текст с конкретной страницы или диапазона страниц.
6. **Извлечение изображений:** Напишите утилиту для извлечения всех изображений из PDF файла.
7. **Слияние PDF:** Создайте инструмент для объединения нескольких PDF файлов в один.
8. **Разделение PDF:** Создайте инструмент для разделения PDF на отдельные страницы или диапазоны страниц.

9. **Создание простого PDF:** Сгенерируйте простой PDF документ (например, счет-фактуру) программно, используя `printpdf` или `genpdf`.
10. **Добавление текста/водяных знаков:** Напишите программу, которая добавляет текстовый водяной знак на каждую страницу существующего PDF.
11. **Работа с аннотациями (Базовая):** Попробуйте извлечь или добавить простые аннотации (например, ссылки).
12. **Поворот страниц:** Реализуйте функцию поворота указанных страниц в PDF.
13. **API для базовых операций:** Создайте API эндпоинты для некоторых из вышеперечисленных операций (например, слияние, извлечение текста).
14. **Обработка ошибок PDF:** Научитесь обрабатывать ошибки, возникающие при парсинге или модификации некорректных PDF файлов.
15. **Оптимизация базовых операций:** Проанализируйте производительность ваших PDF утилит и попробуйте их оптимизировать.

Этап 6: SaaS Концепции, Безопасность, Тестирование (Задачи 71-80)

1. **Аутентификация (JWT):** Реализуйте аутентификацию пользователей с использованием JSON Web Tokens (JWT) в вашем API.
2. **Регистрация и вход:** Добавьте эндпоинты для регистрации новых пользователей и входа существующих (с хешированием паролей).
3. **Авторизация (RBAC):** Реализуйте простую ролевую модель доступа (например, пользователь и администратор) для защиты определенных эндпоинтов.
4. **Управление API ключами:** Разработайте систему для генерации и проверки API ключей для доступа к вашему сервису.
5. **Rate Limiting:** Внедрите ограничение частоты запросов к вашему API.
6. **Валидация ввода:** Добавьте строгую валидацию для всех данных, поступающих от пользователя (используя крейты вроде `validator`).
7. **HTTPS:** Настройте использование HTTPS для вашего API (например, с помощью `rustls` или за прокси-сервером вроде Nginx).
8. **Логирование и Трассировка:** Настройте структурированное логирование (`tracing`, `log`) и основы трассировки запросов.
9. **Модульное тестирование бизнес-логики:** Напишите юнит-тесты для не-API частей вашего приложения (сервисы, утилиты).
10. **Интеграционное тестирование с БД:** Напишите интеграционные тесты, которые взаимодействуют с реальной (тестовой) базой данных.

Этап 7: Продвинутая Обработка PDF и Интеграции (Задачи 81-90)

1. **Работа с формами PDF:** Извлечение данных из заполненных форм PDF. Попробуйте программно заполнять поля форм.
2. **Продвинутое извлечение текста:** Исследуйте методы извлечения текста с сохранением базовой структуры или таблиц (может потребовать анализа координат элементов или внешних инструментов).
3. **Генерация PDF из HTML/Markdown:** Используйте библиотеки (возможно, через FFI к `WeasyPrint` или нативные Rust) для генерации сложных PDF из HTML или Markdown.
4. **Сложная задача: Перевод PDF с сохранением форматирования:**
    - Исследуйте подходы: извлечение текста по блокам с координатами, перевод текста, вставка переведенного текста с попыткой сохранить положение и стиль.
    - Интегрируйте API машинного перевода (Google Translate, DeepL).
    - Осознайте сложности: работа со шрифтами, переполнение текста, сохранение таблиц и графики. Реализуйте лучшее возможное решение, документируя ограничения.
5. **Работа с зашифрованными PDF:** Реализуйте обработку PDF, защищенных паролем (требуется ввод пароля).
6. **Оптимизация PDF:** Исследуйте и примените техники для уменьшения размера PDF файлов (удаление дубликатов, сжатие изображений).
7. **OCR (Распознавание текста):** Интегрируйте внешнюю OCR библиотеку (например, Tesseract через FFI) для извлечения текста из отсканированных PDF (изображений).
8. **Манипуляция изображениями в PDF:** Замена или изменение размера изображений внутри PDF документа.
9. **Сравнение PDF:** Напишите утилиту для визуального или текстового сравнения двух PDF файлов.
10. **API для продвинутых операций:** Создайте API эндпоинты для некоторых продвинутых задач (например, заполнение форм, OCR, перевод).

Этап 8: Архитектура SaaS, Развертывание и Мониторинг (Задачи 91-95)

1. **Контейнеризация:** Напишите Dockerfile для вашего Rust приложения. Оптимизируйте размер образа и время сборки.

2. **CI/CD:** Настройте простой CI/CD пайплайн (например, с помощью GitHub Actions) для автоматической сборки, тестирования и (опционально) деплоя вашего приложения при пуше в репозиторий.
3. **Развертывание:** Разверните ваше приложение на облачной платформе (Heroku, Fly.io, AWS EC2/ECS, Google Cloud Run и т.д.).
4. **Фоновые задачи:** Реализуйте обработку длительных PDF задач (например, OCR или сложный перевод) в фоновом режиме с использованием системы очередей (например, Redis + `rq` или Kafka).
5. **Мониторинг и Оповещения:** Настройте сбор базовых метрик приложения (запросы в секунду, время ответа, использование ресурсов) и оповещения о критических ошибках (например, используя Prometheus + Grafana или облачные сервисы).

Этап 9: Комплексные Проекты и Подготовка к Собеседованиям (Задачи 96-100)

1. **Проект 1: SaaS для слияния/разделения PDF:**
    - Спроектируйте и реализуйте полноценный небольшой SaaS.
    - Включает: регистрацию/вход пользователей, загрузку PDF, выполнение операций слияния/разделения (возможно, в фоне), скачивание результата, управление файлами пользователя.
2. **Проект 2: SaaS API для перевода PDF (Прототип):**
    - Спроектируйте и реализуйте API для задачи перевода PDF (из задачи 84).
    - Сфокусируйтесь на архитектуре, обработке ошибок, интеграции с API перевода, обработке асинхронных задач. Четко документируйте возможности и ограничения.
3. **Open Source Контрибьюция:** Найдите интересный Rust проект (связанный с веб, PDF или системами) на GitHub и сделайте небольшой вклад (исправление бага, добавление фичи, улучшение документации).
4. **Подготовка к собеседованиям:**
    - Изучите типичные вопросы по Rust (владение, времена жизни, конкурентность, async).
    - Повторите основы CS (алгоритмы, структуры данных).
    - Изучите вопросы по системному дизайну (проектирование API, масштабируемость, базы данных).
    - Практикуйтесь в решении задач на платформах вроде LeetCode (используя Rust).
5. **Создание портфолио:** Оформите 2-3 лучших проекта из этого списка (особенно Проекты 1 и 2) в виде портфолио на GitHub/личном сайте. Подготовьтесь рассказывать о них на собеседованиях.