

Вадим

Frontend-разработчик — React / TypeScript / Next.js

Набережные Челны · открыт к удалённой работе · 2+ года коммерческого опыта

Стек: React, TypeScript, Next.js (App Router, SSR), TanStack Query, Zustand, Tailwind CSS, FSD, Vue 2, Nuxt.js, Laravel

Опыт работы

AiDenix — Frontend-разработчик

Сентябрь 2025 — настоящее время

Agro-tech платформа — учёт хозяйства

- Самостоятельно спроектировал архитектуру и определил технические требования проекта (написал ТЗ), провёл переговоры с заказчиком, вёл разработку от концепции до релиза
- Реализовал модули учёта: поля, хозяйство, техника, культуры, ресурсы, отчёты и дашборд
- Разработал функциональность отрисовки контуров полей на карте с использованием **Leaflet.js**
- Реализовал парсинг **GeoJSON** и **KML** файлов для импорта географических данных
- Спроектировал сложную форму создания полей с 3 способами ввода: ручное рисование на карте, загрузка файла, ручной ввод координат — с поддержкой одиночного и пакетного (batch) создания; валидация на **React Hook Form + Zod**
- Интегрировал данные с погодных вышек в реальном времени, настроил обмен событиями через **Pusher.js**
- Выстроил взаимосвязанную модульную архитектуру с кэшированием через **TanStack Query** во всех модулях
- UI на **Radix UI + shadcn/ui + Tailwind CSS**; стек: **React 19, Vite, TypeScript, Axios, Zustand**

R&D: 3D-визуализация в браузере

- Провёл научно-исследовательскую работу (R&D) по оценке возможностей 3D-рендеринга в браузере
- Разработал пилотный проект — интерактивный атлас животных в 3D с использованием **Three.js, React Three Fiber, drei**; стек: **React 19, Zustand**
- Выполнил значительный объём работы по производительности и оптимизации рендеринга для комфортной работы в браузере

Миграция и рефакторинг legacy CRM

- Принял в поддержку legacy-проект (**React 16, Webpack, Effector, MUI**) и провёл миграцию на **React 18 и Vite**, обновил MUI, удалил неиспользуемые зависимости
- Диагностировал и устранил критичную деградацию CI/CD: сборка занимала много времени и пайплайн падал по таймауту — оптимизировал импорты иконочной библиотеки, сократив количество модулей в бандле с ~24 000 до ~2 000, восстановив стабильность CI/CD и ускорив деплой
- Выполнил рефакторинг кода (в т.ч. исправил хуки без корректных массивов зависимостей), реализовал новый функционал и CRUD-модули
- Реализовал PWA service worker с уведомлением о доступности обновления фронтенда

Система управления проездом автотранспорта (шлагбаум/АСУДД)

- Разработал приложение для управления и мониторинга шлагбаума в реальном времени: положение, блокировка полосы, состояние проезда
- Реализовал получение событий в реальном времени через WebSocket (**Laravel Reverb + Pusher.js**): подъезд, проезд, открытие шлагбаума
- Интегрировал камеру как сканер документов с привязкой снимка к записи о проезде
- Настроил логирование ошибок для мониторинга связи и fallback-страницу при потере доступа
- Стек: **React (Vite), TypeScript, Tailwind CSS, TanStack Query, Axios**

Проходное приложение для СКУД (турникет, охрана)

- Разработал фронтенд-модуль для сотрудников охраны, встроенный в **Electron**-оболочку поверх стороннего СКУД-решения
- Самостоятельно нашёл и внедрил Electron-решение за ~1 неделю, что позволило уложиться в сроки сдачи проекта
- Реализовал логику пропуска посетителей по различным условиям
- Интегрировал сканирование документов через камеру и **SDK Regula**, с отправкой результатов на backend
- Стек: **React (Vite), TypeScript, Axios, Zustand**

Поддержка legacy-проектов

- Поддержка СКУД-системы на **Vue 2**: новый функционал по заявкам заказчика, исправление дефектов
- Техническая поддержка крупного проекта на **Nuxt.js + Laravel**: доработки и исправления по запросу заказчика

Предыдущий опыт

Nobelcode — Frontend-разработчик

Май 2024 — сентябрь 2025

Miss Diva — интернет-магазин одежды (Next.js)

- Спроектировал архитектуру фронтенда нового интернет-магазина одежды на **Next.js, TypeScript, Zustand** — определил структуру проекта и подход к организации кода
- Реализовал вёрстку и UI большей части сайта: главная, каталог, карточки товаров, информационные страницы
- Выстроил компонентную структуру с расчётом на дальнейшее подключение API
- Заложенная архитектура и написанные компоненты используются в текущей продакшен-версии сайта — проект был продолжен

- командой на основе его исходной структуры
- Работал с адаптивной вёрсткой под desktop/mobile

«Память об ушедших» — веб-платформа (Next.js + Laravel)

- Разработал платформу с нуля на **Next.js (App Router), TypeScript** (frontend) и **Laravel** (backend) — единолично, включая все архитектурные решения
- Спроектировал ролевую модель доступа: пользователь, менеджер, администратор — с отдельной CRM-панелью администратора для управления контентом и заявками
- Настроил защиту приватных маршрутов через **Next.js middleware** в зависимости от роли пользователя
- Спроектировал взаимодействие с API через **Axios** и **TanStack Query**: кэширование, ревалидация, серверные запросы
- Реализовал сложные многошаговые формы с валидацией на **React Hook Form + Zod**, включая загрузку фото/видео и drag-and-drop зоны
- Интегрировал приём онлайн-платежей через **ЮKassa** и SMS-уведомления через SMS-центр
- Стилизация на **SCSS modules**, оптимизация изображений через **next/image**; адаптивная вёрстка по дизайн-макету в **Figma**
- Полный цикл разработки за ~2 месяца: архитектура, frontend, backend, платежи, интеграции — от макета до продакшена

Nobelcode — корпоративный сайт (Astro + React)

- Разработал корпоративный сайт компании с нуля на **React, TypeScript, Astro (SSR)** — полный цикл разработки
- Реализовал SSR-рендеринг для нескольких страниц/разделов сайта — быстрая загрузка, корректная индексация для SEO
- Выполнил performance optimization загрузки изображений в галерее (lightbox) через progressive image loading / lazy loading — улучшило Core Web Vitals (LCP)
- Интегрировал систему отзывов клиентов и функциональность отмены/остановки заявок
- Организовал взаимодействие с backend через **Axios**, реализовал формы обратной связи с валидацией на **React Hook Form**
- Разработал адаптивную (responsive) вёрстку для desktop, tablet и mobile; спроектировал переиспользуемые UI-компоненты
- Выполнил разработку самостоятельно, без менторской поддержки, за ~1 месяц — полный цикл: анализ → разработка → тестирование → релиз

Личные проекты

Habit Tracker — веб-приложение для трекинга привычек (Fullstack)

- PHP 8.3, Laravel 13, Sanctum, PostgreSQL, Redis, очереди Laravel, PHPUnit · TypeScript, Next.js 16, React 19, TanStack Query/Table, Zustand, React Hook Form + Zod, Tailwind CSS 4, shadcn/ui · Docker, Docker Compose, Nginx · Telegram Bot API, Yandex OAuth
- Единолично разработал приложение со статистикой, календарём выполнения и напоминаниями в Telegram — от проектирования API до деплоя
 - Спроектировал и реализовал **REST API из 26 эндпоинтов** (версия /api/v1): авторизация, CRUD привычек, отметки, календарь и статистика; API Resources, Form Requests и Policies изолируют данные пользователей
 - Реализовал **5 типов расписаний** (ежедневно, по дням недели, N раз в неделю, раз в N дней, по числам месяца) с учётом часового пояса пользователя
 - Серия, лучшая серия и % выполнения за 30 дней пересчитываются при каждой отметке и хранятся в таблице — списки и дашборд отдают метрики без пересчёта истории на каждый запрос
 - Сделал **напоминания в Telegram**: планировщик раз в минуту обходит пользователей пачками по 500, отправка идёт через очередь (3 повтора с backoff 30 с), гарантировано не более 1 рассылки в сутки по локальному времени пользователя; в сообщении есть кнопка открытия Telegram Mini App, при блокировке бота чат автоматически отвязывается
 - Настроил безопасную авторизацию: **Sanctum-токен в HttpOnly cookie** (защита от XSS), в БД хранится только SHA-256-хэш; добавил вход через **Yandex OAuth** и **rate limiting** на логин (5 попыток в минуту на email, 20 на IP)
 - Покрывл бэкэнд **91 автотестом** (PHPUnit, feature + unit): авторизация, расписания, статистика, календарь, Telegram-интеграция, лимиты запросов
 - Фронтенд (142 TS-модуля) построен по **Feature-Sliced Design**: серверное состояние в TanStack Query, формы на Zod, защита роутов через middleware, адаптивная вёрстка, skeleton-загрузка
 - Контейнеризировал проект: **prod-стек из 6 сервисов** (nginx, php-fpm, PostgreSQL, Redis, queue-worker, scheduler); миграции применяются при старте, включён OPcache, контейнеры работают не под root; Next.js собирается в standalone-образ с security-заголовками; работает за TLS-прокси

Технический стек

Категория	Технологии
Языки	JavaScript, TypeScript, PHP 8.3
Фреймворки / библиотеки	React (16–19), Next.js (App Router, SSR), Astro, Vue 2, Nuxt.js, Laravel 13
Сборка / инструменты	Vite, Webpack, PWA (service worker)
State management	Zustand, Effector
Работа с данными / API	Axios, TanStack Query/Table, REST API, WebSocket (Laravel Reverb, Pusher.js)
Формы и валидация	React Hook Form, Zod
UI / стилизация	Tailwind CSS, Radix UI, shadcn/ui, MUI, SCSS Modules
3D / карты	Three.js, React Three Fiber, drei, Leaflet.js
Backend	Laravel, Sanctum, PostgreSQL, Redis, очереди Laravel
Тестирование	PHPUnit (feature + unit)

Инфраструктура	Docker, Docker Compose, Nginx, TLS-прокси, OPcache
Интеграции	ЮKassa, SMS-центр, Regula SDK, Telegram Bot API, Yandex OAuth, GeoJSON/KML парсинг
Прочее	Electron, Feature-Sliced Design, code review, bundle/performance optimization