**Запрос для AI помощника: Создание платформы объявлений услуг массажа и СПА**

**Общее описание проекта**

Необходимо создать полноценную платформу для размещения платных объявлений услуг массажа, СПА и косметологических процедур, аналогичную www.Avito.ru Проект должен быть разработан с использованием современного стека технологий и следовать лучшим практикам разработки.

**Технический стек (обязательно использовать)**

* **Backend**: Laravel 12 (PHP 8.3+)
* **Frontend**: Vue.js 3 + Inertia.js
* **State Management**: Pinia
* **Стилизация**: Tailwind CSS
* **Сборщик**: Vite
* **База данных**: MySQL/PostgreSQL
* **Кеширование**: Redis
* **Поиск**: Laravel Scout + Algolia/Meilisearch
* **Платежи**: Stripe/YooKassa
* **Хранилище файлов**: S3/MinIO

**Пошаговая инструкция разработки**

**Шаг 1: Архитектура и структура проекта**

Создай детальную архитектуру проекта со следующими модулями:

1. **Модуль авторизации и регистрации**
   * Регистрация через email/телефон
   * Двухфакторная аутентификация
   * Социальная авторизация (VK, Google)
   * Восстановление пароля
   * Верификация аккаунта
2. **Модуль профилей пользователей**
   * Личный кабинет исполнителя
   * Личный кабинет клиента
   * Портфолио и сертификаты
   * Рейтинговая система
   * История заказов
3. **Модуль объявлений**
   * Создание/редактирование объявлений
   * Категории и подкатегории услуг
   * Гибкая система тарифов
   * Галерея изображений
   * Расписание работы
   * Геолокация (районы города)
4. **Модуль поиска и фильтрации**
   * Полнотекстовый поиск
   * Фильтры по категориям
   * Фильтры по цене
   * Фильтры по району
   * Фильтры по рейтингу
   * Сортировка результатов
5. **Модуль оплаты и монетизации**
   * Платное размещение объявлений
   * Пакеты продвижения
   * VIP-статусы
   * Подписки для мастеров
   * История транзакций
6. **Модуль коммуникации**
   * Внутренний чат
   * Уведомления (email, SMS, push)
   * Система жалоб и модерации
   * Отзывы и комментарии

**Шаг 2: Детальная структура базы данных**

Создай следующие таблицы с миграциями Laravel:

// Основные таблицы

- users (id, name, email, phone, password, role, verified\_at, etc.)

- profiles (user\_id, bio, avatar, experience, education, certificates, etc.)

- categories (id, name, slug, parent\_id, icon, order)

- services (id, user\_id, category\_id, title, description, price\_from, price\_to, duration, etc.)

- service\_images (id, service\_id, path, is\_main, order)

- locations (id, city, district, address, lat, lng)

- service\_locations (service\_id, location\_id)

- schedules (id, service\_id, day\_of\_week, open\_time, close\_time)

- bookings (id, service\_id, client\_id, date, time, status, notes)

- reviews (id, service\_id, user\_id, rating, comment, created\_at)

- favorites (user\_id, service\_id)

- transactions (id, user\_id, type, amount, status, payment\_method)

- subscriptions (id, user\_id, plan\_id, starts\_at, ends\_at)

- notifications (id, user\_id, type, data, read\_at)

- chats (id, created\_at)

- chat\_participants (chat\_id, user\_id)

- messages (id, chat\_id, user\_id, message, read\_at)

**Шаг 3: Функциональные требования**

**3.1 Для мастеров/исполнителей:**

* Создание детального профиля с портфолио
* Размещение неограниченного количества услуг
* Управление расписанием и бронированиями
* Настройка районов обслуживания
* Загрузка сертификатов и дипломов
* Статистика просмотров и конверсии
* Управление отзывами
* Настройка автоответов

**3.2 Для клиентов:**

* Удобный поиск услуг
* Сравнение мастеров
* Онлайн-бронирование
* Избранные мастера
* История посещений
* Система лояльности

**3.3 Для администраторов:**

* Модерация объявлений
* Управление категориями
* Финансовая аналитика
* Управление пользователями
* SEO-настройки
* Массовые рассылки

**Шаг 4: UI/UX компоненты на Vue.js 3**

Создай следующие компоненты с использованием Composition API:

// Основные компоненты

- HeaderComponent.vue (навигация, поиск, профиль)

- FooterComponent.vue

- ServiceCard.vue (карточка услуги в списке)

- ServiceDetail.vue (детальная страница услуги)

- FilterSidebar.vue (фильтры поиска)

- BookingModal.vue (модальное окно бронирования)

- ReviewsSection.vue (секция отзывов)

- ChatWidget.vue (виджет чата)

- ProfileEdit.vue (редактирование профиля)

- ServiceForm.vue (форма создания/редактирования услуги)

- ImageUploader.vue (загрузка изображений с preview)

- ScheduleManager.vue (управление расписанием)

- PricingTable.vue (таблица тарифов)

- NotificationCenter.vue (центр уведомлений)

**Шаг 5: API endpoints (Laravel)**

Реализуй RESTful API со следующими маршрутами:

// Аутентификация

POST /api/register

POST /api/login

POST /api/logout

POST /api/password/reset

// Профили

GET /api/profiles/{id}

PUT /api/profiles/{id}

POST /api/profiles/{id}/verify

// Услуги

GET /api/services (с пагинацией и фильтрами)

POST /api/services

GET /api/services/{id}

PUT /api/services/{id}

DELETE /api/services/{id}

POST /api/services/{id}/publish

POST /api/services/{id}/promote

// Бронирования

GET /api/bookings

POST /api/bookings

PUT /api/bookings/{id}/status

DELETE /api/bookings/{id}

// Отзывы

GET /api/services/{id}/reviews

POST /api/services/{id}/reviews

PUT /api/reviews/{id}

DELETE /api/reviews/{id}

// Избранное

GET /api/favorites

POST /api/favorites/{service\_id}

DELETE /api/favorites/{service\_id}

// Чат

GET /api/chats

POST /api/chats

GET /api/chats/{id}/messages

POST /api/chats/{id}/messages

// Платежи

POST /api/payments/create

POST /api/payments/callback

GET /api/payments/history

// Поиск

GET /api/search/services

GET /api/search/suggestions

**Шаг 6: Ключевые функции реализации**

**6.1 Система поиска с Algolia/Meilisearch:**

// Настрой индексацию для модели Service

class Service extends Model

{

use Searchable;

public function toSearchableArray()

{

return [

'id' => $this->id,

'title' => $this->title,

'description' => $this->description,

'category' => $this->category->name,

'price\_from' => $this->price\_from,

'district' => $this->locations->pluck('district'),

'rating' => $this->reviews()->avg('rating'),

];

}

}

**6.2 Realtime чат на Laravel Echo + Pusher:**

// Настрой WebSocket соединение для чата

Echo.private(`chat.${chatId}`)

.listen('MessageSent', (e) => {

messages.value.push(e.message);

});

**6.3 Система уведомлений:**

// Создай уведомления для ключевых событий

class BookingConfirmed extends Notification

{

use Queueable;

public function via($notifiable)

{

return ['mail', 'database', 'broadcast'];

}

}

**Шаг 7: SEO и производительность**

1. **SEO оптимизация:**
   * Динамические meta-теги через Inertia
   * Структурированные данные (Schema.org)
   * Sitemap генератор
   * Человекопонятные URL
   * AMP версии страниц
2. **Оптимизация производительности:**
   * Lazy loading компонентов
   * Кеширование на уровне Redis
   * Оптимизация изображений (WebP)
   * CDN для статики
   * Database индексы

**Шаг 8: Безопасность**

1. **Защита данных:**
   * HTTPS везде
   * Шифрование персональных данных
   * Rate limiting для API
   * CSRF защита
   * XSS предотвращение
2. **Модерация контента:**
   * Автоматическая проверка на запрещенный контент
   * Ручная модерация новых объявлений
   * Система жалоб от пользователей
   * Блокировка нарушителей

**Шаг 9: Монетизация**

1. **Способы заработка:**
   * Платное размещение (300₽/месяц за объявление)
   * VIP-размещение (1000₽/месяц - топ выдачи)
   * Продвижение в категории (500₽/неделя)
   * Подписки для мастеров (от 1500₽/месяц)
   * Комиссия с бронирований (5-10%)
2. **Платежные системы:**
   * Интеграция с ЮKassa
   * Поддержка банковских карт
   * Электронные кошельки
   * Автоплатежи для подписок

**Шаг 10: Дополнительные фичи**

1. **Мобильное приложение (PWA)**
   * Push-уведомления
   * Офлайн-режим
   * Установка на домашний экран
2. **Аналитика для мастеров**
   * Статистика просмотров
   * Конверсия в бронирования
   * Демография клиентов
   * ROI от продвижения
3. **Программа лояльности**
   * Накопительные баллы
   * Скидки постоянным клиентам
   * Реферальная программа

**Примеры кода для начала**

**Установка проекта:**

# Создание нового Laravel проекта

composer create-project laravel/laravel spa-marketplace

cd spa-marketplace

# Установка Inertia.js

composer require inertiajs/inertia-laravel

php artisan inertia:middleware

# Установка Vue.js 3 и зависимостей

npm install @inertiajs/vue3 vue@3 @vitejs/plugin-vue

npm install pinia @vueuse/core

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

# Установка дополнительных пакетов

composer require laravel/scout

composer require laravel/cashier

composer require pusher/pusher-php-server

composer require intervention/image

**Базовая структура Vue компонента:**

<template>

<div class="service-card bg-white rounded-lg shadow-md p-4 hover:shadow-xl transition-shadow">

<div class="relative">

<img :src="service.mainImage" :alt="service.title" class="w-full h-48 object-cover rounded-md">

<span class="absolute top-2 right-2 bg-green-500 text-white px-2 py-1 rounded text-sm">

{{ service.category }}

</span>

</div>

<h3 class="mt-4 text-lg font-semibold">{{ service.title }}</h3>

<p class="text-gray-600 text-sm mt-2">{{ service.shortDescription }}</p>

<div class="mt-4 flex items-center justify-between">

<div class="flex items-center space-x-2">

<svg class="w-5 h-5 text-yellow-400" fill="currentColor" viewBox="0 0 20 20">

<path d="M9.049 2.927c.3-.921 1.603-.921 1.902 0l1.07 3.292a1 1 0 00.95.69h3.462c.969 0 1.371 1.24.588 1.81l-2.8 2.034a1 1 0 00-.364 1.118l1.07 3.292c.3.921-.755 1.688-1.54 1.118l-2.8-2.034a1 1 0 00-1.175 0l-2.8 2.034c-.784.57-1.838-.197-1.539-1.118l1.07-3.292a1 1 0 00-.364-1.118L2.98 8.72c-.783-.57-.38-1.81.588-1.81h3.461a1 1 0 00.951-.69l1.07-3.292z"/>

</svg>

<span class="text-sm font-medium">{{ service.rating }}</span>

<span class="text-sm text-gray-500">({{ service.reviewsCount }})</span>

</div>

<div class="text-right">

<p class="text-sm text-gray-500">от</p>

<p class="text-lg font-bold text-green-600">{{ formatPrice(service.priceFrom) }} ₽</p>

</div>

</div>

<div class="mt-4 flex items-center text-sm text-gray-500">

<svg class="w-4 h-4 mr-1" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M17.657 16.657L13.414 20.9a1.998 1.998 0 01-2.827 0l-4.244-4.243a8 8 0 1111.314 0z"/>

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15 11a3 3 0 11-6 0 3 3 0 016 0z"/>

</svg>

{{ service.district }}

</div>

<div class="mt-4 flex space-x-2">

<button @click="openBooking" class="flex-1 bg-green-500 text-white py-2 px-4 rounded hover:bg-green-600 transition-colors">

Записаться

</button>

<button @click="toggleFavorite" class="p-2 border border-gray-300 rounded hover:bg-gray-50">

<svg :class="['w-5 h-5', isFavorite ? 'text-red-500 fill-current' : 'text-gray-400']" fill="none" stroke="currentColor" viewBox="0 0 24 24">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4.318 6.318a4.5 4.5 0 000 6.364L12 20.364l7.682-7.682a4.5 4.5 0 00-6.364-6.364L12 7.636l-1.318-1.318a4.5 4.5 0 00-6.364 0z"/>

</svg>

</button>

</div>

</div>

</template>

<script setup>

import { ref, computed } from 'vue'

import { useServiceStore } from '@/stores/services'

import { formatPrice } from '@/utils/helpers'

const props = defineProps({

service: {

type: Object,

required: true

}

})

const serviceStore = useServiceStore()

const isFavorite = computed(() => serviceStore.isFavorite(props.service.id))

const openBooking = () => {

// Логика открытия модального окна бронирования

}

const toggleFavorite = () => {

serviceStore.toggleFavorite(props.service.id)

}

</script>

**Пример Pinia store:**

// stores/services.js

import { defineStore } from 'pinia'

import { ref, computed } from 'vue'

import axios from 'axios'

export const useServiceStore = defineStore('services', () => {

const services = ref([])

const favorites = ref([])

const loading = ref(false)

const filters = ref({

category: null,

priceFrom: null,

priceTo: null,

district: null,

rating: null

})

const filteredServices = computed(() => {

return services.value.filter(service => {

if (filters.value.category && service.category\_id !== filters.value.category) return false

if (filters.value.priceFrom && service.price\_from < filters.value.priceFrom) return false

if (filters.value.priceTo && service.price\_to > filters.value.priceTo) return false

if (filters.value.district && !service.districts.includes(filters.value.district)) return false

if (filters.value.rating && service.rating < filters.value.rating) return false

return true

})

})

const fetchServices = async (params = {}) => {

loading.value = true

try {

const response = await axios.get('/api/services', { params })

services.value = response.data.data

} catch (error) {

console.error('Error fetching services:', error)

} finally {

loading.value = false

}

}

const toggleFavorite = async (serviceId) => {

try {

if (isFavorite(serviceId)) {

await axios.delete(`/api/favorites/${serviceId}`)

favorites.value = favorites.value.filter(id => id !== serviceId)

} else {

await axios.post(`/api/favorites/${serviceId}`)

favorites.value.push(serviceId)

}

} catch (error) {

console.error('Error toggling favorite:', error)

}

}

const isFavorite = (serviceId) => {

return favorites.value.includes(serviceId)

}

return {

services,

favorites,

loading,

filters,

filteredServices,

fetchServices,

toggleFavorite,

isFavorite

}

})

**Инструкция для пошагового запуска**

1. **Начни с базовой установки Laravel и всех зависимостей**
2. **Создай структуру базы данных через миграции**
3. **Настрой аутентификацию и авторизацию**
4. **Реализуй CRUD для основных сущностей (услуги, профили)**
5. **Добавь систему поиска и фильтрации**
6. **Внедри платежную систему**
7. **Создай систему уведомлений и чат**
8. **Оптимизируй производительность**
9. **Добавь аналитику и отчеты**
10. **Проведи тестирование и деплой**

Каждый шаг должен включать:

* Детальное описание реализации
* Примеры кода
* Тесты для проверки функциональности
* Документацию для будущих разработчиков

**Важные моменты для новичков**

1. **Используй Laravel Sail для локальной разработки** - это упростит настройку окружения
2. **Следуй принципам SOLID и DRY** при написании кода
3. **Пиши тесты для критичного функционала** (PHPUnit, Jest)
4. **Используй Git с понятными коммитами** на каждом этапе
5. **Документируй API через Swagger/OpenAPI**
6. **Настрой CI/CD через GitHub Actions**
7. **Используй Laravel Telescope** для отладки в development
8. **Внедри мониторинг ошибок** через Sentry
9. **Оптимизируй запросы к БД** через Eager Loading
10. **Следи за безопасностью** - регулярно обновляй зависимости

Этот проект должен стать полноценной коммерческой платформой с возможностью масштабирования и поддержки тысяч пользователей одновременно.