Министерство образования и науки Пермского края ГБПОУ «Пермский краевой колледж «Оникс»

П(Ц)К информационных дисциплин

Дипломная работа

Разработка маркетплейса по продаже товаров

Угольников Д.О.

специальность 09.02.07

«Информационные системы и программирование»

курс 4, группа 2-ИС

форма обучения: очная

Руководитель:

Ведерникова Ирина Дмитриевна

преподаватель Пермского

краевого колледжа «Оникс»

Пермь 2025

Оглавление

[Введение 3](#_Toc199162226)

[1. Понятие маркетплейса и способы их разработки 5](#_Toc199162227)

1.1 [Определение и виды маркетплейсов 5](#_Toc199162228)

1.2 [Способы разработки маркетплейсов 6](#_Toc199162229)

1.3 [Определение технологий 6](#_Toc199162230)

1.4 [Выбор технологий и обоснование их использования 7](#_Toc199162231)

[2. Программное обеспечение для разработки веб-приложения 8](#_Toc199162232)

[3. Разработка маркетплейса 11](#_Toc199162233)

3.1 [Техническое задание 11](#_Toc199162234)

3.2 [Введение в разработку маркетплейса 11](#_Toc199162235)

3.3 [Архитектура веб-приложения 12](#_Toc199162236)

3.4 [Выбор фреймворка для стилизации 13](#_Toc199162237)

3.5 [Выбор хостинга и способа развертывания 14](#_Toc199162238)

3.6 [Решение проблемы с загрузкой изображений 15](#_Toc199162239)

3.7 [Интеграция с внешними сервисами 16](#_Toc199162240)

[4. Тестирование веб-приложения 17](#_Toc199162241)

[5. Рекомендации по сопровождению 19](#_Toc199162242)

5.1 [Алгоритм действий покупателя: 19](#_Toc199162243)

5.2 [Алгоритм действий покупателя: 19](#_Toc199162244)

5.3 [Алгоритм действий администратора: 20](#_Toc199162245)

[Список литературы 21](#_Toc199162246)

[Приложение 22](#_Toc199162247)

Введение

В последние годы электронная коммерция стремительно развивается, и маркетплейсы становятся важной частью этой трансформации. Эти платформы обеспечивают удобство для продавцов и покупателей, предлагая широкий ассортимент товаров и услуг в одном месте, что значительно упрощает процесс покупки и продажи. На фоне растущего спроса на онлайн-торговлю, создание эффективного маркетплейса становится актуальной задачей для многих компаний, стремящихся выйти на новый уровень и расширить аудиторию.

Актуальность исследования функциональных возможностей маркетплейсов обусловлена их растущей популярностью среди пользователей и предпринимателей. Идея маркетплейсов берет свое начало с первых онлайн-аукционов, таких как eBay, запущенный в 1995 году, а затем получила развитие с появлением крупных платформ, ориентированных на широкий спектр товаров и услуг. Со временем маркетплейсы стали не просто посредниками между продавцом и покупателем, а полноценными экосистемами, объединяющими логистику, финансы и аналитику, что значительно упростило ведение бизнеса.

Создание собственного маркетплейса становится особенно актуальным для компаний, стремящихся расширить аудиторию и автоматизировать процессы продаж. По данным исследований, глобальный рынок электронной коммерции продолжает расти: в 2023 году его объем превысил 6,3 трлн долларов, а маркетплейсы занимают более 60% от всех онлайн-продаж. В России доля маркетплейсов также увеличивается: например, оборот Wildberries в 2023 году вырос на 70% по сравнению с предыдущим годом. Это подтверждает высокий спрос на подобные платформы, их значительное влияние на рынок и необходимость для компаний адаптироваться к новым условиям цифровой торговли.

Современные маркетплейсы предлагают широкий спектр функциональных возможностей, включая удобные системы регистрации, интеллектуальные поисковые механизмы с фильтрацией, инструменты для управления товарами и заказами, а также интеграцию с платёжными сервисами и логистическими компаниями. Внедрение новых технологий, таких как искусственный интеллект и анализ больших данных, позволяет маркетплейсам персонализировать пользовательский опыт, прогнозировать спрос и повышать эффективность работы платформ. Кроме того, поддержка мобильных приложений и API делает маркетплейсы еще более гибкими и удобными для пользователей и бизнеса.

Исходя из этого был определен исследовательский аппарат:

Проблема: Как разработать маркетплейс?

Цель: Разработка маркетплейс-площадки для компании «ruMarket» по продаже товаров

Задачи:

* Изучить информацию о маркетплейсах, их разработке, дать определения технологиям для разработки.
* Проанализировать программное обеспечение для разработки веб-приложения.
* Разработать маркетплейс для компании «ruMarket».
* Провести тестирование готового продукта.
* Разработать рекомендации по использованию маркетплейса и сопровождению.

Готовый продукт будет сдержать:…

1. Понятие маркетплейса и способы их разработки

Определение и виды маркетплейсов

Маркетплейс – это цифровая платформа, которая объединяет продавцов и покупателей, обеспечивая удобные условия для торговли товарами и услугами. В отличие от обычных интернет-магазинов, где один продавец управляет всей витриной, маркетплейс позволяет множеству продавцов размещать свои товары, а сама платформа выступает в роли посредника, предоставляя инфраструктуру для поиска, оплаты и доставки. [Ссылка н литературу]

Существует несколько классификаций маркетплейсов:

1. По типу участников:

* B2C (business-to-consumer) – платформа связывает компании и конечных покупателей (Ozon, Яндекс.Маркет, Wildberries).
* B2B (business-to-business) – предназначена для сделок между бизнесами (Alibaba, Kompass).
* C2C (consumer-to-consumer) – позволяет пользователям продавать товары друг другу (Avito, eBay).

1. По ассортименту:

* Универсальные – включают товары разных категорий (Amazon, Ozon).
* Нишевые – ориентированы на конкретную отрасль (Lamoda – мода, Booking – аренда жилья).

1. По модели монетизации:

* Комиссия с продаж.
* Платные подписки для продавцов.
* Размещение рекламных объявлений.

Способы разработки маркетплейсов

Процесс создания маркетплейса может быть реализован разными методами в зависимости от потребностей бизнеса:

1. Готовые решения – используют платформы вроде CS-Cart, WordPress (WooCommerce), Shopify. Этот вариант подходит для небольших проектов, но имеет ограничения в кастомизации.
2. Разработка с нуля – создание уникальной платформы с использованием современных технологий, обеспечивающих гибкость, масштабируемость и безопасность.
3. Гибридный подход – комбинация готовых решений с дополнительными модулями или API-интеграцией.

В данной работе для проекта будет использован подход разработки с нуля, так как будет возможность организаций большего спектра инструментов и библиотек, чем в готовых решениях.

Определение технологий

Фреймворк — это готовая архитектурная основа для разработки программных решений, предоставляющая разработчику набор инструментов, стандартов и библиотек, упрощающих создание сложных приложений за счёт повторного использования проверенных решений и структурирования кода.

Библиотека — это набор предопределённых функций и классов, упакованных в единое решение для использования в разработке программного обеспечения. Библиотека предоставляет разработчику инструменты для выполнения специфических задач (например, обработки данных, работы с сетью или интерфейсами) без необходимости разработки этих функций с нуля.

База данных — это организованная совокупность данных, которые хранятся и управляются с помощью системы управления базами данных (СУБД). База данных обеспечивает эффективное хранение, поиск и обновление информации, а также поддерживает операции с большими объёмами данных, обеспечивая целостность и безопасность.

Выбор технологий и обоснование их использования

При создании маркетплейса использован стек современных технологий, обеспечивающих надежность, производительность и удобство разработки:

* Next.js 13+ (App Router) - современный React-фреймворк выбран как основа frontend-части:
* NextAuth.js - Система аутентификации, обеспечивающая: гибкую систему авторизации с поддержкой ролей, безопасное хранение сессий, защита маршрутов на уровне middleware.
* Prisma ORM - современная виртуальная объектная база данных.
* PostgreSQL - надежная реляционная база данных.
* Supabase - платформа для backend-as-service.
* DaData API – сервис для валидации данных.
* Axios – библиотека для HTTP запросов.
* Tailwind CSS – утилитарный CSS-фреймворк.

Данный стек технологий обеспечивает:

* Высокую производительность
* Безопасность данных
* Масштабируемость приложения
* SEO-оптимизацию

1. Программное обеспечение для разработки веб-приложения

Программное обеспечение (ПО) — программа или множество программ, используемых для управления компьютером. Инструментальное программное обеспечение — программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ. Обычно этот термин применяется для акцентирования отличия данного класса ПО от прикладного и системного программного обеспечения. [Ссылка н литературу]

Какое программное обеспечение можно использовать для разработки:

1. Visual Studio Code — это бесплатный, открытый текстовый редактор, разработанный Microsoft, с поддержкой множества языков программирования, плагинов и встроенной интеграцией с Git. Он предлагает множество расширений для различных задач, таких как отладка и форматирование кода.
2. JetBrains WebStorm — это интегрированная среда разработки (IDE) от компании JetBrains, предназначенная для разработки веб-приложений на JavaScript, HTML, CSS и других веб-технологиях, с мощными инструментами для отладки, рефакторинга и тестирования кода.
3. ​Trae IDE — это интегрированная среда разработки от ByteDance с поддержкой ИИ, предназначенная для повышения продуктивности разработчиков. Она предлагает функции редактирования кода, управления проектами, расширений и контроля версий, а также интеграцию с GitHub. Trae IDE поддерживает большинство расширений VSCode и обеспечивает плавный переход с других редакторов. ​
4. Notepad++ — это бесплатный текстовый редактор с открытым исходным кодом, поддерживающий работу с множеством языков программирования и обладающий возможностями подсветки синтаксиса, поиска и замены текста, а также расширяемый с помощью плагинов.
5. Sublime Text — это текстовый редактор с высокой производительностью, предназначенный для разработки программного обеспечения, поддерживающий множество языков программирования и обладающий возможностями расширения через плагины и настройку пользовательских параметров.

Сравнение программного обеспечения:

1. Visual Studio Code:

* Преимущества: Множество плагинов, легкость, поддержка многих языков, интеграция с Git, регулярные обновления.
* Недостатки: Ограниченные возможности для больших проектов, ограниченная отладка.

1. JetBrains WebStorm:

* Преимущества: Полная интеграция с инструментами для JavaScript, поддержка рефакторинга, мощная отладка, высокая производительность.
* Недостатки: Высокая стоимость, требует больше системных ресурсов.

1. Trae IDE:

* Преимущества: Поддерживает бесплатную интеграцию с передовыми ИИ-моделями (GPT-4o и Claude-3.5-Sonnet), поддерживает мультимодальный ввод и режим Builder для пошаговой генерации кода, а также совместима с расширениями VS Code, что делает её мощным и доступным инструментом для ускоренной разработки.
* Недостатки: Требует постоянного интернет-соединения для работы с облачными ИИ-моделями и недоступна с российских IP-адресов, что ограничивает её использование для части разработчиков.

1. Notepad++:

* Преимущества: Легковесность, множество плагинов, поддержка множества языков, бесплатность.
* Недостатки: Ограниченная функциональность для крупных проектов, нет встроенной отладки.

1. Sublime Text:

* Преимущества: Высокая производительность, поддержка множества плагинов, красивый интерфейс.
* Недостатки: Ограниченные возможности по сравнению с более специализированными редакторами, платная версия.

​Trae IDE была выбрана из-за преимущества в наличии интеллектуальных функций, таких как встроенный AI-ассистент и режим Builder, поддержку мультимодального ввода и интеграцию с GitHub, что обеспечивает более интеллектуальную и эффективную разработку.

При разработке так же необходимо использовать систему контроля версий Git, и приложение GitHub Desktop для упрощения контроля изменений.

1. Разработка маркетплейса

Техническое задание

Для разработки веб-приложения разработчику необходимо техническое задание, согласованное с заказчиком. Поскольку заказчика на данное веб-приложение на текущий момент нет, техническое задание составлялось самостоятельно. [Приложение 1]

В нём сформулированы цели проекта — создание функциональной площадки для эффективного взаимодействия продавцов и покупателей — и определён объём функциональности: управление учётными записями, добавление и редактирование карточек товаров, поиск и фильтрация, оформление заказов. Особое внимание уделено требованиям к производительности и масштабируемости, обеспечению защиты пользовательских данных, а также адаптивности интерфейса для разных устройств. Чёткая структура технического задания позволяет контролировать ход разработки и гарантировать выпуск качественного продукта без отклонений от согласованных требований.

Введение в разработку маркетплейса

На начальном этапе были изучены современные требования к разработке маркетплейсов: высокая скорость загрузки, SEO-оптимизация, безопасность аутентификации и масштабируемость. На основании анализа был выбран используемый стек технологий. Также была разработана структура маршрутов, API-ссылок и компонентов, соответствующих различным разделам сайта (каталог, карточка товара, корзина, личный кабинет, страница продавца).

В качестве названия для маркетплейса было выбрано наименование «ruMarket», в честь той страны, в которой он был разработан.

При создании дизайна маркетплейса был выбран основной цвет - глубокий оттенок синего с фиолетовым акцентом (#4438ca). Данный цвет был выбран по нескольким причинам: во-первых, синий традиционно ассоциируется с надёжностью и доверием, что критично для электронной коммерции, где пользователи совершают финансовые операции; во-вторых, лёгкий фиолетовый подтон вносит элемент креативности и современности, подчёркивает технологическую направленность проекта и наконец, этот цвет хорошо смотрится на светлом и тёмном фоне, обеспечивая контраст и читаемость интерфейсных элементов.

Архитектура веб-приложения

Во время проектирования веб-приложения для удобства и безопасности взаимодействия пользователей с платформой были выделены три основные роли:

* Гость — неавторизованный пользователь, который может просматривать товары и общую информацию о маркетплейсе, но не имеет доступа к оформлению заказов и личному кабинету.
* Авторизованный пользователь — зарегистрированный покупатель, который получает возможность оформлять заказы, сохранять избранные товары, просматривать историю покупок и управлять своими данными.
* Продавец — отдельная роль для пользователей, которым предоставлен доступ к личному кабинету продавца, с возможностью добавлять товары, управлять ассортиментом, отслеживать заказы.

Такое разделение позволило гибко настраивать права доступа и создать удобные пользовательские сценарии для каждой группы.

При разработке структуры проекта был выбран подход с использованием App Router из Next.js, который позволяет эффективно организовать маршруты и упрощает управление как страницами, так и API-ссылками. Благодаря этой архитектуре удалось добиться более чистого и гибкого кода, а также лёгкости в поддержке и масштабировании приложения. В рамках этого подхода вся клиентская часть была размещена в папке «app», а серверная логика — в папке «api». Это позволило организовать чёткое разделение ответственности и улучшить читаемость кода.

App Router обеспечил удобную маршрутизацию для различных пользовательских ролей, а также позволил эффективно управлять защищёнными и открытыми маршрутами. В папке app размещены все страницы и компоненты, которые обрабатывают клиентские запросы, взаимодействуя с сервером через API.

В свою очередь, папка api содержит всю серверную логику, включая обработку запросов для аутентификации, работы с базой данных и взаимодействия с пользователями и продавцами. Такой подход позволил интегрировать серверную логику с клиентской частью, избавив от необходимости использовать отдельные серверы или сложные конфигурации. Это ускорило процесс разработки и повысило производительность приложения.

Например, в папке «api/auth» реализована вся логика, связанная с аутентификацией и управлением сессиями через NextAuth.js, что поддерживает гибкую авторизацию с учётом различных ролей. В папке «api/products» расположены конечные точки для работы с товарами, включая их добавление, редактирование и удаление продавцами. Все запросы к серверу обрабатываются через методы «GET, POST, PUT и DELETE», что соответствует принципам RESTful-архитектуры.

Такой подход позволил не только чётко разделить ответственность между фронтендом и бэкендом, но и улучшил структуру проекта, сделав его более масштабируемым и удобным для дальнейшей разработки и поддержки.

Выбор фреймворка для стилизации

При разработке возникла необходимость использования фреймворка для стилизации. При принятии решения об использовании Tailwind CSS основополагающим фактором стала его парадигма utility-first, которая позволяет стилизовать элементы прямо в разметке, избавляя от постоянного переключения между HTML и CSS-файлами и ускоряя процесс разработки интерфейса.

Такой подход делает внесение изменений более безопасным и предсказуемым, поскольку каждая утилитная класс-инструкция влияет только на конкретный элемент, не затрагивая стили других частей приложения.

Кроме того, Tailwind CSS обеспечивает высокую консистентность дизайна: единая система утилитных классов и дизайн-токенов гарантирует, что все отступы, цвета и типографика будут строго соответствовать заданному проекту, что особенно важно для масштабируемого маркетплейса с многочисленными страницами и компонентами.

При этом обширная библиотека классов снижает объём пользовательского CSS и минимизирует необходимость в его написании, что поддерживает чистоту и читаемость кода.

Важным преимуществом стала автоматическая очистка неиспользуемых стилей при сборке (purge), благодаря чему итоговый CSS-файл содержит только необходимые правила, что существенно снижает объём передаваемых данных и ускоряет загрузку страниц.

Наличие режима JIT (Just-In-Time) дополнительно оптимизирует процесс разработки, генерируя утилиты по требованию и позволяя использовать произвольные значения прямо в классах без предварительной настройки в конфигурации.

Выбор хостинга и способа развертывания

При разработке маркетплейса ruMarket активно использовался GitHub с настройкой CI/CD для автоматического развертывания проекта на Vercel. Такой подход значительно ускорил процесс разработки, обеспечивая постоянную интеграцию и непрерывную доставку. Каждый новый коммит в репозиторий автоматически инициировал процесс сборки и развертывания, что позволило избежать ошибок при ручной загрузке и ускорить тестирование новых функций в продакшн-среде.

Использование CI/CD с GitHub и Vercel имеет несколько ключевых преимуществ. Во-первых, это автоматизация процессов сборки и развертывания, что снижает вероятность человеческой ошибки и ускоряет вывод новых изменений в продуктивную среду. Во-вторых, подход обеспечивает постоянную интеграцию, что позволяет разработчикам работать с актуальной версией кода и тестировать изменения в реальном времени. В-третьих, настройка загрузки на Vercel позволяет быстро и эффективно публиковать новые версии приложения с минимальными задержками, а также воспользоваться встроенными возможностями платформы для оптимизации производительности и масштабируемости.

Кроме того, GitHub предоставляет удобные инструменты для совместной работы в команде, такие как pull-requests, ревью кода и управление задачами, что позволяет поддерживать высокое качество кода и минимизировать количество ошибок на всех этапах разработки.

Решение проблемы с загрузкой изображений

Во время разработки маркетплейса появилась проблема с загрузкой изображений товаров: на платформе Vercel запрещено сохранять файлы на её файловой системе, поэтому прямая загрузка не работала. Чтобы обойти это ограничение, была интегрирована библиотека Supabase Storage, где организовал безопасное и масштабируемое хранение медиафайлов — теперь при загрузке картинка сразу уходит в облачное хранилище Supabase, а из интерфейса выводится полученная публичная ссылка. Таким образом удалось решить проблему с ограничениями Vercel и обеспечить удобный доступ к изображениям в приложении.

Интеграция с внешними сервисами

Для улучшения качества данных в маркетплейсе был использован сервис DaData, который предоставляет API для валидации и подгрузки информации о юридических лицах и индивидуальных предпринимателях. Это позволяло эффективно обрабатывать данные о компаниях, такие как наименование, адрес, регистрационные данные, а также проверять их корректность на основе ИНН. С помощью DaData при регистрации продавцов их данные автоматически проверялись на наличие ошибок, а также подгружалась актуальная информация по ИНН, что значительно улучшало точность данных в системе.

При разработке маркетплейса были использованы различные документации к библиотекам и фреймворкам. [Ссылка на литературу]

По причине недостатка опыта в разработке коммерческих веб-приложений в процессе разработки были посмотрены образовательные курсы по фреймворку Next.js и Prisma ORM.[Ссылка на литературу]

1. Тестирование веб-приложения

Для безопасной и структурированной разработки веб-приложения необходимо проводить тестирования продукта при его реализации.

Тестирование — это процесс проверки программного обеспечения на соответствие требованиям, с целью выявления ошибок и обеспечения стабильной работы системы[Ссылка на литературу]. Существует несколько основных подходов к тестированию программного обеспечения:

* Модульное тестирование — проверка отдельных функций или компонентов программы.
* Интеграционное тестирование — проверка взаимодействия между модулями или компонентами.
* Системное тестирование — проверка всей системы целиком на соответствие требованиям.
* Приёмочное тестирование — проверка готовности продукта к передаче заказчику.
* Функциональное тестирование — проверка, соответствует ли программное обеспечение своим функциональным требованиям.
* Регрессионное тестирование — проверка, не нарушили ли новые изменения существующий функционал.

В рамках данного проекта основное внимание было уделено функциональному и регрессионному тестированию. Такой выбор обусловлен необходимостью обеспечить корректную работу ключевых пользовательских сценариев, а также стабильность приложения при внесении изменений в кодовую базу.

В процессе разработки веб-приложения для автоматизации тестирования были написаны тесты на языке программирования Python. Основными преимуществами Python[Ссылка на литературу] являются - простота синтаксиса, высокая читаемость, обширная стандартная библиотека, кроссплатформенность.

В качестве библиотеки для тестирования была выбрана Pytest[Ссылка на литературу] за её простоту, гибкость и широкие возможности для написания читаемых и масштабируемых тестов.

В качестве ядра для автоматизированного тестирования пользовательского интерфейса был выбран Selenium[Ссылка на литературу] — инструмент, позволяющий управлять браузером и эмулировать действия пользователя. Selenium обеспечивает кроссбраузерное тестирование, поддерживает различные языки программирования (включая Python) и хорошо интегрируется с такими библиотеками, как Pytest, что делает его оптимальным выбором для тестирования веб-интерфейсов маркетплейса.

Дополнительно были реализованы сценарии, охватывающие ключевые пользовательские действия: регистрация, авторизация, добавление товаров в корзину, оформление заказа и поиск по товарам. Это позволило выявить и устранить критические ошибки на ранних этапах разработки. Тесты регулярно запускались в процессе разработки, что способствовало поддержанию стабильности и качества веб-приложения.

Также была предусмотрена возможность расширения тестов при добавлении нового функционала, что делает систему тестирования гибкой и пригодной для дальнейшего развития проекта.

По итогам проведённого тестирования удалось достичь высокой стабильности и надёжности работы ключевого функционала веб-приложения. Автоматизация позволила существенно сократить время на проверку регрессионных ошибок при внесении изменений в код. Выявленные в ходе тестирования ошибки были оперативно устранены, что повысило общее качество пользовательского опыта. Наличие системы тестирования также создало основу для безопасного масштабирования и внедрения новых функций в будущем.

1. Рекомендации по сопровождению

Алгоритм действий покупателя:

1. Зарегистрироваться или войти в аккаунт
2. Перейти в каталог или воспользоваться поиском
3. Выбрать товар и перейти в его карточку
4. Нажать «Добавить в корзину»
5. Перейти в корзину
6. Проверить состав заказа
7. Нажать «Оформить заказ»
8. Ввести контактные и адресные данные
9. Оплатить заказ
10. Отслеживать статус в личном кабинете

Алгоритм действий покупателя:

1. Зарегистрироваться или войти в аккаунт
2. Пройти верификацию как продавца
3. Заполнить контактную информацию
4. Дождаться подтверждения от администратора
5. Войти в личный кабинет
6. Перейти в раздел «Товары»
7. Нажать «Добавить товар»
8. Заполнить название, описание, цену, фотографии
9. Указать единицу измерения
10. Опубликовать товар
11. Ожидать заказ
12. Перейти в раздел «Заявки»
13. Просмотреть новый заказ
14. Отправить товар
15. Подтвердить отправку товара в заявках
16. Обновить остатки товара (при необходимости)

Алгоритм действий администратора:

1. Зарегистрироваться или войти в аккаунт администратора
2. Войти в админ-панель
3. Проверить новые заявки от продавцов
4. Подтвердить или отклонить заявки от продавцов
5. Перейти в раздел «Товары»
6. Проверить новые товары
7. Удалить или скрыть нарушающие правила
8. Перейти в раздел «Пользователи»
9. Управлять аккаунтами (удаление, изменение и т.д.)
10. Перейти в раздел «Заказы»
11. Проверить заказы
12. Обработать спорные ситуации
13. Проверить отчёты и статистику
14. Проверить наценку
15. Проверить логи
16. Сделать резервную копию базы
17. Обновить систему (если требуется)

Список литературы

1. <https://kokoc.com/blog/chto-takoe-marketpleysy-prostymi-slovami-kak-oni-rabotayut-plyusy-i-minusy-chem-otlichayutsya-ot-agr/> - Маркетплейсы что это такое простыми словами
2. <https://sky.pro/wiki/profession/chto-takoe-razrabotka-programmnogo-obespecheniya/> - Что такое разработка программного обеспечения?
3. <https://ru.wiki.rademade.com/4-typology-of-marketplaces> - Типология маркетплейсов

Приложение