

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**  
Кафедра Информатики и программирования

**РАЗРАБОТКА ПЛАТФОРМЫ ЕДИНОГО РЕЗЮМЕ**  
**МАГИСТЕРСКАЯ РАБОТА**

студента 2 курса 273 группы  
направления 02.04.03 — Математическое обеспечение и администрирование  
информационных систем  
факультета КНиИТ  
Кулакова Максима Сергеевича

Научный руководитель  
к. э. н., доцент \_\_\_\_\_ Л. В. Кабанова  
Заведующий кафедрой  
к. ф.-м. н. \_\_\_\_\_ М. В. Огнева

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ .....</b>	<b>4</b>
1 Анализ научной литературы .....	6
2 Анализ конкурентных платформ.....	11
2.1 Сервис поиска работы HeadHunter.ru .....	11
2.2 Сервис для поиска работы в ИТ-сфере Хабр Карьера.....	13
2.3 Международный сервис поиска работы Skipp.dev .....	15
2.4 Карьерный ресурс Icanchoose .....	17
2.5 Веб-сервис для хостинга ГитХаб .....	18
2.6 Блочный конструктор веб-сайтов Tilda .....	19
2.7 Онлайн-конструктор резюме Simpledoc .....	19
2.8 Онлайн-конструктор Enhancv .....	21
3 Теоретические аспекты разработки .....	24
3.1 Интерфейсная библиотека React JS .....	24
3.2 Серверная библиотека NextJS .....	25
3.3 Интерфейсная библиотека TailWind .....	26
3.4 Архитектурный стиль Rest API .....	27
3.5 Сервис тестирования API Postman.....	28
3.6 Система управления базами данных MongoDB .....	30
3.7 Платформа сборки и управления Docker .....	31
3.8 Облачная платформа Vercel .....	32
4 Основные аспекты разработки платформы единого резюме .....	34
4.1 Создание прототипа платформы резюме .....	34
4.2 Настройка рабочей среды .....	35
4.3 Общие настройки динамического приложения .....	36
4.4 Реализация клиентской части платформы .....	37
4.5 Реализация серверной части платформы .....	41
4.6 Написание механизма авторизации.....	43
4.7 Реализация страницы резюме .....	47
4.8 Взаимодействие с сервисом hh.ru .....	51
4.9 Реализация страниц сравнения и обновления резюме.....	53
4.10 Реализация страницы процесса разработки.....	55
4.11 Публикация платформы единого резюме .....	57
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>59</b>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	60
Приложение А Реализация подключения к MongoDB.....	64
Приложение Б Реализация компонента отображения опыта .....	65
Приложение В Реализация страницы резюме.....	67
Приложение Г Реализация страницы процесса разработки.....	69

## **ВВЕДЕНИЕ**

Поиск работы всегда стоит перед человеком, важно не только заработать, но и получить удовлетворение от труда. Одной из главных проблем для соискателя является поиск подходящего работодателя и представление своих навыков. Существует множество онлайн-сервисов для поиска работы, где работодатели могут размещать вакансии и рассматривать кандидатов. Некоторые платформы позволяют создавать резюме прямо на сайте соискателя. Из-за разнообразия сервисов для поиска работы работодателям может быть неудобно использовать определенную платформу, что может затруднить поиск подходящих кандидатов. Для соискателей становится сложнее ориентироваться на различных сайтах, особенно если их резюме долго висит и требует обновления. Управление несколькими резюме на разных платформах становится все более сложным, что затрудняет процесс поиска работы. Для решения данной проблемы необходимо прибегнуть к разработке абсолютно уникальной платформы, при помощи которой контроль актуальности своего резюме не будет являться рутинной проблемой. Самый актуальный и удобный способ – разработка специализированного веб-приложения.

В свою очередь, современная веб-разработка требует эффективных и масштабируемых решений, таких как клиент-серверная архитектура. Эта модель не только управляет данными эффективно, но и обеспечивает бесперебойную работу для множества пользователей. Использование инструментов, таких как ReactJS для динамических интерфейсов, Rest API для стандартизированного подхода к веб-разработке и TailWind CSS для оптимизированного стиля, помогает создавать современные веб-приложения. Разработчики могут обеспечить масштабируемость и совместимость своих приложений, следуя принципам REST.

Целью научно-исследовательской работы является разработка и масштабирование платформы единого резюме с функционалом автоматического обновления данных на различных сервисах поиска работы до клиент-серверного приложения. Для достижения поставленной цели предусмотрены следующие задачи исследования:

1. Провести обзор уже существующих решений платформ создания и размещения резюме в интернете с целью выявления существующих подходов и методов;

2. Рассмотреть и проанализировать существующие платформы для создания резюме с учетом их функционала, архитектуры и пользовательского опыта;
3. Сформулировать собственные методы разработки единой платформы резюме, опираясь на литературный обзор и анализ существующих решений;
4. Разработать и настроить клиентскую и серверную части платформы, выбрав оптимальные технические средства;
5. Реализовать механизм взаимодействия платформы с различными сервисами поиска работы, обеспечивающий автоматическое обновление резюме на этих платформах;
6. Подвести итоги выполненной научно-исследовательской работы, сделать выводы о полученных результатах и предложить рекомендации для дальнейших разработок в данной области.

## **1 Анализ научной литературы**

Рассматриваемая литература будет затрагивать тему аспектов составления резюме, принципы их составления и критерии, по которым работодателю с наибольшей вероятностью понравится грамотно составленное резюме. После проведения анализа данной темы нам предоставится возможность выделить основные пункты, которые будут учитываться при разработке собственной единой платформе резюме.

Для начала стоит рассмотреть научные статьи, связанные с доказательством важности правильного составления резюме в настоящее время, и какие изменения оно претерпевает. В статье К.В. Косолаповой «Типологические особенности современного резюме на английском языке» автор выделяет основные пункты в резюме, которые было принято считать достаточными:

1. Полные ФИО;
2. Возраст;
3. Место проживания на текущий момент;
4. Место учёбы, уровень образования;
5. Список умений;
6. Опыт работы (при его наличии);
7. Контактные данные.

Автор также подчёркивает, что резюме является «визитной карточкой» для работодателя, и в зависимости от того, как оно составлено, будет зависеть решение о приглашении соискателя на собеседование. В статье раскрывается современное понятие «резюме», а также отмечается, что постепенное развитие и становление резюме определило его типологическое разнообразие. Так, традиционными типами резюме принято считать хронологическое (Chronological CV), функциональное (Functional CV) и комбинированное (Combination CV). После обзора на каждый из типов резюме автор рекомендует придерживаться следующим пунктам при составлении документа:

1. Личные сведения в начале резюме (имя, адрес, контактные телефоны, адрес электронной почты);
2. Краткое описание себя от первого или третьего лица в виде небольшого параграфа;
3. Список ключевых навыков, умений соискателя, а также занимаемых ранее должностей;

4. Опыт работы, профессиональные успехи, поставленные цели;
5. Образование, курсы, стажировки, академические степени, квалификации, членство в профессиональных организациях;
6. Дата рождения, пол, наличие водительского удостоверения;
7. Хобби и интересы.

Также, опираясь на зарубежные исследования, автор упоминает ряд рекомендаций по составлению резюме таким образом, чтобы работодатель с наибольшей вероятностью выбрал именно Вас на желаемую Вами вакансию [1].

Что касается сохранения единого резюме для вакансий, однообразие может конфликтовать с желанием соискателя попробовать себя в альтернативной сфере. Так автор Е.А.Шинкаренко в статье «Резюме как элемент практик поиска работы молодежью» в своей статье описывает эволюцию популярных вакансий, по которым чаще откликаются молодые люди возрастом 18-25 лет. Несмотря на смену интересов современной молодежи, в последнее время работодатели ставят более строгие критерии на свои вакансии, вследствие чего будущий соискатель должен будет обладать (в большинстве случаев) уровнем образования не ниже незаконченного высшего. В своём выводе автор статьи подчеркивает, что стремление молодых соискателей в профессиональные сферы при составлении нескольких резюме можно рассматривать как стратегию адаптации к разнообразию вакансий на рынке труда, применение образовательных компетенций к разным типам трудовой деятельности, разному содержанию труда, но то же время это делает рынок труда и профессий очень дифференцированным, что порождает стремление попробовать себя в разных его сегментах [2]. Так для выпускника высшего образовательного учреждения одно из резюме может быть попыткой реализовать диплом, но при этом другое резюме может быть составлено абсолютно по другой траектории развития (например, творческая специальность, связанная с хобби, вместо технической). Это можно рассматривать как желание занять максимально выгодные позиции, так и с точки зрения снижения рисков в условиях трансформации рынка труда и появления новых профессиональных сфер [3].

Рассматривая реализацию технической части будущего продукта, становится необходимым изучить вопрос хранения кода на сторонних сервисах, а также принципы его написания в качестве самостоятельного веб-сервиса. В статье С.А. Резина «Что должен знать начинающий Node.js разработчик» отмечается,

что наиболее популярный фреймворк Node.js обладает такой репутацией благодаря своей гибкости, лёгкости и высокой производительности, автор предлагает воспользоваться веб-фреймворками, которые обрабатывают запросы, пришедшие по протоколу HTTP, для работы по созданию собственного веб-сервиса [4]. Однако в статье автора также подчёркивается, что знания кодовой базы Node.js не будет достаточным для запуска веб-приложения, для чего необходимо дополнительно изучить возможности реализации интерфейсной части (при помощи JavaScript, HTML и CSS), хранения динамических данных непосредственно в базе данных и развертывания проекта на облачной платформе, например, Netlify, с последующим его запуском [5].

В статье В.М. Ефремова «Разработка веб-сайта с использованием Bootstrap» автор описывает преимущества библиотеки Bootstrap, подтверждая их на примере процесса создания веб-сайта и отмечая критерии качества, такие как адаптивность, кроссбраузерность и быстрота загрузки. Библиотеку Bootstrap создавали для того, чтобы уменьшить недостатки блочного отображения и свести использование альтернативных путей к минимально возможному. Также автор статьи демонстрирует преимущества использования данной библиотеки для веб-разработки, такие как особое внимание на работу с мобильными девайсами и большими возможностями адаптации сайта, кроссбраузерность, большой пакет внутренних CSS-классов, встроенные JS-функции, а также наличие различной документации на русском и английском языках [6]. Из недостатков применения Bootstrap автор указывает на размеры файлов в библиотеке, что влияет на скорость работы сайта, и блочный способ отображения, который не является гибким для воплощения веб-дизайнерских идей [7].

Для более комфорtnого взаимодействия сервера с клиентом в статье П.А. Безрука «Разработка системы распределенного мониторинга компьютерной сети на основе rest API» подчёркивается возможность использования REST API в качестве удобного архитектурного стиля [8]. В статье упоминается принцип работы REST API, а также рассматриваются решения для пользовательского интерфейса системы распределенного мониторинга, такие как SPA (Single Page Application) и фреймворк AngularJS, известный своим обширным функционалом [9] [10].

В статье М.А. Гарина, Д.К. Егоровой и С.Ф. Сайфетдинова «Создание Workflow аналитической платформы KNIME для анализа данных на примере

вакансий сайта HeadHunter» рассматривается процесс разработки кроссплатформенного приложения, написанного на языке Java с открытым исходным кодом для анализа данных. Работа приложения рассматривается на примере некоторых типов вакансий, размещенных на сервисе HeadHunter, с пошаговым наблюдением всех этапов и описания процессов и инструментов, применяемых на данном шаге [11]. Авторы статьи отмечают наличие открытого API на используемом ими сервисе в качестве примера [12].

Для хранения кода проекта часто приходится прибегать к специализирующимся на этом сервисах, о чём авторы Н.А. Грузин и А.А. Голубничий в статье «Обзор и сравнение хостингов для git-репозиториев: Bitbucket, Github и Gitlab» рассказывают в виде сравнения трёх основных хостингов: BitBucket, GitHub и GitLab. В статье упоминается, что использование репозиторий позволяет отследить ошибки, последние изменения и позволяет управлять релизами, списками рассылки и проектной документацией [13]. В качестве веб-сервиса хостинга авторы статей рассматривают в первую очередь Bitbucket, преимущества использования которым заключается в наличии бесплатных тарифных планов аккаунта, который поддерживает неограниченное количество приватных репозиториев, а также поддержка таких функций, как проверки слияния, поиск кода, Git Large File Storage, дополнения и интеграции, использование REST API для создания сторонних приложений, которые могут использовать любой язык разработки, сниппеты и умное зеркалирование [14]. В случае с GitHub авторы статьи отмечают, что основной функционал доступен пользователям бесплатно, пока как продвинутые профессиональные и корпоративные являются коммерческими. Несмотря на это, подчеркивается, что основной целью GitHub является облегчение контроля версий и аспектов отслеживания проблем при разработке программного обеспечения. В качестве преимущественных функций GitHub авторы отмечают документацию, включая автоматически визуализированные файлы, README в различных форматах файлов, подобных Markdown, GitHub Actions, различные диаграммы с участниками, коммитами, частотой кода, возможность подписать кого-то на уведомления, упомянув их, возможность использования эмодзи, программа просмотра PDF-документов, а также оповещения системы безопасности об известных распространенных уязвимостях и уязвимостях в разных пакетах [15]. Рассматривая, инструмент GitLab, авторы статьи отмечают его развитие от решения для управления исходным кодом

до комплексного решения разработки программного обеспечения. В качестве ключевых различий между тремя платформами авторы статьи выделяют виды поддержки репозиториев с открытым исходным кодом в платформе, импорт самих репозиториев, дистрибуцию проектов, а также приводят сравнительную таблицу хостингов для git-репозиториев [16].

## **2 Анализ конкурентных платформ**

Существуют решения, предлагающие создание резюме на своей платформе с различными плюсами и минусами. Рассмотрим самые популярные и востребованные решения.

### **2.1 Сервис поиска работы HeadHunter.ru**

HeadHunter является одним из крупнейших сервисов по поиску работы и сотрудников в России и по всему миру. Каждый месяц на сайте обрабатывается свыше сотни тысяч вакансий, и ещё большее количество людей имеют возможность найти работу мечты [17].

Со стороны соискателей алгоритм отклика на вакансию выглядит следующим образом:

1. Зайти в свой аккаунт на hh.ru;
2. Найти кнопку «Создать резюме»;
3. Заполнить пункт с контактными данными. Структура пункта представлена на рисунке 2.1.

**Контактные данные**

The screenshot shows a user interface for entering contact details. It consists of four input fields arranged vertically. The first field is labeled 'Имя' (Name) with a placeholder 'Имя'. The second field is labeled 'Фамилия' (Surname) with a placeholder 'Фамилия'. The third field is labeled 'Мобильный телефон' (Mobile phone) with a placeholder '+7' and a small '+' icon. The fourth field is labeled 'Город проживания' (City of residence) with a placeholder and a three-dot ellipsis icon.

Рисунок 2.1 – Структура пункта контактных данных

4. Заполнить пункты основной информации. Структура пункта представлена на рисунке 2.2.
5. Указать желаемую специальность и заработную плату. Структура пункта представлена на рисунке 2.3.
6. Указать опыт работы (при его наличии), а также навыки, которые предлагаются пользователю в качестве отдельных ключевых слов;

## Основная информация

Дата рождения	День	Месяц	Год
Пол	<input type="radio"/> Мужской		
	<input type="radio"/> Женский		
Гражданство	<input type="text"/> <span>≡</span>		
	Россия		
Опыт работы	<input type="radio"/> Есть опыт работы		
	<input type="radio"/> Нет опыта работы		

Рисунок 2.2 – Структура пункта основной информации

## Специальность

Желаемая должность	<input type="text"/>	
Зарплата	<input type="text"/>	руб. <span>▼</span> на руки

Рисунок 2.3 – Структура пункта специальности

7. Указать уровень образования, место его получения и года выпуска (либо «по настоящее время» для школьников или студентов);
8. Указать владение языками и его уровень (для иностранных). Структура пункта представлена на рисунке 2.4.

## Владение языками

Родной язык	Русский <span>▼</span>	
Иностранные языки	Английский <span>▼</span>	C2 — В совершенстве <span>▼</span>
	Немецкий <span>▼</span>	B1 — Средний <span>▼</span>
<a href="#">Указать ещё один язык</a>		

Рисунок 2.4 – Структура пункта владения языками

9. Пункт «другой важной информации» содержит в себе сведения о готовности к переезду, желаемой занятости, графика работы, наличии автомобиля

и водительских прав, а также категорий в них. Для иностранных граждан присутствует пункт «разрешения на работу».

После публикации резюме его может быть не видно большинству работодателей, если некоторые из пунктов являются незаполненными. Сам алгоритм составления резюме не является сложным, а возможность откликнуться на вакансию часто подразумевает прикрепление сопроводительного письма помимо самой «визитки» соискателя. Дополнительно сервис hh.ru предлагает услуги экспертов для составления грамотного резюме за небольшую плату (от 3 до 8 тысяч рублей в зависимости от разновидности услуги).

## **2.2 Сервис для поиска работы в IT-сфере Хабр Карьера**

Являясь одним из популярных в России коллективным IT-блогом, Хабр смог развить не только форум для программистов, но и отдельные сервисы, которые связаны с помощью начинающих и опытных разработчиков, тестировщиков, дизайнеров и прочих информационных вакансий. Одним из подобных сервисов для поддержки начинающих и опытных IT-специалистов является Хабр Карьера [18].

В отличие от hh.ru, Хабр Карьера публикует вакансии исключительно связанные с IT сферой. На сервисе предлагаются вакансии как небольших компаний, так и компаний-гигантов (например, Яндекс, Авито, Mail.ru). Для своих коллег и знакомых на Хабр Карьера пользователю предоставляется возможность оставить профессиональную рекомендацию.

Составление резюме на сайте начинает свой путь с процесса авторизации на сервисе. Это можно сделать как при помощи стандартной регистрации с подтверждением почты, так и через сервисы, доступные в России (Вконтакте, Google Account).

Составить своё резюме Хабр Карьера предлагает сразу же после авторизации, причём существует возможность импортировать резюме с сервиса hh.ru. Данное окно представлено на рисунке 2.5.

Для ручного ввода или создания первого резюме пользователю потребуется 3 шага:

1. На первом указывается фамилия и имя, пол и дата рождения, а также основная цель регистрации на Хабр Карьера. Структура пункта представлена на рисунке 2.6. В той же вкладке (если выбрана роль соискателя) необходимо выбрать основную специализацию и отдельный профиль, а

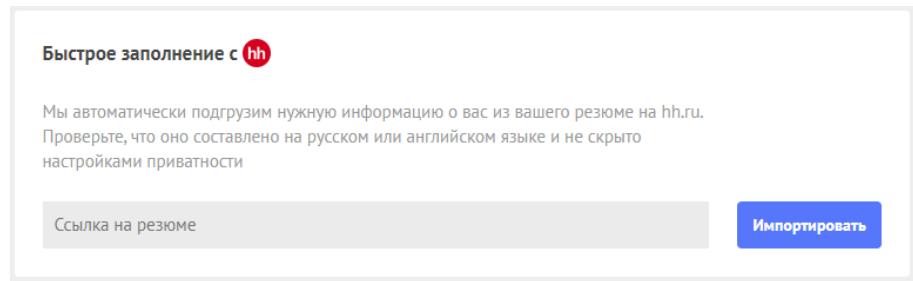


Рисунок 2.5 – Предложение импорта резюме с hh.ru

A screenshot of a registration form on 'Хабр Карьера'. The top section is titled 'Давайте знакомиться!' with fields for 'Имя \*' (Name) and 'Фамилия \*' (Surname). Below these are gender selection ('Пол') with options 'Мужской' (Male) and 'Женский' (Female), and a date of birth field ('Дата рождения'). The next section is titled 'Основная цель \*' (Main goal) with a note: 'Выберите вашу цель на Хабр Карьере. Если со временем она изменится – не страшно! Сейчас это поможет нам понять, какой мини-курс в письмах вам отправить: для соискателей или для эйчаров и рекрутеров.' It includes two radio buttons: 'Я ищу работу' (I am looking for work) and 'Я нанимаю сотрудников' (I am hiring staff).

Рисунок 2.6 – Первый шаг регистрации

также квалификация (Intern, Junior, Middle, Senior, Lead). В отличие от hh.ru и в связи с ограниченной сферой деятельности, все специализации представлены наглядно и распределены по категориям для удобства выбора. Дополнительно указываются профессиональные навыки, которым владеет соискатель. Список таких навыков очень обширен и позволяет гибко найти нужные профессиональные «скиллы». Часть из них предлагается уже на старте как «Самые популярные». Структура пункта представлена на рисунке 2.7.

2. Вторая вкладка будет содержать в себе контактную информацию, что также важно при составлении резюме, ведь одних данных об имени и фамилии будет недостаточно для связи с соискателем. По содержанию оно аналогично сервису hh.ru, например, указать город проживания и пункт о готовности к переезду или удаленной работе. Однако для связи соискатель может оставить не только номер телефона или ссылку-портфолио, но и свой логин в мессенджерах, например, в Telegram. Структура пункта представлена на рисунке 2.8.

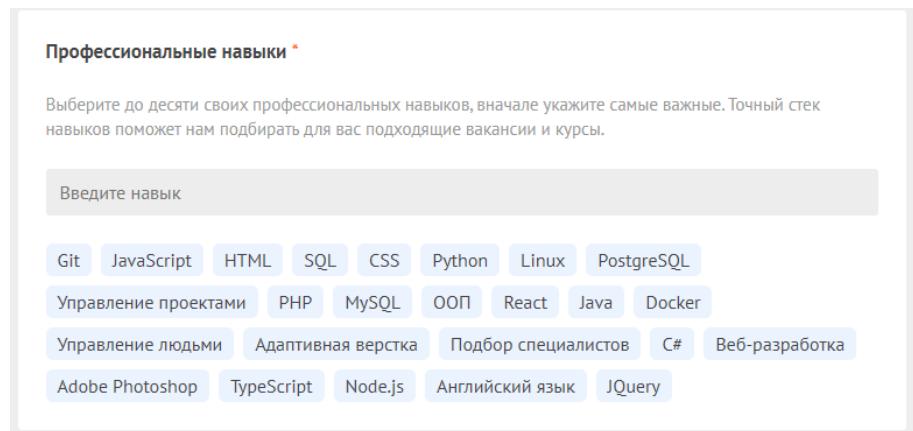


Рисунок 2.7 – Выбор профессиональных навыков

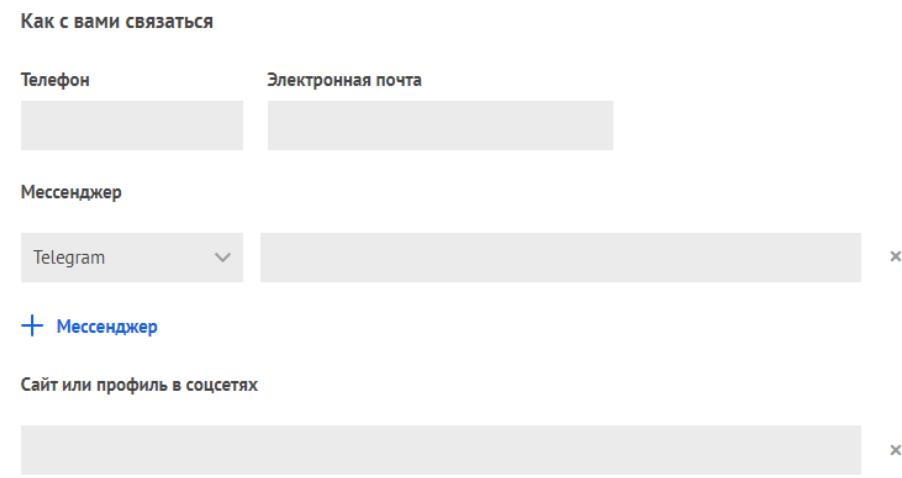


Рисунок 2.8 – Выбор контактных данных

3. Последняя вкладка содержит в себе пункты, связанные с опытом работы. По статистике, большинство работодателей присматриваются к кандидатам, у которых за спиной есть даже самый незначительный, но указанный в резюме опыт. Однако на Хабр Карьера отсутствует пункт, связанный с «фрилансом», что мешает заполнению опыта работы в данной сфере, но его можно добавить самостоятельно.

### 2.3 Международный сервис поиска работы Skipp.dev

Skipp.dev является международным сервисом как для поиска работы, так и для найма работников со стороны работодателей. Различие между предыдущими сервисами отличается не только в плане дизайна, но и порядка заполнения пунктов. Для того, чтобы начать поиск желаемой вакансии, необходимо также пройти авторизацию, и после этого поочередно заполнить пункты специализации и навыков:

1. Необходимо верифицироваться по номеру мобильного телефона. Этот пункт необходим для того, чтобы проверить подлинность аккаунта, а также для дальнейшей авторизации на сервисе при помощи смс-кода.
2. На первом шаге соискателю предлагается выбрать свои навыки. Структура пункта представлена на рисунке 2.9.

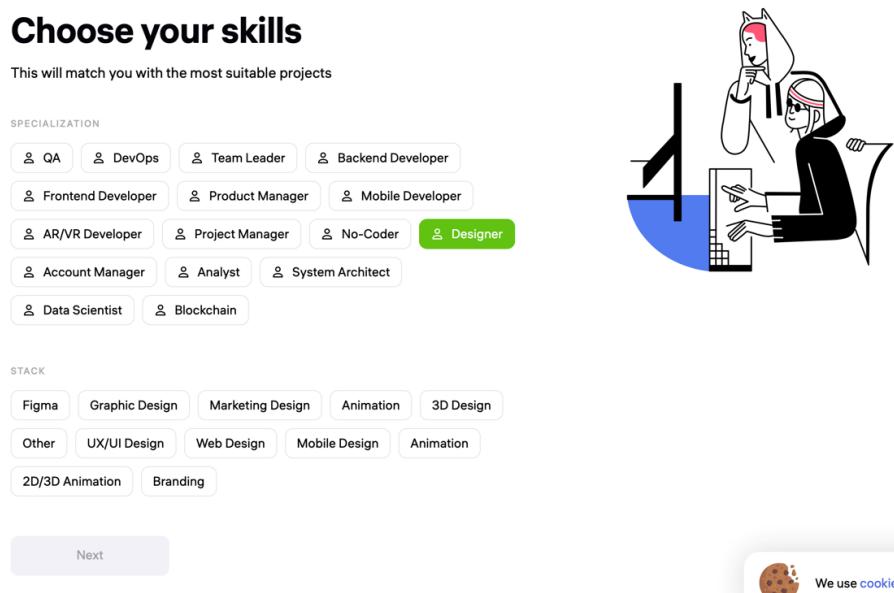


Рисунок 2.9 – Выбор навыков

3. После выбора навыков необходимо указать уровень навыков, которые были выбраны на предыдущей странице, что является нестандартным для отечественных сервисов. Структура пункта представлена на рисунке 2.10.

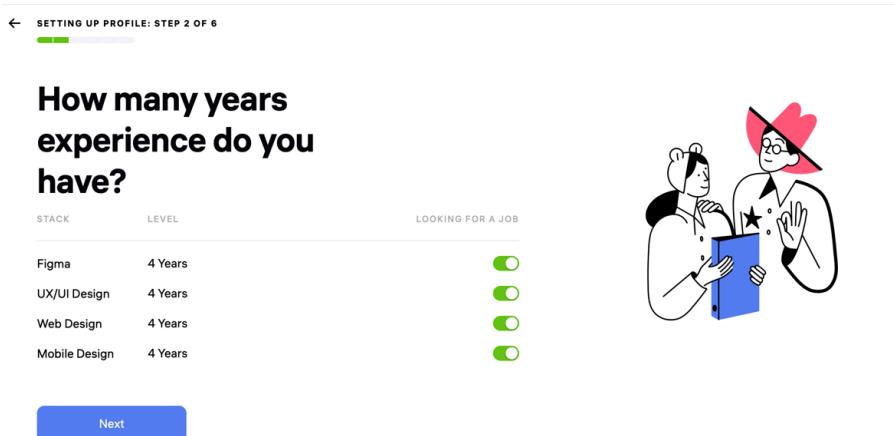


Рисунок 2.10 – Выбор уровня навыков

4. Заполнение опыта работы не сильно отличается от рассмотренных нами предыдущих платформ, но имеет свои преимущества. Например, для бо-

лее подробного описания должностных обязанностей на предыдущих местах работы пользователю предлагаются различные виды деятельности, также начиная с самых популярных на сайте. Также к опыту работы есть возможность приложить изображения в качестве портфолио. Структура пункта представлена на рисунке 2.11.

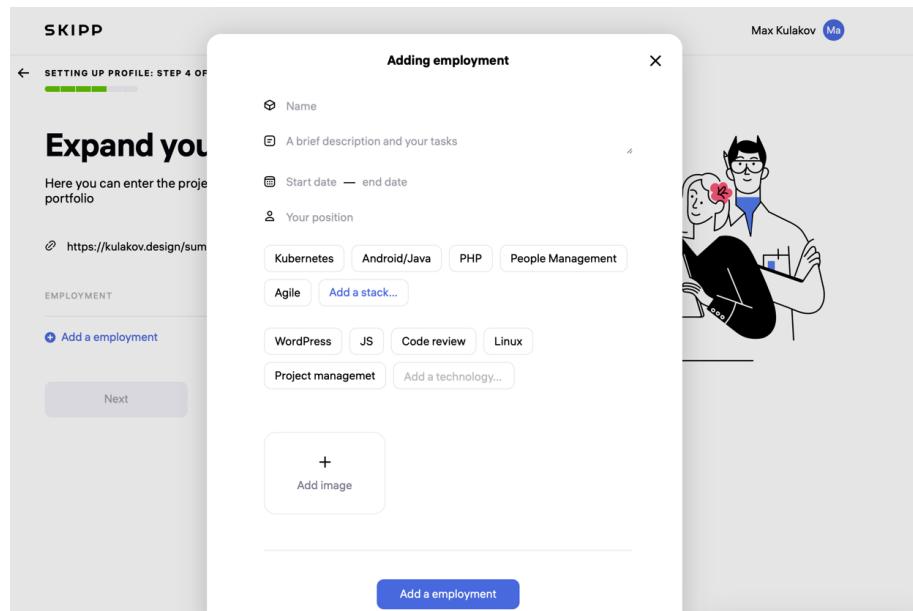


Рисунок 2.11 – Выбор опыта работы

5. Последний пункт содержит в себе форму заполнения контактной информации, начиная от ФИО до номера телефона и e-mail.

Из положительных сторон данного сервиса можно выделить приятный внешний вид и неторопливое пошаговое заполнение всех пунктов. Акцент делается на опыте работы, чтобы сконцентрировать внимание пользователя на том, как будет лучше преподнести себя будущим работодателям. Однако сервис не захватывает резюме с других платформ, что делает эту платформу абстрагированной от других себе подобных.

## 2.4 Карьерный ресурс Icanchoose

ICanChoose рекомендует себя в качестве карьерного ресурса нового формата. Из новизны, предполагаемо, сервис предлагает помочь в поиске работы, а также предлагает карьерные советы.

Авторизация на сервисе доступна при помощи ВКонтакте, а так же с email. При создании резюме сразу предлагается импорт из сервиса hh.ru, представленный на рисунке 2.12.

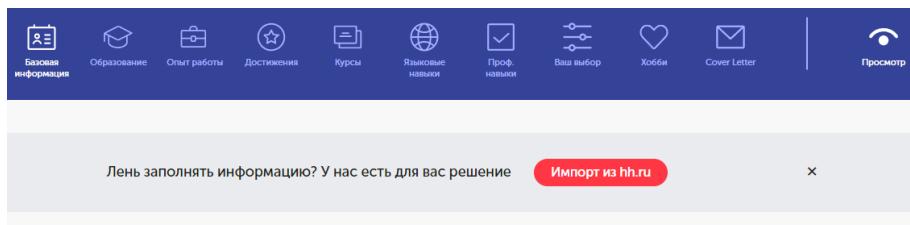


Рисунок 2.12 – Возможность импорта резюме

Процесс составления резюме происходит так же, как на вышерассмотренных сервисах, но с добавлением пунктов о хобби, достижениях и сопроводительное письмо. Также на сервисе существует возможность предварительного просмотра готового резюме, которое работает только при полном заполнении основной информации.

## 2.5 Веб-сервис для хостинга ГитХаб

ГитХаб зарекомендовал себя крупнейшим сервисом для хостинга с возможностью их совместной разработки, но, благодаря своим инструментам, на платформе возможно создание резюме. На аккаунте начинающего разработчика такое решение будет являться хорошим продолжением стратегии его развития, так как при рассмотрении профиля в качестве портфолио работодателю будут видны все разработанные проекты, на основе чего высока вероятность получить приглашение самому разработчику.

Для начала работы с составлением резюме пользователю необходимо создать новый репозиторий с названием, которое будет повторять «юзернейм» на GitHub. Сервис подчеркнёт его в качестве уникального и захватит его нужным образом.

Вся информация по резюме будет находиться в файле README.md. Другими словами, всё написанное и отформатированное будет видно на странице в GitHub и будет служить в дальнейшем красочной визиткой для тех, кто будет просматривать профиль разработчика [19].

Удобство написания резюме в GitHub подкрепляется отсутствием шаблона и полной свободой мысли, однако из-за отсутствия точных критериев содержания информации необходимо для себя составлять структуру будущей визитки, чтобы она выглядело не только гармонично, но и понятно.

## 2.6 Блочный конструктор веб-сайтов Tilda

Tilda изначально является конструктором, позволяющим на внутренних шаблонах создать любой сайт, который удовлетворяет задаче, в том числе резюме. Для последнего на сайте есть готовые шаблоны, которые упрощают работу с сервисом, в частности для начинающих пользователей [20]. Визуально это представлено на рисунке 2.13.

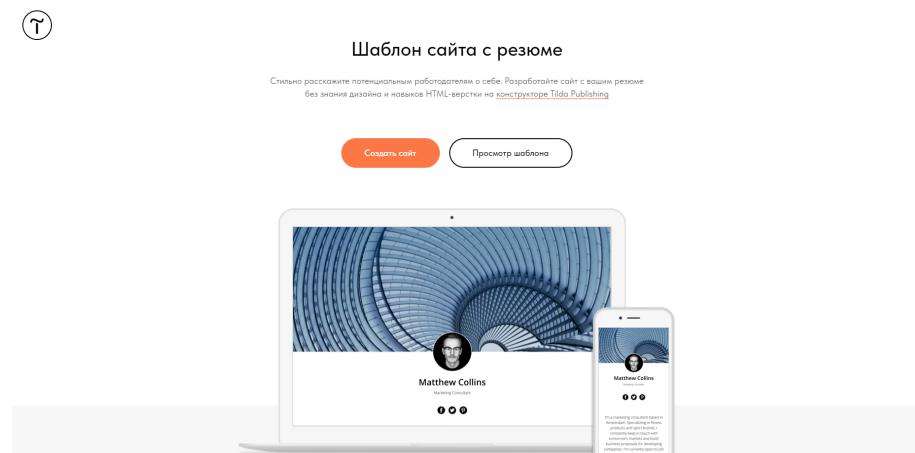


Рисунок 2.13 – Предложение использования шаблона

Составленный из блоков шаблон также можно будет использовать в качестве собственного лендинга.

Профиль на Тильде также является портфолио, что позволяет работодателю, желающему создать сайт на данном сервисе, найти необходимого ему разработчика при помощи системы заказов на Tilda Express.

Резюме на Tilda Express представляет из себя небольшую страницу с размещенными на них проектами, разработанными на Tilda, а также окошком с заполнением заявки на обратную связь при наличии желания заказать у данного разработчика собственный сайт. Пример резюме представлен на рисунке 2.14.

## 2.7 Онлайн-конструктор резюме Simpledoc

SimpleDoc является сервисом исключительно по составлению резюме. На сайте предлагается составление резюме в трёх основных этапах: заполнение формы, выбор дизайна для резюме и его скачивание или отправка по e-mail, представленных на рисунке 2.15.

Сам процесс заполнения формы занимает около 10ти минут и содержит в себе стандартные для резюме пункты:

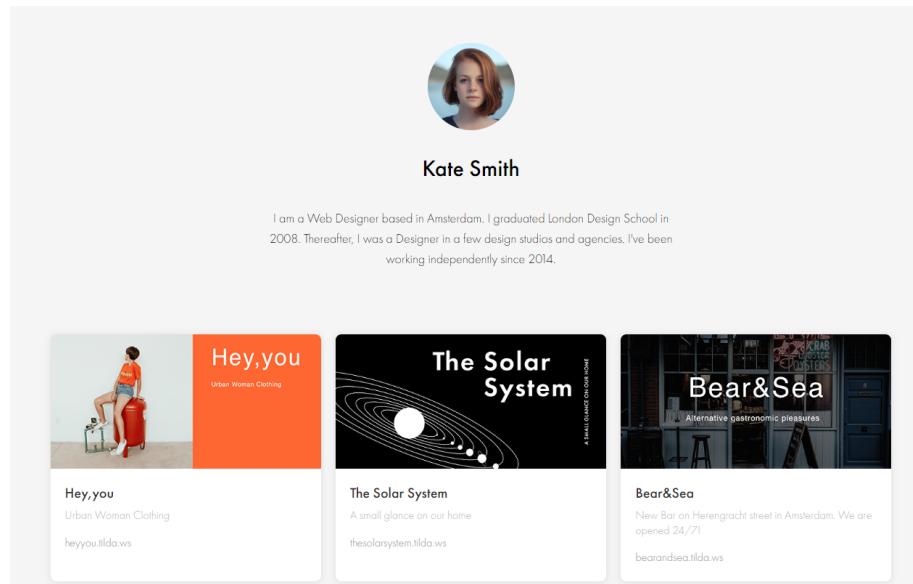


Рисунок 2.14 – Пример резюме на платформе Tilda

### Составьте эффективное резюме

Для этого достаточно выполнить три простых действия:

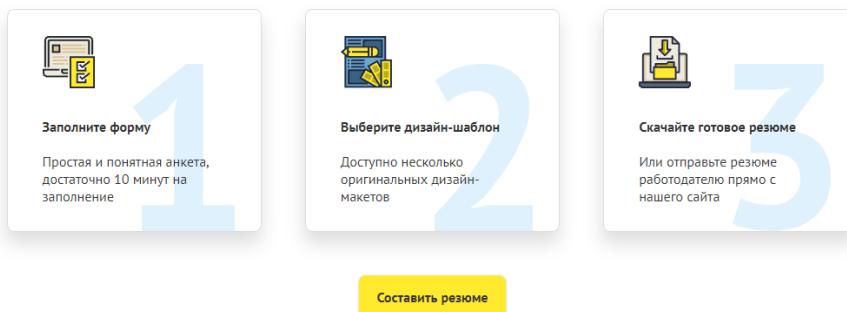


Рисунок 2.15 – Этапы создания резюме

1. Основная информация;
2. Личная информация;
3. Опыт работы;
4. Образование;
5. Курсы и тренинги;
6. Иностранные языки и компьютерные навыки;
7. Дополнительная информация.

После заполнения пунктов сервис предлагает выбрать один из четырёх шаблонов для дальнейшего скачивания, представленных на рисунке 2.16.

После составления резюме по шаблону пользователь сможет скачать его и редактировать в течение месяца после разовой оплаты на самом сервисе. Сервис также не позволяет импортировать резюме с других сервисов.

Заполните поля выделенные желтым маркером

Консервативный	Классический	Современный	Прогрессивный
<div style="border: 1px solid #ccc; width: 100px; height: 100px; margin-bottom: 10px;"></div> <div style="background-color: #f0e68c; padding: 2px 10px; border-radius: 5px; display: inline-block;">Фамилия Имя Отчество</div>			
<b>Личная информация</b> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>Гражданство:</span> <span style="background-color: #f0e68c; padding: 2px 10px; border-radius: 5px; display: inline-block;">не указано</span> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>Образование:</span> Высшее         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>Дата рождения:</span> 8 января 2005 (18 лет)         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>Пол:</span> Мужской         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>Семейное положение:</span> Холост         </div>		<b>Желаемая зарплата</b> не указано <input checked="" type="checkbox"/> Полная занятость <input checked="" type="checkbox"/> Полный день	
<b>Опыт работы</b> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span style="background-color: #f0e68c; padding: 2px 10px; border-radius: 5px; display: inline-block;">не указано</span> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span style="background-color: #f0e68c; padding: 2px 10px; border-radius: 5px; display: inline-block;">не указано</span> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>январь 2022 - январь 2023 (1 год)</span> </div>		<b>Контакты</b> <div style="background-color: #f0e68c; padding: 2px 10px; border-radius: 5px; display: inline-block;">не указано</div> <div style="background-color: #f0e68c; padding: 2px 10px; border-radius: 5px; display: inline-block;">не указано</div>	
<b>Образование</b> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span style="background-color: #f0e68c; padding: 2px 10px; border-radius: 5px; display: inline-block;">не указано</span> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span style="background-color: #f0e68c; padding: 2px 10px; border-radius: 5px; display: inline-block;">не указано</span> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>2023, очная форма обучения</span> </div>		<b>Компьютерные навыки</b> <input checked="" type="checkbox"/> Печать, сканирование, копирование документов <input checked="" type="checkbox"/> Интернет <input checked="" type="checkbox"/> Электронная почта	
<b>Дополнительная информация</b> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>Наличие водительских прав (категории):</span> В, ВЕ         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <span>Личные качества:</span>            Отсутствие вредных привычек, энергичность, инициативность, самостоятельность, ответственность, коммуникабельность, быстрая обучаемость         </div>			

Рисунок 2.16 – Вариант шаблона резюме

## 2.8 Онлайн-конструктор Enhancv

Enhancv также является сервисов для составления резюме, но в отличие от SimpleDoc имеет приятный дизайн, представленный на рисунке 2.17 уже со стартовой страницы:

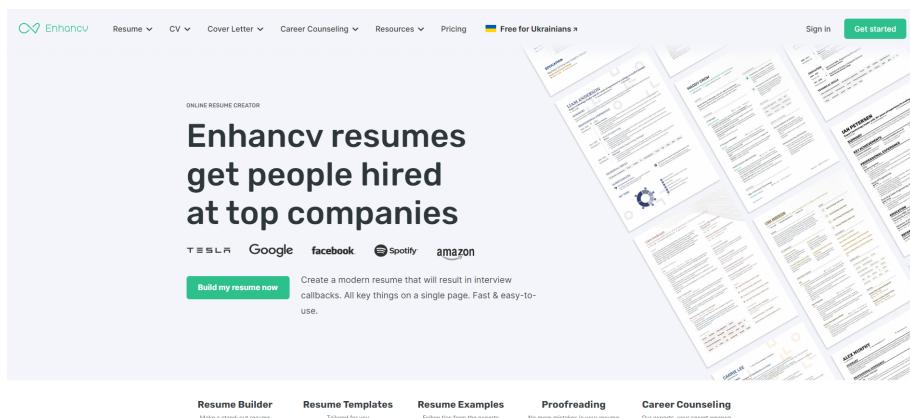


Рисунок 2.17 – Внешний вид сервиса Enhancv

При начале работы нам помогает «виртуальный» помощник, показанный на рисунке 2.18, с которым мы заполняем пункты по очереди:

1. Имя Фамилия
2. Наличие опыта в создании резюме (для возможности его импортировать)
3. Готовые шаблоны для будущего резюме.

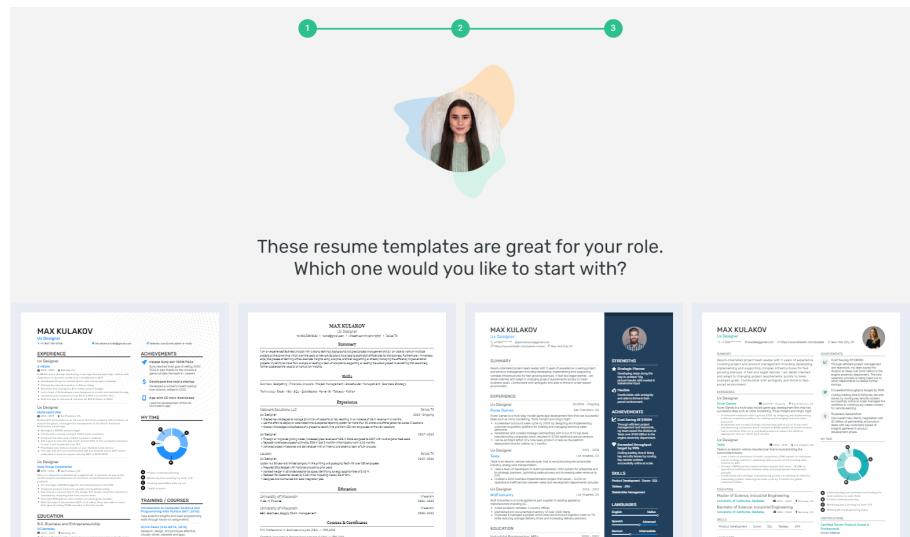


Рисунок 2.18 – Помощник в составлении резюме

После чего предлагается отредактировать информацию в резюме напрямую в одном из шаблонов, показанном на рисунке 2.19:

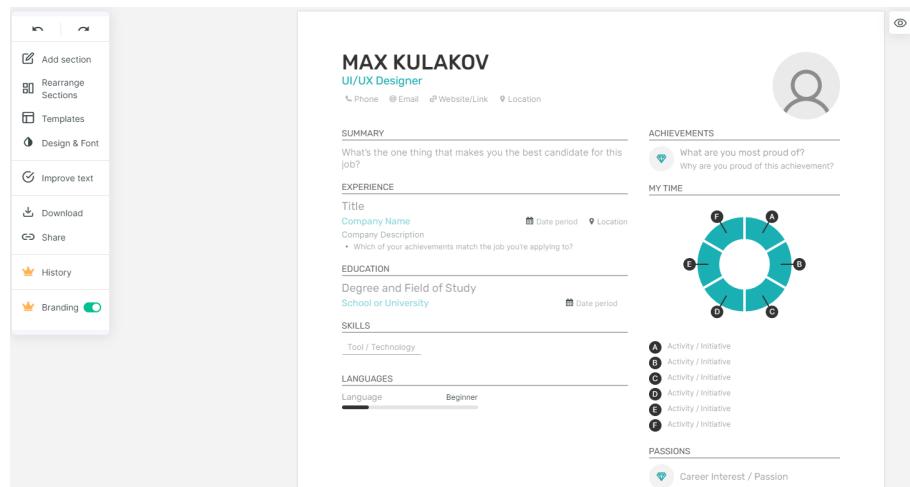


Рисунок 2.19 – Внешний вид шаблона резюме

На выбор предлагается множество инструментов для комфортной работы с шаблоном: от возможности редактирования шрифта до редактируемой инфографики. Также является платным сервисом.

На основе просмотренных конкурентных платформ можно вынести их основные недостатки:

1. Не все сервисы предлагают импорт резюме из других платформ;
2. Сервисы, работающие исключительно на составление резюме, берут деньги за свои услуги;
3. Некоторые сервисы при указании опыта работы требуют официальное название компании или должности, даже если они не зарегистрированы, и не всегда удается указать нужное название;
4. В дополнение к пункту 3, не везде возможно указание опыта «свободных заказов» вне сервисов по типу «Фриланс.ру»
5. Для русскоязычного сегмента сервисов поиска работы основным является именно HeadHunter, импорт резюме с которого предлагается лишь в части платформ.

### **3 Теоретические аспекты разработки**

#### **3.1 Интерфейсная библиотека React JS**

React JS - это библиотека JavaScript с открытым исходным кодом, которая предназначена для разработки пользовательских интерфейсов веб-приложений, особенно для интерактивных веб-страниц. Она позволяет использовать "компоненты которые являются маленькими блоками кода, для создания сложных пользовательских интерфейсов.

React JS прост и удобен в эксплуатации. Понимание основных требований практически не требует времени и может быть выполнено в течение нескольких часов. Разработчики предпочитают использовать React JS по следующим причинам:

1. Динамические приложения требуют более сложного кодирования, и использование React JS упрощает данную задачу;
2. Возможность повторного использования компонентов. В React очень тесно связаны разметка и логика, благодаря чему появляется возможность хранить все данные о внешнем виде компонента и его изменения в процессе работы в одном файле;
3. Использование виртуального дерева вместо физического для повышения производительности будущего приложения. Для JavaScript структура dom-дерева является интерфейсом, через который скрипты приводят в движение html и css-объекты, поэтому производительность в данном случае играет очень значимую роль;
4. Снижается нагрузка на сервер и время разработки. Повышается производительность сайта: быстрее открываются страницы и отзывается на действия пользователей интерфейс;
5. По сравнению с обычными сайтами у сайтов на React более чистая архитектура, в которой проще обнаруживать и исправлять баги и которую проще поддерживать. [21] [22]

Фреймворк React использует JSX, расширение для языка JavaScript, представляя все HTML-элементы интерфейса в роли компонентов, используя подход «разделяй и властвуй», упрощая разработку проекта, особенно удобно для командной разработки. Подобный подход используется во всех современных фреймворках для разработки динамического web-интерфейса [23].

React очень удобно рассматривать для разработки одностраничных при-

ложений. Несмотря на то, что при снятии нагрузки с сервера нагружается его клиентская часть, современные браузеры способны справиться с отрисовкой страниц благодаря достаточному запасу памяти и подхода SPA. Его суть в том, что весь сайт — это одна страница, которую React постоянно перерисовывает — но не целиком. Если в одностраничных страницах есть возможность оптимизации ресурсов, то и в многостраничных решениях оно будет присутствовать вдоволь.

В простом приложении для перехода со страницы на страницу пользователь делает к ней запрос на сервер, и сервер возвращает разметку, стили и файлы скриптов. В случае одностраничных приложений пользователь, переходя между разделами сайта, формально находится на одной и той же странице, но файлы скриптов и стили у него уже есть — остается доделать то, чего не хватает.

По состоянию на конец 2023 года React находится в лидирующих позициях среди JavaScript-технологий для разработки фронтенда [22].

### 3.2 Серверная библиотека NextJS

Переходя к основоположникам применения серверных компонентов можно отнести создателей средства Next.js, развивающих упомянутую концепцию. Next.js — это фреймворк, основанный на React, который позволяет создавать веб-приложения с улучшенной производительностью, используя рендеринг на серверной стороне [24].

Дополнительный функционал NextJS обеспечивает реализацию оптимизированных и полезных возможностей, таких как:

1. Server Side Rendering — позволяет рендерить полноценную страницу на сервере, после чего она полностью отправляется на клиентскую часть и сразу же отображается, что повышает производительность веб-страниц;
2. Search Engine Optimization — помогает веб-приложению занимать более высокие места в поисковых системах. SEO работает в tandemе с SSR и позволяет трекерам сканировать больше контента;
3. Упрощенный процесс разворачивания приложения [25].

Next.js может быть использован для разработки клиент-серверных приложений с улучшенной производительностью и возможностью использования серверного рендеринга. В качестве фреймворка, основанного на React, Next.js предоставляет средства для создания веб-приложений, которые взаимодействуют с сервером. Он обеспечивает серверный рендеринг, что означает, что при-

ложение может получить полноценную HTML-страницу на сервере и отправить ее на клиентскую сторону для отображения. Это позволяет повысить производительность сайта и улучшить SEO-оптимизацию.

Следовательно, основные подходы к использованию Next.js для разработки клиент-серверных приложений включают в себя применение серверного рендеринга для оптимизации производительности, управление маршрутизацией, обработку данных на стороне сервера, а также управление локальным состоянием приложения.

### 3.3 Интерфейсная библиотека TailWind

Tailwind CSS является набирающим популярность CSS-фреймворком, который дает возможность вносить изменения в оформление сайтов и приложений, не покидая HTML-разметку, в том числе и в компонентах React, не используя тег `style`. Эти служебные классы позволяют инкапсулировать общие шаблоны стилей, такие как поля, отступы, цвета и макеты флексбоксов, что позволяет быстро создавать и настраивать пользовательский интерфейс [26].

Одним из ключевых преимуществ является гибкость и расширяемость. Фреймворк предоставляет широкие возможности настройки, позволяющие разработчикам настраивать структуру в соответствии с конкретными требованиями проекта. Эта гибкость делает его подходящим для широкого спектра проектов: от небольших личных веб-сайтов до крупномасштабных корпоративных приложений.

Tailwind CSS является его ориентация на производительность и оптимизацию. Используя служебные классы, разработчики могут избежать раздувания, которое часто возникает при использовании традиционных платформ CSS, что приводит к меньшим размерам файлов CSS и более быстрому времени загрузки. Более того, поощряет модульный подход к стилизации, который может привести к созданию более удобных в обслуживании и масштабируемых баз кода.

Несмотря на свой подход, ориентированный на полезность, Tailwind CSS не жертвует гибкостью дизайна. Платформа предоставляет полный набор служебных классов, которые охватывают широкий спектр вариантов стилей, что позволяет разработчикам с легкостью создавать визуально привлекательные и адаптивные проекты. Кроме того, Tailwind CSS легко интегрируется с популярными интерфейсными платформами, такими как React, Vue.js и Angular, что упрощает его включение в существующие проекты [10].

Основными преимуществами TailWind CSS являются следующие возможности:

1. Наличие базовых классов для оформления страниц. В библиотеке есть базовый конфигурационный файл, и в него по умолчанию включена большая коллекция классов, необходимых для стилизации приложения, таких как flex, grid, block для display, m-10 вместо margin и подобных;
2. Наличие расширенной цветовой палитры. Цвета здесь имеют привычные названия и постфикс с насыщенностью в цифровом обозначении, например, red-50 будет являться бледно-розовым цветом, когда red-900 уже приобретет более бордовый оттенок;
3. Продвинутые CSS-свойства. Различные анимации, радиусы закругления рамок, повороты, отступы;
4. Переменные. Помимо классов, в Tailwind CSS есть и переменные в стиле тех, что используется в CSS-препроцессорах. С помощью них можно уложить несколько классов в один и использовать его как классический семантический селектор. Делается это с помощью директивы @apply;
5. Добавление собственных классов. В этом плане Tailwind можно настроить под индивидуальные требования и предпочтения. Для этого требуется прописать новые свойства в конфигурационный файл фреймворка [26].

Подводя итог, Tailwind CSS предлагает современный и эффективный подход к стилизации веб-приложений. Благодаря своей философии приоритета полезности, широким возможностям настройки и ориентации на производительность дает разработчикам возможность создавать красивые и отзывчивые пользовательские интерфейсы с меньшими усилиями и большей гибкостью [27].

### 3.4 Архитектурный стиль Rest API

Rest API - это интерфейс программирования приложений, основанный на принципах REST. Web API или Web Service API – это интерфейс обработки приложений между веб-сервером и веб-браузером. Все веб-сервисы являются api, но не все api являются веб-сервисами. Rest API – это особый тип Web API, в котором используется стандартный архитектурный стиль, описанный выше [28].

Различные термины, которые относятся к API, например, Java API или сервисные API, существуют в связи с исторически ранним происхождением, тк API-архитектуры начали появляться до запуска Всемирной паутины. Современные web API – это REST API, и эти термины могут использоваться взаимозаменяющими образом.

няемо.

Данный архитектурный стиль стал популярным из-за своей простоты, гибкости и масштабируемости. Он использует стандартные HTTP-методы, такие как GET, POST, PUT и DELETE, для работы с ресурсами на сервере. Каждый ресурс имеет уникальный идентификатор, с которым клиенты могут взаимодействовать.

В качестве основных преимуществ архитектурного стиля Rest API можно выделить следующие ключевые моменты:

1. Гибкость – позволяет разработчикам использовать различные форматы данных, такие как json или html. Это позволяет клиентам и серверам обмениваться данными в удобном для них формате;
2. Простота – использует стандартные HTTP-методы, что делает его простым в использовании и понимании. Разработчики могут легко создавать, изменять и расширять api без необходимости в специальных инструментах или библиотеках;
3. Масштабируемость – позволяет разрабатывать распределенные системы, где клиенты и серверы могут находиться на разных компьютерах или даже в разных частях мира. Это делает его идеальным для создания клиент-серверных приложений;
4. Кэширование – поддерживает кэширование данных на клиентской стороне, что позволяет улучшить производительность и снизить нагрузку на сервер;
5. Независимость от платформы – не привязан к определенной платформе или языку программирования. Это означает, что клиенты и серверы могут быть реализованы на различных технологиях, а это уже обеспечивает большую гибкость и возможность использования уже существующих разработанных систем. [29]

В настоящее время Rest API является широко используемым и популярным подходом для разработки клиент-серверных приложений, благодаря своей простоте, гибкости, масштабируемости и прочим незаменимым, полезным возможностям.

### **3.5 Сервис тестирования API Postman**

Postman — это инструмент, используемый разработчиками для создания, тестирования и документирования API. Его основная особенность заключается

в том, что он упрощает каждый этап жизненного цикла API и оптимизирует совместную работу для более качественной API-разработки [30]. Postman позволяет выполнять различные типы HTTP-запросов (GET, POST, PUT, PATCH), сохранять среды для последующего использования, преобразовывать API в код для использования в различных языках (например, JavaScript и Python).

Postman обладает широким спектром инструментов, предназначенных для проведения качественного тестирования различных программных приложений и веб-сервисов. Включая коллекции запросов и удобный механизм ввода параметров, этот функционал существенно облегчает работу тестировщика, позволяя эффективно автоматизировать рутинные задачи и выявлять потенциальные ошибки при взаимодействии с API [31].

Postman предлагает ряд функций, которые упрощают процесс разработки:

1. Универсальные методы запросов: Postman поддерживает множество методов HTTP-запросов, включая GET, POST, PUT, DELETE и PATCH. Эта универсальность позволяет разработчикам всесторонне взаимодействовать с API;
2. Управление окружениями: Postman позволяет создавать разные "окружения" для тестирования API. Так, можно иметь отдельное окружение для разработки, тестирования и продакшна, каждое из которых может использовать разные базы данных или внешние сервисы;
3. Упрощенная аутентификация: Postman упрощает тонкости аутентификации, обеспечивая поддержку различных методов, таких как ключи API, OAuth и базовая аутентификация. Это упрощает процесс обеспечения взаимодействия API, обеспечивая надежную и безопасную среду разработки;
4. Автоматизация тестов: Postman предоставляет возможности для написания тестов, которые могут автоматически проверять ответы API на соответствие ожидаемым результатам. Это включает в себя проверку статус-кодов ответов, структуры данных и других ключевых аспектов ответа;
5. Эффективная документация: Postman хорошо генерирует документацию по API непосредственно из запросов. Эта функция обеспечивает оптимизированный и централизованный подход к документированию API, полезный как для внутренних команд разработчиков, так и для внешних заинтересованных сторон. Процесс документирования является эффектив-

тивным, обеспечивая ясность и доступность [32].

В качестве основных задач и этапов разработки Postman находит свое применение в веб-приложении, которое используются для проверки и тестировании API. Это веб-приложение ускоряет процесс разработки, так как разработчики могут сразу проверить корректность реализации API. Если при разработке возникают ошибки, Postman позволяет быстро тестировать изменения в API и анализировать ответы, что помогает находить и устранять проблемы. Postman также поддерживает совместные проекты, позволяя командам разработчиков делиться коллекциями запросов и окружениями, что упрощает совместную работу над API. Для новых сотрудников и сторонних разработчиков наличие хорошо документированных и легко доступных API через Postman упрощает их понимание и взаимодействие с веб-приложением. Будучи комплексным и удобным для пользователя инструментом, Postman дает разработчикам возможность точно ориентироваться в тонкостях взаимодействия API, способствуя оптимизации рабочего процесса [32].

### **3.6 Система управления базами данных MongoDB**

MongoDB представляет из себя документо-ориентированную СУБД, которая долгое время играет значительную роль в разработке веб-приложений. Свою популярность MongoDB получила за счёт высокой производительности, гибкости структур данных, масштабируемости и легкости интеграции с различными платформами и языками программирования.

Модель документа MongoDB проста для изучения и использования разработчиками, но при этом предоставляет все возможности, необходимые для удовлетворения самых сложных требований в любом масштабе:

1. Документо-ориентированность: В MongoDB информация хранится в виде документов JSON (BSON - Binary JSON), что делает СУБД гибкой в отношении схем данных. Это позволяет разработчикам легко и быстро модифицировать схему базы данных без необходимости остановки базы данных или затраты больших усилий на миграции данных;
2. Масштабируемость: MongoDB поддерживает горизонтальное масштабирование через механизм шардирования, позволяющий распределить данные по нескольким серверам. Это может существенно повысить производительность и объём хранилища;
3. Высокая доступность: СУБД использует модель репликации, обеспечивая

автоматическое создание копий данных на различных серверах. Это не только повышает устойчивость и доступность приложения в случае сбоев, но и улучшает время ответа на запросы за счет распределенной обработки [33] [34].

Что касается разработки веб-приложений, MongoDB активно используется в данном направлении благодаря своей способности быстро обрабатывать большие объемы структурированных и неструктурированных данных. Её документо-ориентированный подход своевременно удовлетворяет требования современных веб-приложений, где может потребоваться изменение структуры данных в силу изменения бизнес-требований или пользовательских предпочтений. Например, это может быть разработка социальных сетей, где требуется хранение большого количества медиафайлов, интернет-магазинов, с применением гибкости схемы СУБД для каталогов с различными атрибутами товаров, и сфера аналитики и Big Data, в которой необходимо поддерживать хранение и быструю обработку информации, в частности для систем с аналитическими запросами в реальном времени. Также MongoDB легко интегрируется с большинством современных технологических стеков и языков программирования, таких как JavaScript (Node.js), Python, Java, PHP, Ruby и других [33] [35].

Применение MongoDB в разработке веб-приложений демонстрирует значительные преимущества в управлении гибкостью, масштабируемостью и производительностью. Она подходит для различных типов приложений, от простых веб-сайтов до сложных корпоративных решений. Успешно зарекомендовав себя в мире высоконагруженных интернет-приложений и сервисов, MongoDB остается одним из лидеров среди NoSQL систем управления базами данных.

### 3.7 Платформа сборки и управления Docker

Docker представляет собой инновационную платформу, целью которой является обеспечение эффективной разработки, развертывания и запуска приложений в изолированных контейнерах. Термин "Docker" стал ассоциироваться с концепцией контейнеризации в современной технологической среде. Применение Docker и контейнеризация в целом приобрели ключевое значение в сфере веб-разработки, обеспечивая упаковку и изоляцию приложений с целью оптимизации процессов их развертывания и управления [36].

Как и стандартный многоразовый пластиковый контейнер, Docker обладает рядом ключевых атрибутов. Внутри контейнера можно хранить различные

артефакты, которые могут находиться как в пределах контейнера, так и за его границами. Контейнер Docker мобилен и может быть использован на локальном компьютере или на сервере облачного провайдера, такого как AWS.

Особенности контейнеров Docker включают возможность удобного добавления и извлечения содержимого, а также взаимодействия с внешними средствами. Например, контейнеры обладают портами, которые могут быть открыты для доступа к приложениям через браузер. Работа с контейнерами также возможна с помощью командной строки.

Образы Docker-контейнера сохраняются в специальных репозиториях и могут быть загружены и использованы для создания контейнера по запросу. Используя Docker, можно одновременно запускать несколько контейнеров на одном компьютере, управлять их жизненным циклом, исследовать содержимое и создавать новые контейнеры.

Dockerfile представляет собой файл, содержащий инструкции для сборки образа контейнера. В нем описывается базовый образ, который представляет начальный слой контейнера. Популярные базовые образы включают в себя python, ubuntu, alpine. Помимо базового образа, в образ контейнера можно добавлять дополнительные слои с помощью инструкций из Dockerfile. Например, в случае описания образа для машинного обучения, можно указать инструкции для добавления библиотек NumPy, Pandas и Scikit-learn.

Для запуска контейнера необходим образ контейнера и наличие среды, поддерживающей выполнение команд, создающих контейнер с указанным образом и запускающих его в среде, поддерживающей Docker [37] [38].

### 3.8 Облачная платформа Vercel

Облачная платформа Vercel является инновационным инструментом для разработки и развертывания веб-приложений, обладающим рядом преимуществ и возможностей. Она предоставляет разработчикам удобное и эффективное окружение для создания современных веб-приложений, применяя современные технологии и подходы. Главными его преимуществами являются высокая скорость развертывания, автоматическое масштабирование, простота использования и интеграции с различными фреймворками, такими как Next.js и другими. Vercel также обладает встроенной поддержкой CI/CD (непрерывной интеграции и непрерывной доставки), что способствует автоматизации процессов разработки и деплоя приложений [39].

Использование платформы Vercel позволяет разработчикам сосредоточиться на создании функциональности приложения, минуя сложности настройки инфраструктуры и управления серверами. Vercel поддерживает различные языки программирования, фреймворки и инструменты, что делает ее универсальным и гибким решением для разработчиков. Платформа обеспечивает автоматическое масштабирование приложения в зависимости от нагрузки, что позволяет эффективно управлять ресурсами и обеспечивать отзывчивость сервиса для пользователей [39] [40].

Разработка веб-приложения на React с использованием Vercel начинается с создания репозитория на GitHub, GitLab или Bitbucket. После подключения репозитория, каждый push кода активирует процесс автоматического развертывания. Если проект использует Next.js, Vercel автоматически определит этот фреймворк и применит оптимизации, специфичные для серверного рендеринга (SSR) или статической генерации (SSG).

Рассматривая практическое применение Vercel в веб-разработке, чаще всего данный инструментарий используется для реализации следующих категорий проектов:

1. Развитие стартапа: Компания-стартап, желающая быстро развернуть и оптимально масштабировать свой веб-проект, может воспользоваться Vercel для размещения MVP (минимально жизнеспособного продукта) и последующего продолжения разработки, периодически внося изменения и обновления через CI/CD. Такие успешные примеры разработки можно увидеть непосредственно на сайте разработчиков Vercel [41];
2. Разработка личного портфолио: Веб-разработчик, стремящийся продемонстрировать свои работы и навыки, может использовать Vercel для хостинга своего личного портфолио. Таким образом, развертывание и поддержка сайта будут минимальными, а работа будет доступна широкой аудитории;
3. Онлайн-магазин: Для запуска онлайн-магазина, где требуется быстрая загрузка страниц и высокая доступность, команда разработчиков может воспользоваться Vercel, чтобы обеспечить стабильную работу приложения и удобное управление контентом.

Благодаря своим преимуществам, Vercel остается незаменимым инструментом для веб-разработки за последнее время.

## **4 Основные аспекты разработки платформы единого резюме**

С учётом собранных данных при анализе научной литературы и конкурентных сервисов для разработки собственной платформы единого резюме необходимо придерживаться следующим пунктам:

1. Реализовать сбор информации о текущих резюме с уже существующих сервисов;
2. Предоставить возможность объединения, замены, дополнения или удаления смежных или отсутствующих пунктов данных;
3. Создание единого бланка с отредактированными или созданными полями, с возможностью хранения их на личном пространстве;
4. Обновление резюме на сторонних площадках новыми пунктами информации.

Для составления необходимого функционала платформы единого резюме, следует реализовать следующее:

1. Удобный пользовательский интерфейс, соответствующий современным web-стандартам и позволяющий пользоваться функционалом сервиса на любом устройстве, поддерживающим работу с браузерами;
2. Авторизацию пользователей и агрегацию данных на сервере проекта;
3. Возможность создания пользователем персональной страницы на основе шаблонов, для возможности доступа к актуальному резюме при переходе по ссылке сервиса;
4. Возможность переноса резюме из стороннего ресурса в шаблон платформы;
5. Публичное пространство, которое может быть использовано как место просмотра резюме работодателями для предложений о сотрудничестве;
6. Публикацию серверного проекта на хостинг, поддерживающий динамическое создание и обновление страниц согласно подготовленным данным.

### **4.1 Создание прототипа платформы резюме**

Прототипирование – разработка интерактивной модели приложения, симулирующее коммуникацию пользователя с интерфейсом и созданное для тестирования базового функционала. Данный этап необходим чтобы удобно и корректно выстроить логику взаимодействия с продуктом и достижения поставленной задачи.

Для создания прототипа использовано приложение Figma, так как оно является бесплатным (платная версия отличается от бесплатной количеством единовременных редакторов проекта и невозможностью организовать команду) и полностью подходит для реализации данного этапа разработки. Были изображены экраны бланка, страницы ввода и свободногоредактора контента, показанные на рисунке 4.1.

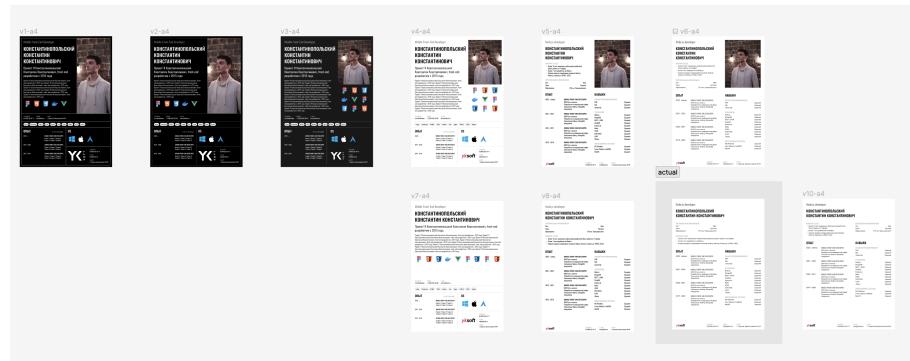


Рисунок 4.1 – Внешний вид прототипов платформы

Далее идёт проверка прототипа на реальных людях, проведённая коридорным тестированием – перед случайными людьми ставится задача взаимодействовать с интерфейсом приложения, тогда как разработчик убеждается в корректности задуманного поведения и находит ошибки. Такие тесты помогают максимально быстро получить обратную связь, потратив минимальное количество времени на поиск респондентов. Результатом тестирования стало подтверждение правильности разработанного прототипа, поэтому можно перейти к увеличению детализации экранов и написанию кода. Важным аспектом при разработке прототипа было сохранение адаптивности страницы, чтобы контент вписывался в различные разрешения экрана и кроме того мог быть адаптирован под печатную версию в формате А4.

## 4.2 Настройка рабочей среды

Создание приложения происходило на компьютере под управлением Mac OS, что позволяет использовать Terminal – средство взаимодействия с системой посредством командной строки при помощи bash-команд. В качестве среды разработки был выбран Visual Studio Code – легковесный текстовый редактор для кроссплатформенной разработки.

С помощью терминала установим пакетный менеджер Homebrew, который необходим для загрузки и запуска компонентов системы. Сделать это можно с

помощью команды:

```
1 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
→ HEAD/install.sh)"
```

Для работы с библиотеками разрабатываемой платформы необходимо выполнить установку Node.js – среды запуска Java Script приложений и Watchman – утилиты для контроля изменений файлов:

```
1 brew install node
2 brew install watchman
```

Инициализируем проект, создав стартовое рабочее приложение:

```
1 mkdir cv-editor
2 cd cv-editor
3 npm init
```

После перехода в папку проекта и инициации проекта необходимо добавить требуемые зависимости, такие как parcel, необходимый для итоговой сборки и работы в режиме сервера и editor.js, необходимый для использования возможностей динамического ввода и сохранения информации пользователя:

```
1 yarn add --dev parcel
2 yarn add @editorjs/editorjs
```

### 4.3 Общие настройки динамического приложения

Для компенсации недостатков статических сайтов и динамической генерации страниц из набора входящих параметров и данных инициализируем проект с NextJS и добавим интерфейсную библиотеку NextUI для удобства разработки следующими командами:

```
1 npx create-next-app@latest
2 npm i @nextui-org/react framer-motion
```

Для безопасной работы с проектом необходимо создать файл process.env, в котором будут храниться переменные окружения, например токен доступа к системе.

Дополнительно для каждой страницы необходимо задать общий шаблон в файле layout.tsx, находящимся в корневой папке приложения. Данная структура представляет из себя обёртку над стандартным представлением HTML

файла, внутрь которого добавлено реактивное представление. Дополнительно внутри атрибутов указываются стили для блоков, такие как минимальная высота контейнера, цвет фона, размеры контейнеров, отступов и их поведение, выбор шрифта и текущая цветовая схема. В дальнейшем содержимое атрибутов className будет игнорироваться.

```
1 export default function RootLayout({ children, }: {  
2     children: React.ReactNode;  
3 }) { return (  
4     <html lang="en" suppressHydrationWarning>  
5         <head></head>  
6         <body className={clsx( "min-h-screen bg-background font-sans antialiased",  
7             → fontSans.variable )}>  
8             <Providers themeProps={{ attribute: "class", defaultTheme: "dark" }}>  
9                 <div className="relative flex flex-col h-screen">  
10                     <Navbar/>  
11                     <main className="container mx-auto max-w-7xl pt-16 px-6 flex-grow">  
12                         {children}  
13                     </main></div></Providers></body>  
14     </html>  
15 ); }
```

#### 4.4 Реализация клиентской части платформы

Входным файлом проекта назначим index.html. Опишем его стандартную структуру и внутрь тега head добавим ссылки на файл стилей main.css и файл шрифта Mona-Sans.woff2 и стили библиотеки Bootstrap:

```
1 <title>CV Editor</title>  
2 <link rel="stylesheet" href="css/main.css">  
3 <link rel="preload" href=".src/font/Mona-Sans.woff2" as="font"  
4     → type="font/woff2">  
5 <link href="bootstrap.min.css" rel="stylesheet">
```

В том же файле добавим блок вызова редактора, кнопку сохранения введённых данных и после в конец файла перед закрытием тега body подключим файл скриптов проекта index.js и ссылку на файл скриптов библиотеки стилей:

```
1 <div id="editorjs"></div>  
2 <button type="button" id="saveButton">Save</button>  
3 <script src=".index.js" type="module"></script>  
4 <script src="bootstrap.bundle.min.js"></script>
```

После краткого описания языка разметки нам необходимо реализовать функционал редактора. Для этого в файле index.js импортируем саму библиотеку и вспомогательные модули для отображения и работы с заголовками и стилями. Для описания самого редактора создадим новый объект редактора, включающий в себя вышеописанные библиотеки:

```
1 import EditorJS from '@editorjs/editorjs';
2 import Header from '@editorjs/header';
3 import List from '@editorjs/list';
4
5 const editor = new EditorJS({
6   holder: 'editorjs',
7   tools: {
8     header: {class: Header, inlineToolbar: ['link']},
9     list: {class: List, inlineToolbar: true},
10   },
11 });


```

Для использования введённой пользователем информации из блока редактора реализуем функцию получения данных по нажатию кнопки сохранения. Для этого создадим обработчик события, находящегося на странице по идентификационному значению и через него вызовем метод сохранения. Чтобы избежать падения сервера опишем обработчик исключений, возвращающий данные в случае успешного выполнения и выводящий ошибку в случае возможного возникновения проблем:

```
1 document.getElementById("saveButton").onclick = function() {
2   editor.save().then((outputData) => {
3     console.log('Article data: ', outputData)
4   }).catch((error) => {
5     console.log('Saving failed: ', error)
6   });
7 };


```

Создадим заголовок страницы, добавим классы для корректного позиционирования кнопок на странице с помощью библиотеки Bootstrap. Для этого создадим тег с классом контейнера и внутренними отступами с помощью классов container и p-4. Добавим краткую подсказку для запуска проекта на время разработки и стилизуем кнопку: btn btn-outline-primary. Итоговый вид и функционал редактора показан на рисунке 4.2.

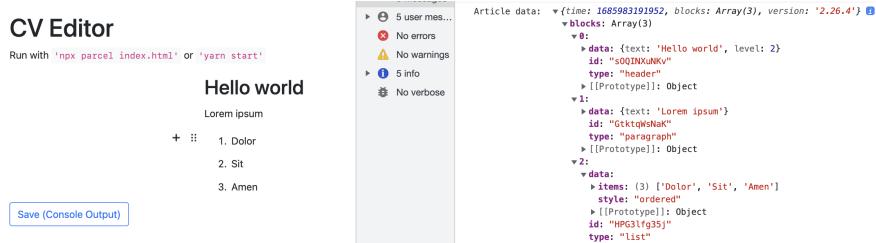


Рисунок 4.2 – Внешний вид свободного редактора данных

Реализуем вкладки для шаблона с двумя экранами: заполненным резюме и полями ввода. Для этого внутрь контейнера добавим список из двух значений, добавим им атрибуты вкладок и обозначим первый таб за активный, чтобы при заходе на страницу платформы бланк загружался сразу.

```

1 <div class="container p-4">
2   <ul class="nav nav-tabs" id="myTab" role="tablist">
3     <li class="nav-item" role="presentation">
4       <button class="nav-link active" id="home-tab" data-bs-toggle="tab"
5         &gt; data-bs-target="#home" type="button" role="tab"
6         &gt; aria-controls="home" aria-selected="true">CV Template</button>

```

Согласно дизайну сверстаем шаблон и предусмотрим адаптивное отображение страницы, чтобы сайт корректно показывался как на смартфонах, так и на планшетах и компьютерах. Для этого необходимо правильно описать стили в классах тегов – от наименьшего или стандартного разрешения к самому большому.

```

1 <h5 class="h5 text-uppercase text-secondary mt-5">
2   <strong>Персональная информация</strong></h5>
3   <div class="row">
4     <div class="col-6 col-lg-3">Пол</div>
5     <div class="col-6 col-lg-3 text-end">Муж</div>
6   </div>
7   <div class="row">
8     <div class="col-6 col-lg-3">Язык</div>
9     <div class="col-6 col-lg-3 text-end">Русский</div>
10  </div>

```

Дополнительно для удобства вёрстки воспользуемся сеткой, представляющей из себя вертикальные направляющие, делящие html-страницу на равные части относительно центра страницы с равными отступами от краёв экрана и равным отступом между колонками. Так на компьютерах с разрешением 1920

пикселей боковой отступ будет составлять 365 пикселей, размер рабочего контейнера – 1190 пикселей, всего будет 12 колонок с расстоянием между ними в 30 пикселей. Помимо этого внутренние контейнеры также адаптивно делятся в своих пропорциях, образуя сетку в сетке. Итоговый вид шаблона и первой вкладки показан на рисунке 4.3.



Рисунок 4.3 – Отображение вкладки переключения и шаблона

После вёрстки бланка резюме необходимо реализовать ручные поля ввода, из которых будет приходить информация для страницы. Разобьём их по категориям и представим пользователю в доступном виде, необходимом для последовательного заполнения. Для этого воспользуемся оформлением стилей из библиотеки Bootstrap и реализуем однострочные, многострочные поля ввода и селекторы, в которых изначально предложены варианты выбора данных и стандартное значение.

```
1 <h3 class="h3 pb-2">Personal information</h3>
2   <div class="row g-4 pb-4">
3     <div class="col-md">
4       <div class="form-floating">
5         <select class="form-select" id="floatingSelectGrid">
6           <option selected>Select</option>
7           <option value="1">Male</option>
8           <option value="2">Female</option>
9         </select>
10        <label for="floatingSelectGrid">Gender</label>
11      </div>
12    </div>
```

Результатом написания данного блока является вкладка с множеством полей ввода, показанных на рисунке 4.4.

The screenshot shows a user interface for a CV template. At the top, there are two tabs: 'CV Template' (which is selected) and 'CV Fields'. Below the tabs, there are two main sections: 'Main information' and 'Personal information'. The 'Main information' section contains fields for 'Full name' and 'Position'. The 'Personal information' section contains dropdowns for 'Gender' (set to 'Male'), 'Language', and 'Education', along with a large text area for 'About yourself'.

Рисунок 4.4 – Отображение вкладки ввода данных

Структура хранения данных о пользователе представляет из себя json-файл, содержащий все описанные выше поля, вроде имени, даты рождения, пола и так далее. Данный выбор формата универсален в рамках нашего приложения и позволяет взаимодействовать с информацией быстрым и удобным способом.

```
1 "last_name": "Константинопольский",
2 "first_name": "Константин",
3 "middle_name": "none",
4 "age": 33,
5 "birth_date": "1991-04-30",
6 "gender": "male",
```

## 4.5 Реализация серверной части платформы

После написания клиентской части приложения необходимо реализовать часть, отвечающую за перенос резюме из сторонних проектов. Для решения поставленной задачи воспользуемся парсером html-страницы на основе библиотеки Puppeteer. Инициируем новый проект аналогично тому, что было в реализации клиентской части и добавим библиотеку для скраппинга:

```
1 mkdir parser-cv-editor
2 cd parser-cv-editor
3 npm init
4 yarn add puppeteer
```

Работа парсера заключается в создании экземпляра браузера и выполнении заранее прописанных действий. Для этого создадим файл browser.js и

опишем логику работы компонента. Так как браузер будет открываться в терминальном состоянии – его визуальное представление необходимо только для отладки работы и во время публикации параметр можно выключить:

```
1 const puppeteer = require('puppeteer');
2 async function startBrowser(){
3     let browser;
4     try {
5         console.log("Opening the browser");
6         browser = await puppeteer.launch({
7             headless: true,
8             args: ["--disable-setuid-sandbox"],
9             'ignoreHTTPSErrors': true
10        });
11    } catch (err) {
12        console.log("Could not create a browser instance => : ", err);
13    }
14    return browser;
15 }
16 module.exports = {
17     startBrowser
18 };
```

За управлением экземпляра браузера отвечает pageController.js. В его задачи входит проход по указанной странице и сохранение полученных данных в хранилище в формате json. Для этого подключается с помощью require в Node JS подключается работа с файловой системой, после чего инициализируется объект браузера и через асинхронную функцию происходит считывание страницы с её последующей записью. Дополнительно реализуем обработчик исключений чтобы работа серверной части не нарушалась в случае возникновения ошибок как в момент перехода на страницу, так и в момент записи информации:

```
1 const fs = require('fs');
2 async function scrapeAll(browserInstance){
3     let browser;
4     try{
5         browser = await browserInstance;
6         let scrapedData = {};
7         scrapedData = await pageScraper.scrapedData(browser);
8         fs.writeFile("data.json", JSON.stringify(scrapedData), 'utf8',
9             → function(err) {
```

```

9             if(err) {return console.log(err);}
10            console.log("Done");
11        });
12    } catch(err){console.log("Error in ", err);}
13 }
14 module.exports = (browserInstance) => scrapeAll(browserInstance)

```

После описания вспомогательных файлов серверной части платформы необходимо реализовать точку входа в приложение. Для этого в index.js подключим вышеописанные компоненты, создадим экземпляр браузера и запустим работу контроллера с данным объектом:

```

1 const browserObject = require('./browser');
2 const scraperController = require('./pageController');
3 let browserInstance = browserObject.startBrowser();
4 scraperController(browserInstance)

```

## 4.6 Написание механизма авторизации

Для реализации авторизации создадим файл auth.ts, в котором укажем конфигурацию с сервисом Github посредством открытого протокола безопасности OAuth и с классической проверкой логина и пароля. Для этого создадим два соответствующих свойства и в первое передадим переменные приложения для авторизации через сторонний сервис, а в втором реализуем проверку введённых в поля данных и в случае соответствия вернём самого пользователя с его именем, фотографией профиля и ролью [10].

```

1 export const authConfig: AuthOptions = { providers: [
2     GithubProvider({
3         clientId: process.env.GITHUB_ID!,
4         clientSecret: process.env.GITHUB_SECRET! },
5     Credentials({
6         credentials: {
7             email: { label: 'email', type: 'email', required: true },
8             password: { label: 'password', type: 'password', required: true }, },
9             async authorize(credentials) {
10                 if (!credentials?.email || !credentials.password) return null;
11                 const currentUser = users.find(user => user.email ===
12                     credentials.email)
13                 if (currentUser && currentUser.password === credentials.password) {
14                     const { password, ...userWithoutPass } = currentUser;
15
16
17
18
19
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
305
306
307
308
309
309
310
311
312
313
314
315
315
316
317
318
319
319
320
321
322
323
324
325
325
326
327
328
329
329
330
331
332
333
334
335
335
336
337
338
339
339
340
341
342
343
344
344
345
346
347
348
348
349
349
350
351
352
353
353
354
355
356
356
357
358
359
359
360
361
362
362
363
364
364
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
13
```

```

14         return userWithoutPass as User;}
15     return null })
16 ], pages: { signIn: '/signin' } }

```

Ограничение доступа неавторизованного пользователя к внутренним и закрытым страницам достигается созданием файла middleware.ts, в котором через параметры указываются пути навигации и, в случае попадания в заданный путь, перевод клиента на страницу signIn.

```

1 export const config = { matcher: ['/profile/:path*', '/hh',
2 '/resume','/protected/:path*'] }

```

Дополнительно к предыдущему пункту контроля доступа опишем корневой файл providers.tsx, созданный для окружения корневого элемента страницы в её компоненты, свойства стилей и сессию пользователя.

```

1 export function Providers({ children, themeProps }: ProvidersProps) {
2   return (
3     <SessionProvider><NextUIProvider>
4       <NextThemesProvider {...themeProps}>
5         {children}
6       <NextThemesProvider>
7     <NextUIProvider><SessionProvider>);

```

Процесс авторизации реализуется на клиентской и серверной части приложения, поэтому для корректного взаимодействия необходимо создать обработчик событий по API в динамическом маршруте [..auth]/route.tsx с следующими параметрами:

```

1 const handler = NextAuth(authConfig);
2 export { handler as GET, handler as POST }

```

Реализуем форму авторизации через компонент SignInForm с двумя текстовыми формами, для этого инициализируем роутер переходов, обратные вызовы для динамического контента и обработчик нажатия кнопки отправки формы, получающий данные полей и выполняющий вход в систему.

```

1 const SignInForm = () => {
2   const router = useRouter();
3   const [isVisible, setIsVisible] = React.useState(false);
4   const toggleVisibility = () => setIsVisible(!isVisible);

```

```

5  const handleSubmit: FormEventHandler<HTMLFormElement> = async (event) => {
6      event.preventDefault();
7      const formData = new FormData(event.currentTarget);
8      const res = await signIn("credentials", {
9          email: formData.get("email"),
10         password: formData.get("password"),
11         redirect: false
12     }})

```

Кнопка авторизации через Github работает через параметры поисковой строки, получаемые в процессе возврата на страницу платформы после успешного входа на стороне внешнего сервиса.

```

1 export const GithubButton = () => {
2     const searchParams = useSearchParams();
3     const callbackUrl = searchParams.get("callbackUrl") || "/profile";
4     return (
5         <Button onClick={() => signIn("github", { callbackUrl })}>
6             Log In with GitHub
7         </Button>
8     );
}

```

После написания компонентов необходимо создать саму страницу авторизации, для этого внутрь всей иерархической оболочки страницы передадим заголовок вместе с компонентами SignInForm и GithubButton, после которых страница будет располагаться по маршруту app/signIn. Итоговый вид страницы авторизации отображён на рисунке 4.5.

Для запуска базы данных на локальном устройстве необходимо воспользоваться средством виртуализации Docker. Скопируем образ из официального репозитория сервиса, запустим из этого образа полноценный контейнер с названием проекта с переадресацией порта на системный 27017 и подключимся к нему из командной строки для проверки работоспособности. Данный порт и адрес потребуются для подключения к базе изнутри сервиса.

```

1 docker pull mongodb/mongodb-community-server:latest
2 docker run --env editor mongo -p 27017:27017 -d
   → mongodb/mongodb-community-server:latest
3 mongosh --port 27017

```

Интегрируем базу данных в механизм авторизации. Для этого добавим адаптер MongoDB с помощью следующей команды:

# Sign In

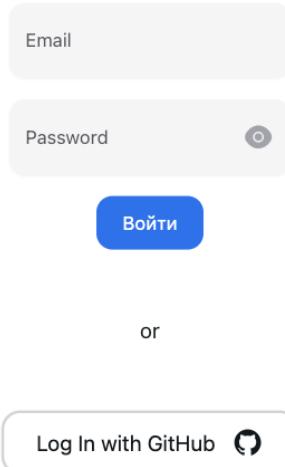


Рисунок 4.5 – Страница авторизации

```
1 npm install @auth/mongodb-adapter mongodb
```

После подключения библиотеки в проект необходимо создать переменную Mongodb-URI в файле process.env с ссылкой на подключение к проекту и в файле auth.ts импортировать необходимые зависимости и инициализировать адаптер в файле авторизации.

```
1 import NextAuth from "next-auth";
2 import { MongoDBAdapter } from "@auth/mongodb-adapter";
3 import clientPromise from "./lib/db";
4
5 export const { handlers, auth, signIn, signOut } = NextAuth({
6   adapter: MongoDBAdapter(clientPromise),
7 });

```

После настройки и подключения зависимостей необходимо создать файл взаимодействия с базой данных db.ts и описать соединение. Импортируем клиент и сервер взаимодействия, последовательно получая и проверяя ссылку на подключение с помощью обработчика ошибок. Опишем переменные с включёнными свойствами версии API, его режимом и проверкой целостности.

```
1 import { MongoClient, ServerApiVersion } from "mongodb";
2
3 if (!process.env.MONGODB_URI) {
4   throw new Error('Invalid/Missing environment variable: "MONGODB_URI"');
```

```

5  }
6  const uri = process.env.MONGODB_URI;
7  const options = {
8      serverApi: {
9          version: ServerApiVersion.v1,
10         strict: true,
11         deprecationErrors: true,
12     },};

```

Создадим синхронного и асинхронного клиента подключения к базе данных и добавим проверку типа запущенного приложения, чтобы в режиме разработки использовать глобальную переменную окружения для сохранения значения при перезагрузке модуля соединения, вызванных особенностью библиотеки HMR (горячая замена модуля). В зависимости от того, запущен проект на внешнем сервере или на локальном устройстве клиенту присваивается описанная ранее ссылка с переданными свойствами и производится подключение к базе данных. Полный код реализации компонента подключения к базе данных можно найти в приложении А.

```

1 let client; let clientPromise: Promise<MongoClient>;
2
3 if (process.env.NODE_ENV === "development") {
4     let globalWithMongo = global as typeof globalThis & {
5         _mongoClientPromise?: Promise<MongoClient>
6     };
7     if (!globalWithMongo._mongoClientPromise) {
8         client = new MongoClient(uri, options);
9         globalWithMongo._mongoClientPromise = client.connect();
10    }
11    clientPromise = globalWithMongo._mongoClientPromise;
12 } else {
13     client = new MongoClient(uri, options);
14     clientPromise = client.connect();
15 };
16 export default clientPromise;

```

## 4.7 Реализация страницы резюме

Для написания модуля необходимо создать структуру данных резюме. Информация о профиле хранится в текстовом формате обмена данными JSON и содержит в себе следующие поля:

1. Id – генерируемый уникальный идентификатор резюме;
2. Name – фамилия и имя человека;
3. Description – краткое описание профиля;
4. Social links – ссылки на социальные сети;
5. Profile image – изображение работника для резюме;
6. About – текстовое поле с рассказом о себе, являющееся аналогом сопроводительного письма;
7. Education, experience – группы образования и опыта работы, включающие в себя следующие пункты:
  - a) Name – наименование места работы или учебного учреждения;
  - б) Description – описание чем занимался соискатель в данное время;
  - в) Start date, end date – дата начала и окончания;
  - г) Current date – поле, показывающее что дата окончания отсутствует и при подсчёте её следует считать текущим днём;
8. Projects – поле проектов, следующее из опыта работы. Включает в себя:
  - a) Name – наименование проекта;
  - б) Description – описание проекта и над чем происходила работа;
  - в) Image – изображение проекта, его логотип, интерфейс.

Последовательно реализуем компоненты, начиная с блока контактов и ссылок на социальные сети, получающий на вход список адресов и возвращающий горизонтальный список кнопок с переходом и миниатюрами соответствующих сервисов. Изображения предварительно описываются в формате svg в компоненте icons.tsx, после чего из строки выделяется имя хостинга с последующим добавлением в массив элементов для отображения на странице.

```

1 function IconVariant(service: string, icon_color:string, icon_size:number) {
2   switch(service) {
3     case "github.com":
4       { return <GithubIcon className={icon_color} size={icon_size}/>; }
5     default:
6       { return <InternetIcon className={icon_color} size={icon_size}/>; }
7   };
8   export const SocialLink = (props : SocialProps) => {
9     const icon_size, icon_color = 36, "text-default-500";
10    const social_icons = props.links.map((element, index) => {
11      var url = new URL(element.toString());
12      return

```

```

13     <Link isExternal href={element.toString()} key={index}>
14         {IconVariant(url.host.toString(), icon_color, icon_size)}
15     </Link>;
16 } );};

```

Итоговый вид компонента показан на рисунке 4.6.

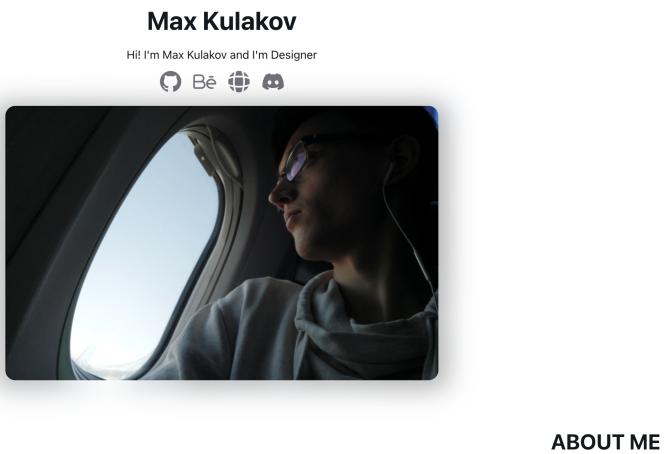


Рисунок 4.6 – Компонент отображения контактов и ссылок

Опишем компонент вывода образования и опыта работы. Для этого реализуем функционал подсчёта текущего затраченного времени через получение двух строковых и одной логической переменных из файла резюме, преобразования текста в объект даты и нахождения разницы между двумя величинами. После вычисления необходимо округлить полученное значение до целого числа и преобразовать в удобочитаемый внешний вид с помощью проверки количества месяцев и лет. Результатом работы функций станет вывод временного значения в строке компонента. Полный код реализации компонента можно найти в приложении Б.

```

1 function diffDates(day_one:any, day_two:any) {
2     return (day_one - day_two) / (60 * 60 * 24 * 1000);
3 };
4 function getFormatedStringFromDays(numberOfDays:number) {
5     var months = Math.floor(numberOfDays % 365 / 30);
6     var yearsDisplay = years > 0 ?
7         years + (years == 1 ? " year " : " years ") : "";
8     return yearsDisplay + monthsDisplay;
9 };

```

```

10 export const Experience = (props : ExperienceProps) => {
11   const experience_item_list = props.experience_list.map((element, index)=>{
12     return { getFormatedStringFromDays(diffDates(element.start_date,
13       ↵ element.end_date))})
13   return  <>{experience_item_list}</>
14 };

```

Итоговый вид компонента показан на рисунке 4.7.

EDUCATION		
2022-09-01 - 2023-10-30 1 year 4 months	Master's degree, Saratov State University	Computer Science, Mathematical Support and Administration of Information Systems
2017-09-01 - 2022-06-30 4 years 10 months	Bachelor's degree, Saratov State University	Computer Science, Mathematical Support and Administration of Information Systems. Graduate work: «Development of mobile application with implementation of accessibility»
EXPERIENCE		
2021-04-25 - 2023-10-30 2 years 8 months	Senior Designer, YKSoft	Audit, development and implementation of solutions in the products of various companies. Implementation of new projects and bringing them to release. Support for existing sites and the application in order to refine the functionality. Interaction with customers and related teams in the process of working on projects. Managing a mini-team of designers
2019-10-01 - 2021-02-01 1 year 4 months	Interface Designer, Venrok	Launch and development of projects from an idea to a final solution with subsequent support
2019-02-01 - 2019-10-01 8 months	Designer, Youcon Events Group	Participation in the organization of events. Work on the website, printed and digital products

Рисунок 4.7 – Компонент отображения опыта работы и образования

Для реализации портфолио в шахматном представлении воспользуемся классовыми атрибутами сетки из библиотеки Tailwind. Получим порядковый индекс списка проектов, начинающийся с нуля, и дальше в зависимости от чётности раздельно и поочерёдно позиционируем текст с 1 по 6 колонку и изображение с 7 по 13.

```

1 export const Projects = (props : ProjectsProps) => {
2   const projects = props.projects_list.map((element, index) => {
3     var position_image = index % 2 == 0 ?
4       "col-start-7 col-end-13" : "col-start-1 col-span-6 ";
5     var position_text = index % 2 == 0 ?
6       "col-start-1 col-end-6" : "col-start-8 col-end-13 order-1";
7     return (
8       <div className={position_text}>
9         <h2>{element.name}</h2>
10        <p>{element.description}</p>
11        </div>
12        <Image className={position_image}/>
13    )));
14   return <>{projects}</>
15 };

```

Итоговый вид компонента списка проектов показан на рисунке 4.8.

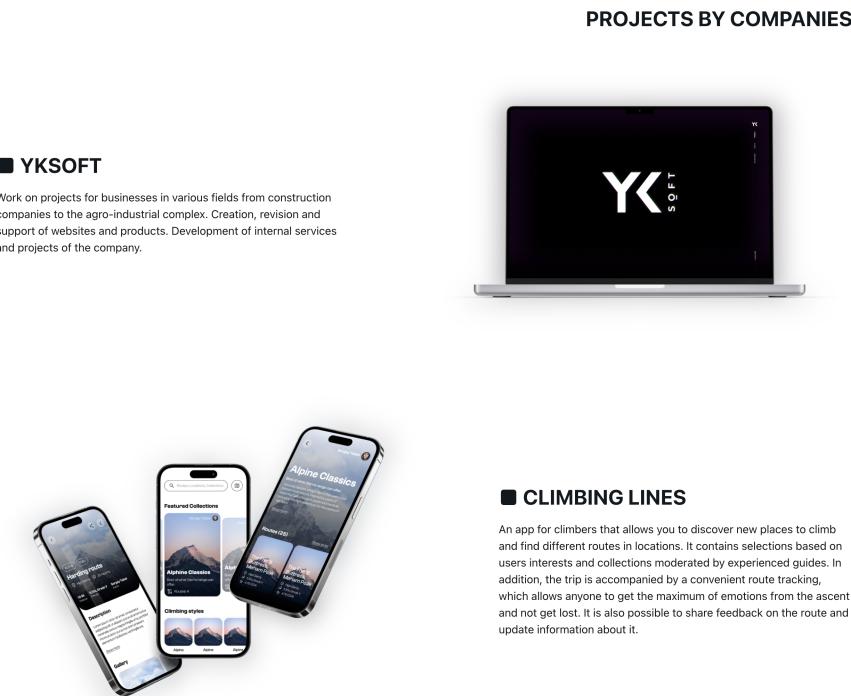


Рисунок 4.8 – Компонент отображения списка проектов

После описания компонентов реализуем страницу резюме, для этого необходимо произвести запрос данных резюме и передать его поля в соответствующие теги и компоненты. Полный код реализации страницы можно найти в приложении В.

```
1 export default function Template1Page() {  
2   const profile_data = require("@/data-template/template-1-data.json");  
3   return (  
4     <SocialLink links={profile_data.social_links}/>  
5     <div className="container pt-5 pb-5">  
6       <h2 id="projects">projects by companies</h2>  
7       <Projects projects_list={...profile_data.projects}/></div>);  
}
```

## 4.8 Взаимодействие с сервисом hh.ru

Реализуем взаимодействие пользователя с сервисом hh.ru через API. Первоначальная конфигурация в виде авторизации состоит из 3 этапов:

1. Перенаправление пользователя по адресу сервиса с переданными параметрами строки;
2. Получение временного кода авторизации;
3. Обмен кода авторизации на долгосрочный токен доступа к профилю и кода обновления токена в случае необходимости. [12]

Первый этап достигается с помощью передачи параметров строки запроса через вопросительный знак, имя параметра и значение. В ответ на полученную от сервиса hh.ru ссылку с переходом необходимо обработать ответный параметр временного кода авторизации и обмена на долгосрочный. Опишем данный функционал в api/hh/route.tsx, для этого необходимо реализовать GET запрос, предварительночитывающий строку запроса и параметр code, в заголовок которого добавляем желаемое действие, открытый и секретный идентификаторы приложения и полученный на предыдущем обработчике код. Так как функция запросов асинхронна – воспользуемся обработчиком событий и через ожидающий асинхронный обратный вызов функции передадим запрос по адресу сервиса, получив в ответ необходимый набор данных: ответ об успешности запроса и токены [4].

```
1 export async function GET(req: Request) {
2   const { searchParams } = new URL(req.url)
3   const code = searchParams.get('code');
4   if (code != null) {
5     myHeaders.append("Content-Type", "application/x-www-form-urlencoded");
6    .urlencoded.append("grant_type", "authorization_code");
7    .urlencoded.append("client_id", process.env.HH_ID!);
8    .urlencoded.append("client_secret", process.env.HH_SECRET!);
9    .urlencoded.append("code", code!);
10    try {
11      const res = await fetch("https://hh.ru/oauth/token", {
12        method: 'POST', headers: myHeaders,
13        body:.urlencoded, redirect: 'follow' });
14      const result = await res.json();
15      return NextResponse.json(result);
16    }
17  }
18}
```

После получения долгосрочного токена для конкретного пользователя, у приложения появляется возможность воспользоваться доступом к сервису hh.ru. Получим все существующие резюме человека и выведем их на страницу. Для этого отправим GET запрос с токеном авторизации по заданному маршруту и вернём данные в формате JSON.

```
1 const token = searchParams.get('token');
2 if (token != null) {
3   var myHeaders = new Headers();
4   myHeaders.append("Authorization", `Bearer ${token}`);
```

```

5   try {
6     const res = await fetch("https://api.hh.ru/resumes/mine", {
7       method: 'GET', headers: myHeaders, redirect: 'follow' });
8     const result = await res.json();
9     return NextResponse.json(result);
10  }

```

## 4.9 Реализация страниц сравнения и обновления резюме

Страница сравнения навыков представляет из себя таблицу, содержащую наименование поля, значения совпадающих ключей для каждого из входных резюме и поле взаимодействия, содержащее возможность добавления собственного значения, а также удаления строки. Так как резюме содержит в себе значениях разных типов, их отображение будет возможно только при разделении. Для этого реализуем компонент сравнения, получающий значение свойства из JSON и далее в случае одиночной строки возвращающий тег с его содержимым, в случае списка строк обрабатывающий каждую из них, присваивающую уникальный индекс отображения, а в случае более глубокой вложенности добавляющей перед элементом его наименование.

```

1 if (typeof props.cv_editor == "string") {
2   cv_editor_item_list = <div>{props.cv_editor}</div>;
3 } else {
4   Object.keys(props.cv_editor).map((element, index) => {
5     let arrObj = props.cv_editor[element];
6     if (typeof arrObj == "object") {
7       cv_editor_item_list = props.cv_editor.map((element: any, index: any) => {
8         return (
9           {Object.keys(element).map((el, index) => {
10             return( <div key={index}>{el}: {element[el]}</div> );})}
11         );
12       } else {
13         cv_editor_item_list = props.cv_editor.map((element: any, index: any) =>
14           { return {element}; })
15     });
16   });
17 }

```

Итоговый вид страницы сравнения навыков показан на рисунке 4.9.

В серверной части платформы реализуем API метод для обновления данных резюме на сервисе hh.ru. Для этого в файле api/hh/route.tsx опишем PUT запрос, в заголовок которого передаётся токен пользователя и ссылка на идентификатор резюме, а в тело наименование поля данных с его содержимым. После

Field Name	CV Editor	hh.ru	Custom
Full Name	<input checked="" type="radio"/> Max Kulakov	<input type="radio"/> Max Kulakov	<input type="radio"/> <a href="#">Add item</a> <a href="#">Delete</a>
Description	<input checked="" type="radio"/> Hi! I'm Max Kulakov and I'm Designer	<input type="radio"/> Hi! I'm Max Kulakov and I'm Designer	<input type="radio"/> <a href="#">Add item</a> <a href="#">Delete</a>
Social Links	<input checked="" type="radio"/> <a href="https://github.com/nescbox/TIC-80">https://github.com/nescbox/TIC-80</a> <a href="https://www.behance.net/KulakovMax">https://www.behance.net/KulakovMax</a> <a href="https://t.me/MaxKulakov">https://t.me/MaxKulakov</a> <a href="https://discord.com/servers">https://discord.com/servers</a>	<input type="radio"/> <a href="https://github.com/nescbox/TIC-80">https://github.com/nescbox/TIC-80</a> <a href="https://www.behance.net/KulakovMax">https://www.behance.net/KulakovMax</a> <a href="https://t.me/MaxKulakov">https://t.me/MaxKulakov</a> <a href="https://discord.com/servers">https://discord.com/servers</a>	<input type="radio"/> <a href="#">Add item</a> <a href="#">Delete</a>
About	<input checked="" type="radio"/> I am currently a senior designer at YKSoft. I live and work in Saratov and pursue a Master's degree in Computer Science at the Saratov State University.	<input type="radio"/> I am currently a senior designer at YKSoft. I live and work in Saratov and pursue a Master's degree in Computer Science at the Saratov State University.	<input type="radio"/> <a href="#">Add item</a> <a href="#">Delete</a>

Рисунок 4.9 – Страница сравнения навыков

выполнения запроса в ответ от внутреннего сервера вернётся JSON с кодом и сообщением об успешности выполненной операции.

```

1 export async function PUT(req: Request) {
2   if (resume != null) {
3     const body = await req.json()
4     var raw = JSON.stringify( body );
5     try { const res = await fetch(`https://api.hh.ru/resumes/${resume}`,{`  

6       method: 'PUT', headers: myHeaders,  

7       body: raw, redirect: 'follow' })
8       return NextResponse.json({ message: "Resume has been updated" });
9     }`}

```

Реализуем страницу обновления навыков, для которой запросим и выведем данные из резюме hh.ru в виде списка тегов с возможностью их удаления из списка. Для этого создадим страницу и инициализируем начальное состояние значений и функцию удаления, проверяющую количество элементов и в случае их полного удаления возвращающую к исходному значению.

```

1 export default function ResumePage() {
2   const [skill_set, setSkill_set] = React.useState(initialSkills);
3   const handleClose = (skillToRemove:any) => {
4     setSkill_set(skill_set.filter((skill: any) =>
5       skill !== skillToRemove));
6     if (skill_set.length === 1) {setSkill_set(initialSkills);}
7   };

```

Добавим кнопку обновления резюме, по нажатию на которую вызывается функция-обработчик массива строк с преобразованием их к объектному формату.

ту JSON и помещающая итоговое значение в тело PUT запроса, отправляющего сообщение через внутренний сервер на сервис hh.ru.

```
1 const skills_body = JSON.parse(JSON.stringify
2     (`{"skill_set": [${skill_set.map((x: any) => `"${x}"`)}]}`))
3 return (
4     skill_set.map((skill: any, index: any) =>
5         <Chip key={index} onClose={() => handleClose(skill)} variant="flat">
6             {skill}
7         </Chip>
8         <Button onClick={updateResume}>Обновить резюме</Button>
9     )))

```

Итоговый вид страницы обновления навыков, содержащей статус изменения списка, редактируемое поле ввода и теги представлен на рисунке 4.10.

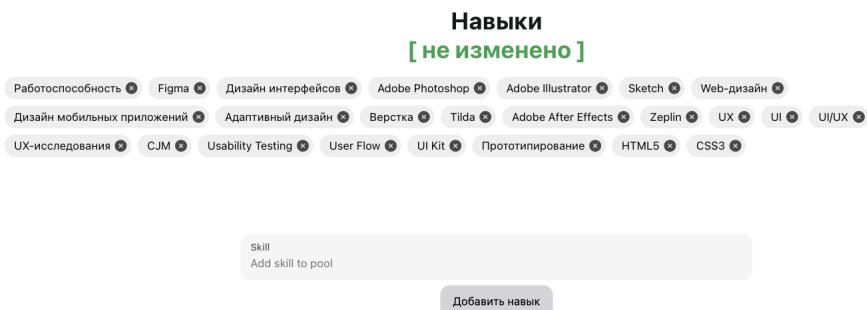


Рисунок 4.10 – Страница обновления навыков

## 4.10 Реализация страницы процесса разработки

Для написания страницы необходимо создать структуру данных в текстовом формате JSON, содержащую в себе следующие поля:

1. Type – тип карточки, делится на работу, находящуюся в процессе и уже выполненную;
2. Version – порядковый номер завершённой работы, может принимать отсутствующее значение в случае текущего выполнения, иначе соответствующее семантическому версионированию, например 1.2.3, где:
  - a) 1 – характеризует мажорную версию, в которой происходит значительная переработка поведения, функционала и кода, зачастую с потерей совместимости с предыдущей мажорной версией;

- б) 2 – соответствует минорной версии, в которой добавляется новый функционал или вносятся изменения в уже существующий проект;
  - в) 3 – является патч версией, включающей исправление ошибок или незначительную переработку кода;
3. Date – дата публикации выполненной работы;
  4. Task-list – список звершённых задач, являющийся массивом строк;
  5. Author – профиль автора, включающий в себя следующие пункты:
    - а) Thumbnail – миниатюра профиля;
    - б) Name – имя или псевдоним разработчика;
    - в) Team – команда проекта.

Создадим функцию getDate, принимающую дату в международном формате уууу-мм-дд и преобразующую её в новый объект с типом данных «Дата», обрезанным от лишних для отображения значений дня недели и времени, и читаемым текстовым видом без дефисов.

```

1 function getDate(date: string) {
2   const temp_date = new Date(date).toUTCString().slice(5, 16);
3   return temp_date;
4 }

```

Получим список публикаций и запишем его в константу, предварительно выполнив сортировку по номеру версии. Для этого с помощью стрелочной функции получаем значение у ближайшей пары и сравниваем записанные значения, после чего наименьшее помещаем в конец. Если номера нет и работа находится в процессе, то данный элемент встаёт в начало списка.

```

1 const release_list = require("@/data-template/release-data.json").sort(
2   (a:any, b:any) => {if (a.version > b.version) { return -1; }})
3 )

```

После этого необходимо отобразить обработанные данные. Для этого создадим контейнер с сеткой, колонками, отступами между блоками и полной шириной, и последовательно передадим в карточку каждого элемента его порядковый номер в списке для унификации компонентов и сами данные.

```

1 <div className="gap-4 grid grid-cols-1 sm:grid-cols-4 mt-4 w-full">
2   {release_list.map((item:any, index:any) => (...))}

```

Опишем компоненты карточки, состоящей из тега Card, шапки карточки, содержащей проверку на тип публикации и в зависимости от этого выводящее

версию или отображение текущей работы, вместе с соответствующей датой. После разделителя внутри центральной части карточки идёт перечисление всех задач в списке, и в конце миниатюра с именем и командой автора. Полный код реализации страницы процесса разработки можно найти в приложении Г.

```
1 <Card key={index}>
2   <CardHeader className="container">
3     {item.type == "WIP"?
4       <Chip color="primary" variant="bordered">WIP</Chip> :
5       <Chip color="primary" variant="solid">v{item.version}</Chip>}
6   </CardHeader><Divider/><CardBody>...</CardBody><Divider/>
7   <CardFooter>
8     <Image
9       isZoomed isBlurred radius="full"
10      width={40} height={40}
11      src={item.author.thumbnail}/>
12      {item.author.name}{item.author.team}
13    </CardFooter>
14  </Card>
```

После написания структуры хранения, функций обработки даты и сортировки, вёрстки и стилей, страница принимает вид, представленный на рисунке 4.11.

The screenshot shows a 'Releases' page with four cards:

- WIP** 16 May 2024: In Progress tasks include MongoDB Implementation, Save auth users, Save hh user token in db, Save Releases in DB, Save user cv in DB, Personal cv page, and Personal update cv page. Author: Max Kulakov, CV Editor Team.
- WIP** 16 May 2024: In Progress tasks include Create CV Editor Logo and Create CV Editor Identity. Author: Valeria Zevakina, CV Editor Team.
- v0.1.2** 05 May 2024: Release Note tasks include Add Release Card Component, Structure JSON to save Release data, and Add sorting to Release data. Author: Max Kulakov, CV Editor Team.
- v0.1.1** 04 May 2024: Release Note tasks include Add Guide Page and Add Releases Page. Author: Max Kulakov, CV Editor Team.

Рисунок 4.11 – Страница процесса разработки

## 4.11 Публикация платформы единого резюме

Перед публикацией необходимо изменить файл настроек проекта, указав точку входа и конфигурацию скрипта в файле package.json:

```
1 "scripts": {
2   "build": "next build",
3   "start": "next start"
4 },
```

Так как платформа работает в режиме сервера – публикация в виде статичной страницы не представляется возможным, поэтому воспользуемся хостингом Vercel, позволяющим реализовать данную потребность. Первоначально необходимо загрузить свой код в систему контроля версий GitHub, после чего связать аккаунт с аккаунтом хостинга, синхронизировать репозитории и настроить параметры публикации. В итоговом виде платформа соберётся в оптимизированный для получения пользователем вид, закэшируется в папке dist и будет выдаваться браузеру при обращении по ссылке. Настройки публикации сайта в Vercel представлены на рисунке 4.12.

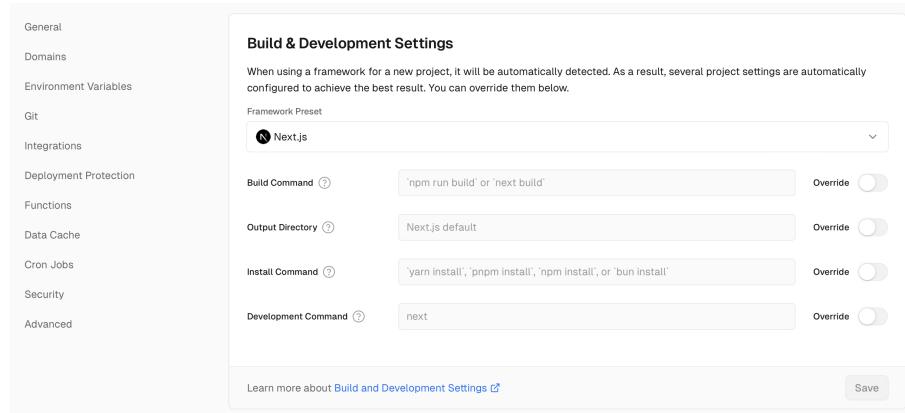


Рисунок 4.12 – Настройка публикации сайта

После написания кода мы получили готовый сервис, в котором реализован весь запланированный функционал: от базовых элементов и компонентов до серверной части с авторизацией и обновлением резюме.

## **ЗАКЛЮЧЕНИЕ**

В результате проведения исследовательской работы были приобретены навыки анализа качества и эффективности научной литературы в областях веб-разработки, разработки сервисов с автоматическим обновлением данных, а также навык анализа конкурентных платформ для создания резюме. Анализ статей по составлению резюме, социологическому подходу к этой теме и инструментам веб-разработки позволил сформулировать требования для разработки единой платформы резюме.

Проведенный анализ конкурентных платформ выявил недостатки существующих сервисов, что послужило основой для разработки собственной платформы. Данное решение включает серверную и клиентскую части приложения, а также подразумевает публикацию проекта на серверном хостинге для эффективного взаимодействия с функциональностью платформы.

Результаты данной работы были представлены на студенческой научной конференции факультета КНиИТ в апреле 2024 года.

По итогам выполнения исследовательской работы все поставленные цели и задачи были успешно достигнуты. Были разработаны собственные методы разработки и масштабирования единой платформы резюме. В качестве объектов анализа научной литературы для реализации данного сервиса выступили статьи по темам разработки клиент-серверного приложения средствами библиотек Next JS, TailWind и применением архитектуры Rest API. С учётом проведённого анализа научной литературы были составлены базовые технические требования для масштабирования уже существующего приложения.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 *Косолапова, Т. В.* Типологические особенности современного резюме на английском языке / Т. В. Косолапова // *Поволжский педагогический вестник*. — 2020. — Т. 8, № 2 (27). — С. 30–35.
- 2 *Шинкаренко, Е. А.* Резюме как элемент практик поиска работы молодежью / Е. А. Шинкаренко // *Вестник Нижегородского университета им. Н. И. Лобачевского. Серия: Социальные науки*. — 2022. — № 2. — С. 135–140.
- 3 *Шинкаренко, Е. А.* Сценарии поиска работы студентами и выпускниками социоэкономических специальностей / Е. А. Шинкаренко // *Вестник Пермского национального исследовательского политехнического университета. Социально-экономические науки*. — 2021. — № 2. — С. 35–50.
- 4 *Разин, С. А.* Что должен знать начинающий Node.js разработчик / С. А. Разин // *E-Scio*. — 2020. — Т. 9, № 48.
- 5 *Холмс, С.* Стек MEAN. Mongo, Express, Angular, Node / С. Холмс; Под ред. Н. Гринчик. — Санкт-Петербург: Питер, 2023.
- 6 *Прохоренок, Н. А.* Bootstrap и CSS-препроцессор Sass. Самое необходимое / Н. А. Прохоренок. — Санкт-Петербург: БХВ, 2021.
- 7 *Ефремов, В. М.* Разработка веб-сайта с использованием Bootstrap / В. М. Ефремов // *Инновационная наука*. — 2020. — № 6. — С. 40–42.
- 8 *Безрук, П. А.* Разработка системы распределенного мониторинга компьютерной сети на основе Rest API / П. А. Безрук // *Актуальные проблемы авиации и космонавтики*. — 2022. — Т. 2, № 13. — С. 94–95.
- 9 *Миковски, М.* Разработка одностраничных веб-приложений / М. Миковски, П. Джош; Под ред. А. А. Слинкин. — Москва: ДМК Пресс, 2020.
- 10 *Файн, Я.* Angular и TypeScript. Сайтостроение для профессионалов / Я. Файн, А. Моисеев. — Санкт-Петербург: Питер, 2022.
- 11 *Гарин, М. А.* Создание Workflow аналитической платформы KNIME для анализа данных на примере вакансий сайта HeadHunter / М. А. Гарин, Д. К. Егорова, С. Ф. Сайфетдинов // *Огарёв-Online*. — 2021. — Т. 12, № 165. — С. 1–6.

- 12 API HeadHunter. Общая информация [Электронный ресурс]. — URL: <https://github.com/hhru/api/blob/master/docs/general.md> (дата обращения 28.04.2024). Загл. с экрана. Яз. рус.
- 13 Грузин, Н. А. Обзор и сравнение хостингов для git-репозиториев: Bitbucket, Github и Gitlab / Н. А. Грузин, А. А. Голубничий // E-Scio. — 2021. — № 1 (52). — С. 588–596.
- 14 Архипов, Л. В. Концепция применения контентно-адресуемых систем хранения для контроля версий / Л. В. Архипов // Инновации в науке. — 2015. — № 12 (49). — С. 42–46.
- 15 Чакон, С. Git для профессионального программиста / С. Чакон, Б. Штрауб. — Санкт-Петербург: Питер, 2017.
- 16 Bitbucket vs Github vs Gitlab Infographics [Электронный ресурс]. — URL: <https://www.educba.com/bitbucket-vs-github-vs-gitlab/> (дата обращения 28.04.2024). Загл. с экрана. Яз. англ.
- 17 О компании HeadHunter [Электронный ресурс]. — URL: <https://saratov.hh.ru/article/28> (дата обращения 28.04.2022). Загл. с экрана. Яз. рус.
- 18 Гриднева, М. А. Особенности работы с персоналом в it-сфере / М. А. Гриднева, М. Ю. Федосова // Телескоп. — 2021. — № 1. — С. 106–112.
- 19 Пишем резюме на GitHub. Краткий гайд [Электронный ресурс]. — URL: <https://javarush.com/groups/posts/3424-pishem-rezjume-na-github> (дата обращения 28.04.2024). Загл. с экрана. Яз. рус.
- 20 Создание сайта с резюме [Электронный ресурс]. — URL: <https://tilda.cc/ru/tpls/sozdanie-saita-rezume/> (дата обращения 28.04.2024). Загл. с экрана. Яз. рус.
- 21 Разница между React JS и React Native [Электронный ресурс]. — URL: <https://itanddigital.ru/react> (дата обращения 05.05.2024). Загл. с экрана. Яз. рус.
- 22 ReactJS: философия, области применения, требования к бэкенду и всё, что нужно знать о технологии сейчас [Электронный ресурс]. — URL: <https://vc.ru/dev/567245> (дата обращения 05.05.2024). Загл. с экрана. Яз. рус.

- 23 Ершов, Т. А. Обзор современных технологий разработки для web-приложений / Т. А. Ершов // *E-Scio*. — 2023. — № 8 (83). — С. 38–42.
- 24 Чепрасов, И. А. Оценка целесообразности использования Next.js в React-приложениях / И. А. Чепрасов // *Научно-технические инновации и веб-технологии*. — 2023. — № 2. — С. 60–65.
- 25 Никитин, А. А. Использование фреймворка Next.js в современных веб-приложениях / А. А. Никитин, Д. А. Носов // *Информационно-телекоммуникационные системы и технологии*. — 2022. — С. 136–137.
- 26 Обзор фреймворка TailwindCSS: чем он хорош и кому будет полезен [Электронный ресурс]. — URL: <https://timeweb.com/ru/community/articles/chto-takoe-tailwindcss-zachem-nuzhen-i-chem-horosh> (дата обращения 05.05.2024). Загл. с экрана. Язык: русский.
- 27 Rifandi, F. Website Gallery Development Using Tailwind CSS Framework / F. Rifandi, T. V. Adriansyah, R. Kurniawati // *Jurnal E-Komtek (Elektro-Komputer-Teknik)*. — 2022. — Т. 6, № 2. — С. 205–214.
- 28 Калинин, Н. А. Внедрение технологий взаимодействия между микросервисами и внешними клиентами при проектировании информационных систем / Н. А. Калинин, А. И. Архипова, Г. М. Тулегенов // *Информационные технологии в науке, промышленности и образовании*. — 2023. — С. 97–101.
- 29 Masse, M. REST API design rulebook / M. Masse. — O'Reilly Media, 2011. — С. 94.
- 30 What is Postman? [Электронный ресурс]. — URL: <https://www.postman.com/product/what-is-postman> (дата обращения 12.05.2024). Загл. с экрана. Язык: английский.
- 31 Маринчук, А. С. Тестирование различных API сервисов с помощью инструмента Postman / А. С. Маринчук // *Постулам*. — 2020. — Т. 1, № 51. — С. 115–123.
- 32 Introduction to Postman for API development [Электронный ресурс]. — URL: <https://www.geeksforgeeks.org/introduction-postman-api-development/> (дата обращения 12.05.2024). Загл. с экрана. Язык: английский.

- 33 What is MongoDB [Электронный ресурс]. — URL: <https://www.mongodb.com/company/what-is-mongodb> (дата обращения 10.05.2024). Загл. с экрана. Яз. англ.
- 34 Data Modeling Introduction [Электронный ресурс]. — URL: <https://www.mongodb.com/docs/v5.0/core/data-modeling-introduction/> (дата обращения 10.05.2024). Загл. с экрана. Яз. англ.
- 35 Кокоулин, А. Н. Анализ уязвимостей современных NoSQL БД Redis, mongodb, Clickhouse и методы их защиты / А. Н. Кокоулин, Н. А. Гагарин // Автоматизированные системы управления и Информационные технологии. — 2022. — Т. 2, № 1. — С. 201–204.
- 36 Белодед, Н. И. Контейнеризация и использование Docker в веб - разработке / Н. И. Белодед, А. А. Юрьев // Взаимодействие науки и общества - путь к модернизации и инновационному развитию. — 2023. — Т. 2, № 1. — С. 68–69.
- 37 Документация Docker [Электронный ресурс]. — URL: <https://docs.docker.com> (дата обращения 15.05.2024). Загл. с экрана. Яз. англ.
- 38 Краткий экскурс в Docker [Электронный ресурс]. — URL: [https://www.nic.ru/help/kratkij-ekskurs-v-docker\\_11188](https://www.nic.ru/help/kratkij-ekskurs-v-docker_11188) (дата обращения 13.05.2024). Загл. с экрана. Яз. англ.
- 39 Официальный сайт Vercel [Электронный ресурс]. — URL: <https://vercel.com/> (дата обращения 13.05.2024). Загл. с экрана. Яз. англ.
- 40 Литвинцев, П. А. Сравнительный анализ подходов к хостингу односторонних и классических веб - приложений в образовательных целях / П. А. Литвинцев // Вестник науки. — 2023. — Т. 64, № 7. — С. 227–231.
- 41 MURAL – Case Study [Электронный ресурс]. — URL: <https://vercel.com/customers/mural> (дата обращения 13.05.2024). Загл. с экрана. Яз. англ.

## ПРИЛОЖЕНИЕ А

### Реализация подключения к MongoDB

```
1 import NextAuth from "next-auth";
2 import { MongoDBAdapter } from "@auth/mongodb-adapter";
3 import clientPromise from "./lib/db";
4 import { MongoClient, ServerApiVersion } from "mongodb";
5
6 const { handlers, auth, signIn, signOut } = NextAuth({
7     adapter: MongoDBAdapter(clientPromise),
8 });
9
10 if (!process.env.MONGODB_URI) {
11     throw new Error('Invalid/Missing environment variable: "MONGODB_URI"');
12 }
13
14 const uri = process.env.MONGODB_URI;
15 const options = {
16     serverApi: {
17         version: ServerApiVersion.v1,
18         strict: true,
19         deprecationErrors: true,
20     },
21 };
22
23 let client; let clientPromise: Promise<MongoClient>;
24
25 if (process.env.NODE_ENV === "development") {
26     let globalWithMongo = global as typeof globalThis & {
27         _mongoClientPromise?: Promise<MongoClient>
28     };
29     if (!globalWithMongo._mongoClientPromise) {
30         client = new MongoClient(uri, options);
31         globalWithMongo._mongoClientPromise = client.connect();
32     }
33     clientPromise = globalWithMongo._mongoClientPromise;
34 } else {
35     client = new MongoClient(uri, options);
36     clientPromise = client.connect();
37 };
38
39 export default clientPromise;
```

## ПРИЛОЖЕНИЕ Б

### Реализация компонента отображения опыта

```
1 "use client";
2 import * as React from "react";
3
4 function diffDates(day_one:any, day_two:any) {
5     return (day_one - day_two) / (60 * 60 * 24 * 1000);
6 };
7
8 function getFormatedStringFromDays(numberOfDays:number) {
9     numberOfDays = Math.abs(numberOfDays)
10    var years = Math.floor(numberOfDays / 365);
11    var months = Math.floor(numberOfDays % 365 / 30);
12    var yearsDisplay = years > 0 ?
13        years + (years == 1 ? " year " : " years ") : "";
14    var monthsDisplay = months > 0 ?
15        months + (months == 1 ? " month " : " months ") : "";
16    return yearsDisplay + monthsDisplay;
17 }
18
19 interface ExperienceItemProps {
20     name: string;
21     description: string;
22     start_date: string;
23     end_date: string;
24     current_date?: boolean;
25 }
26
27 interface ExperienceProps {
28     experience_list: ExperienceItemProps[];
29 }
30
31 export const Experience = (props : ExperienceProps) => {
32     const experience_item_list = props.experience_list.map((element, index) => {
33         const date1 = new Date(element.start_date);
34         const date2 = element.current_date ?
35             new Date() : new Date(element.end_date)
36
37         return (
38             <div key={index} className="grid grid-cols-12 gap-2 mb-4">
39                 <div className="col-start-1 col-end-3">
```

```
40     {element.start_date} - {element.end_date}
41     <div className="text-default-500">
42         {getFormatedStringFromDays(diffDates(date1, date2))}
43     </div>
44     </div>
45     <div className="col-start-4 col-end-6">
46         {element.name}
47     </div>
48     <div className="col-start-7 col-end-13">
49         {element.description}
50     </div>
51     </div>
52 )
53 }
54 return <>{experience_item_list}</>
55 };
```

## ПРИЛОЖЕНИЕ В

### Реализация страницы резюме

```
1 "use client";
2 import React from "react";
3 import {Image} from "@nextui-org/react";
4 import { SocialLink } from "@components/social-link";
5 import { Experience } from "@components/experience";
6 import { Projects } from "@components/projects";
7 import { About } from "@components/about";
8
9 export default function Template1Page() {
10 const profile_data = require("@/data-template/template-1-data.json");
11 return (
12     <h1 className="h6 text-uppercase text-center pb-4">
13         {profile_data.name}
14     </h1>
15     <p className="text-uppercase text-center pb-2">
16         {profile_data.description}
17     </p>
18     <SocialLink links = {profile_data.social_links}/>
19     <div className="grid grid-cols-8 gap-4 mb-12">
20         <div className="col-start-3 col-end-7">
21             <Image
22                 isZoomed
23                 isBlurred
24                 alt={profile_data.name + " Image"}
25                 src={profile_data.profile_image}
26             />
27         </div>
28     </div>
29     <div className="container pt-5 pb-5 text-left">
30         <h2 className="h6 uppercase text-end pb-4" id="about">
31             about me
32         </h2>
33         <About about_text={...profile_data.about}/>
34     </div>
35     <div className="container pt-5 pb-5 text-left ">
36         <h2 className="h6 uppercase text-end pb-4" id="experience">
37             education
38         </h2>
39         <Experience experience_list={...profile_data.education}/>
```

```
40     </div>
41     <div className="container pt-5 pb-5 text-left ">
42         <h2 className="h6 uppercase text-end pb-4" id="experience">
43             experience
44         </h2>
45         <Experience experience_list={...profile_data.experience}/>
46     </div>
47     <div className="container pt-5 pb-5">
48         <h2 className="h6 uppercase text-end pb-4" id="projects">
49             projects by companies
50         </h2>
51         <Projects projects_list={...profile_data.projects}/>
52     </div>
53 ) ;}
```

## ПРИЛОЖЕНИЕ Г

### Реализация страницы процесса разработки

```
1 "use client";
2
3 import { title } from "@/components/primitives";
4 import { Avatar, Checkbox, CheckboxGroup, Divider, Image } from
5   "nextui-org/react";
6 import { Card,CardBody,CardFooter,CardHeader } from "@nextui-org/card";
7 import {Chip} from "@nextui-org/chip";
8
9
10 function getDate(date: string) {
11   const temp_date = new Date(date).toUTCString().slice(5, 16);
12   return temp_date;
13 }
14
15 export default function ReleasesPage() {
16   const current_date = new Date().toUTCString().slice(5, 16);
17   const release_list =
18     require("@/data-template/release-data.json").sort((a:any, b:any) => {
19       if (a.version > b.version) {
20         return -1;
21     }
22   });
23   return (
24     <div>
25       <h1 className={title()}>Releases</h1>
26       <div className="gap-4 grid grid-cols-1 sm:grid-cols-4 mt-4
27         w-full">
28         {release_list.map((item:any, index:any) => (
29           <Card key={index}>
30             <CardHeader className="container">
31               {item.type == "WIP"?
32                 <Chip color="primary"
33                   variant="bordered">WIP</Chip>
34                 :
35                 <Chip color="primary"
36                   variant="solid">v{item.version}</Chip>
37               }
38               {item.type == "WIP"?
39                 <p className="text-gray-500
40                   pl-3">{current_date}</p>
41                 :
42                 <Image alt="Placeholder image" width="100%" height="100%"/>
43               }
44             </CardHeader>
45             <CardBody>
46               <div>
47                 <Avatar>
48                   <Image alt="User profile picture" />
49                 </Avatar>
50                 <div>
51                   <strong>User Name</strong>
52                   <small>Role: Admin</small>
53                 </div>
54               </div>
55             </CardBody>
56           </Card>
57         )
58       )
59     )
60   )
61 }
```

```

34             <p className="text-gray-500
35                 ↵   pl-3">{getDate(item.date)}</p>
36         }
37     </CardHeader>
38     <Divider/>
39     <CardBody>
40         <div className={item.type == "WIP"?
41             ↵   "text-gray-500" : ""}>
42             {item.type == "WIP"?
43                 <h4>In Progress</h4>
44                 :
45                 <h4>Release Note</h4>
46             }
47         <ul className="list-disc">
48             {item.task_list.map((item:any, index:any)
49                 ↵   => (
50                     <li key={index}>{item}</li>
51                 )));
52         </ul>
53     </CardBody>
54     <Divider/>
55     <CardFooter>
56         <Image
57             isZoomed isBlurred radius="full"
58             width={40} height={40}
59             className="aspect-square"
60             src={item.author.thumbnail}
61         />
62         <div className="container pl-3 text-start">
63             <p>{item.author.name}</p>
64             <p className="text-sm
65                 ↵   text-gray-500">{item.author.team}</p>
66         </div>
67     </CardFooter>
68     </Card>
69     ))})
70 </div>
71 </div>
72 );
73 }

```