

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

УТВЕРЖДАЮ

Зав.кафедрой,

к. ф.-м. н.

\_\_\_\_\_ М. В. Огнева

**ОТЧЕТ О ПРАКТИКЕ**

студента 2 курса 273 группы факультета КНиИТ  
Кулакова Максима Сергеевича

вид практики: производственная (научно-исследовательская работа)

кафедра: Информатики и программирования

курс: 2

семестр: 1

продолжительность: 18 нед., с 01.09.2023 г. по 14.01.2024 г.

Руководитель практики от университета,

к. э. н., доцент

\_\_\_\_\_

Л. В. Кабанова

Руководитель практики от организации (учреждения, предприятия),

к. э. н., доцент

\_\_\_\_\_

Л. В. Кабанова

Тема практики: «Масштабирование платформы создания единого резюме»

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
1 Теоретические аспекты разработки .....	5
1.1 Интерфейсная библиотека React JS .....	5
1.2 Серверная библиотека NextJS .....	6
1.3 Интерфейсная библиотека TailWind .....	7
1.4 Архитектурный стиль Rest API .....	8
2 Реализация платформы единого резюме .....	10
2.1 Общие настройки динамического приложения .....	10
2.2 Написание механизма авторизации .....	11
2.3 Реализация страницы резюме .....	13
2.4 Взаимодействие с сервисом hh.ru .....	17
2.5 Реализация страниц сравнения и обновления резюме .....	18
ЗАКЛЮЧЕНИЕ .....	22
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	23
Приложение А Реализация компонента отображения опыта .....	25
Приложение Б Реализация страницы резюме .....	27

## ВВЕДЕНИЕ

Современная веб-разработка требует надежных, интерактивных и масштабируемых решений, одним из которых является клиент-серверная разработка. Данная архитектура не только обеспечивает эффективное управление получаемыми и обрабатываемыми данными, но и позволяет организовать бесперебойную работу со множеством пользователей.

На данный момент существует множество мощных инструментов, способных разработать веб-приложение, соответствующее всем необходимым минимальным требованиям. Одними из таких популярных разработок являются ReactJS для разработки надежных динамических пользовательских интерфейсов на основе виртуального DOM, архитектурное решение Rest API, прекрасно дополняющий и обеспечивающий стандартизированный подход к разработке веб-сервисов, и TailWind CSS, способный заменить неоптимизированное стилевое оформление в более приятные компоненты.

Разработчики могут обеспечить масштабируемость, модульность и совместимость своих приложений, придерживаясь принципов REST.

Цель научно-исследовательской работы заключается в масштабировании платформы создания единого резюме до клиент-серверного приложения.

В соответствии с поставленной целью можно выделить следующие задачи исследования:

1. Обзор технических средств реализации приложения, включающий в себя React, Next JS, TailWind, Rest API;
2. Написание и конфигурирование основного функционала приложения;
3. Реализация взаимодействия платформы с существующими сервисами;
4. Подведение итогов проведенной научно-исследовательской работы.

## **1 Теоретические аспекты разработки**

### **1.1 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. [1] [2]

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 [3].

React очень удобно рассматривать для разработки одностраничных при-

ложений. Несмотря на то, что при снятии нагрузки с сервера загружается 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 догрузить то, чего не хватает.

По состоянию на конец 2023 года React находится в лидирующих позициях среди JavaScript-технологий для разработки фронтенда [2].

## **1.2 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 [4].

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 тандеме с SSR и позволяет трекерам сканировать больше контента;
3. Упрощенный процесс разворачивания приложения [5].

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.

### **1.3 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 [6].

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 [7].

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. Продвинутое 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 [6].

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 [8].

#### **1.4 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 [9].

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 взаимоза-



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. [10]

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.

## 2 Реализация платформы единого резюме

### 2.1 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 );}
```

## 2.2 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 введенных в поля данных и в случае соответствия вернём самого пользователя с его именем, фотографией профиля и ролью [7].

```
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        return userWithoutPass as User;  
16      }  
17    },  
18    return null })  
19  ], 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

```

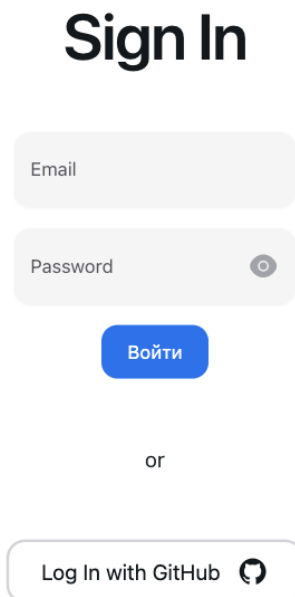
Кнопка авторизации через Github работает через параметры поисковой строки, получаемые в процессе возврата на страницу платформы после успешного входа на стороне внешнего сервиса.

```

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 внутри всей иерархической оболочки страниицы передадим заголовок вместе с компонентами `SignInForm` и `GithubButton`, после которых страница будет располагаться по маршруту `app/signIn`. Итоговый вид страницы авторизации отображён на рисунке 2.1.



The image shows a 'Sign In' form. At the top, the text 'Sign In' is displayed in a large, bold, black font. Below it, there are two input fields: 'Email' and 'Password'. The 'Email' field is a simple light gray box. The 'Password' field is a light gray box with a small eye icon on the right side to toggle visibility. Below these fields is a blue button with the text 'Войти' (Login) in white. Underneath the button is the word 'or' in a small, gray font. At the bottom, there is a rounded rectangular button with the text 'Log In with GitHub' and the GitHub logo icon.

Рисунок 2.1 – Страница авторизации

### 2.3 Реализация страницы резюме

Для написания модуля необходимо создать структуру данных резюме. Информация о профиле хранится в формате 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:
  - а) 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;
  - з) 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:
- а) 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 }
9 export const SocialLink = (props : SocialProps) => {
10   const icon_size, icon_color = 36, "text-default-500";
11   const social_icons = props.links.map((element, index) => {
12     var url = new URL(element.toString());
13   return
14     <Link isExternal href={element.toString()} key={index}>
15       {IconVariant(url.host.toString(), icon_color, icon_size)}
16     </Link>;
17   });
18 }
```

Итоговый вид компонента показан на рисунке 2.2.

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

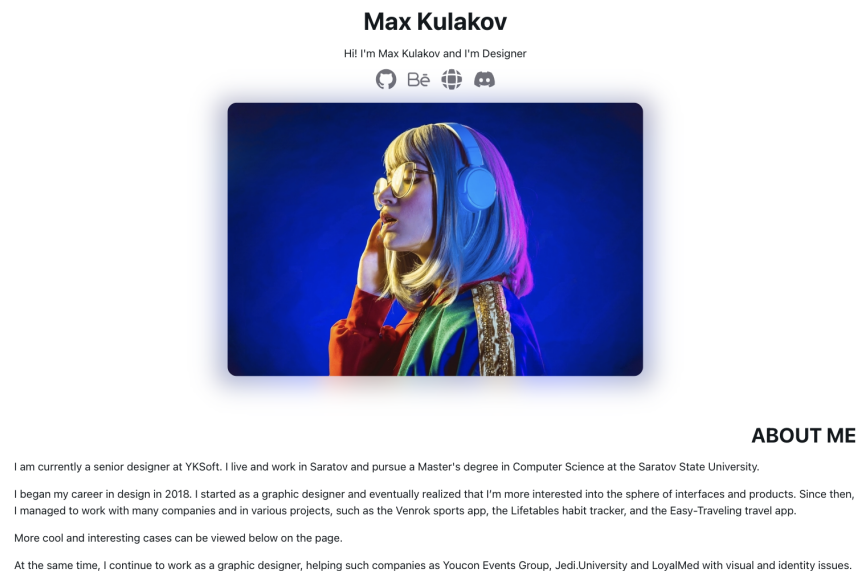


Рисунок 2.2 – Компонент отображения контактов и ссылок

месяцев и лет. Результатом работы функций станет вывод временного значения в строке компонента. Полный код реализации компонента можно найти в приложении А.

```

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))}})
14     return <>{experience_item_list}</>
15 };

```

Итоговый вид компонента показан на рисунке 2.3.

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.

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 2.3 – Компонент отображения опыта работы и образования

```

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 2.4.

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 Б.

```

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>);}

```



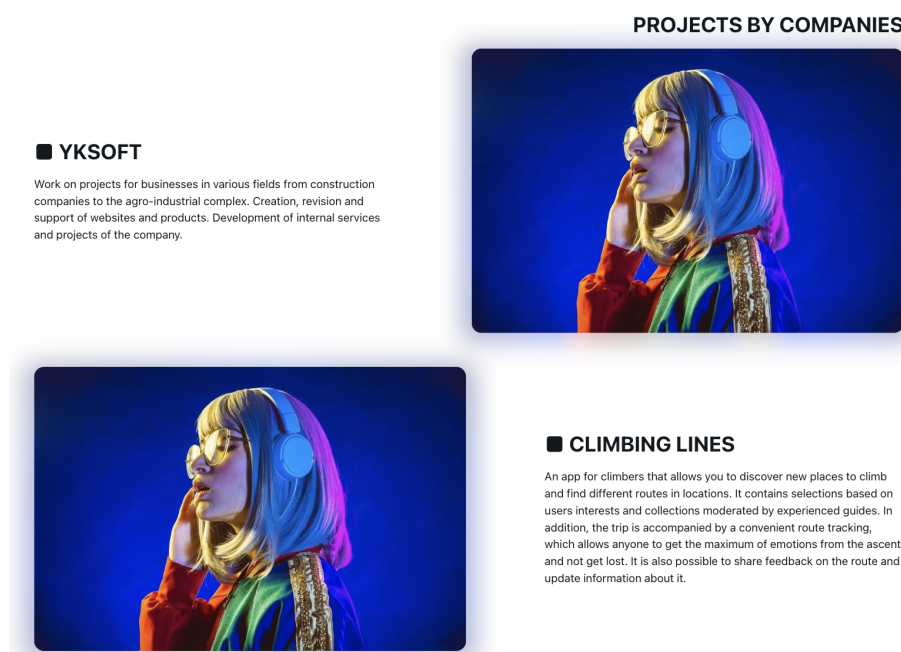


Рисунок 2.4 – Компонент отображения списка проектов

## 2.4 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. [11]

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, предварительно считывающий 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 [12].

```

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    }

```

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  }

```

## 2.5 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              });}

```

Итоговый вид страницы сравнения навыков показан на рисунке 2.5.

### HH Dev Page

Field Name	CV Editor	hh.ru	Custom
Full Name	<input checked="" type="radio"/> Max Kulakov	<input type="radio"/> Max Kulakov	<input type="radio"/> <span>Add item</span> <span>🗑</span>
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"/> <span>Add item</span> <span>🗑</span>
Social Links	<input checked="" type="radio"/> <a href="https://github.com/nesbox/TIC-80">https://github.com/nesbox/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/nesbox/TIC-80">https://github.com/nesbox/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"/> <span>Add item</span> <span>🗑</span>
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"/> <span>Add item</span> <span>🗑</span>

Рисунок 2.5 – Страница сравнения навыков

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

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 возвращающего 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 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 2.6.

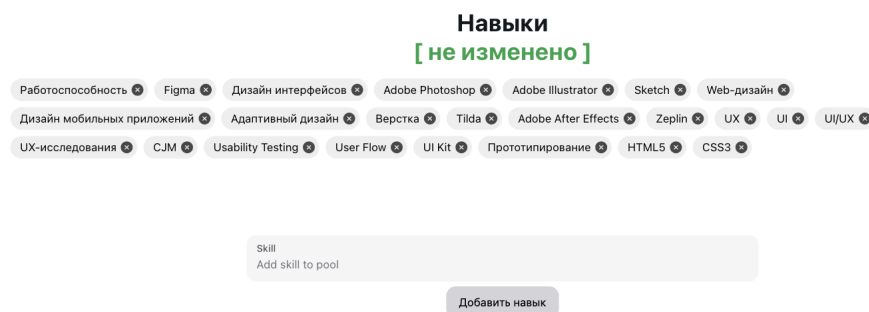


Рисунок 2.6 – Страница обновления навыков

После написания кода мы получили готовый сервис, в котором реализован весь запланированный на данном этапе 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 области разработки сервисов с автоматическим обновлением данных, достигнут навык анализа конкурентных платформ для создания резюме и сформулирован собственный метод масштабирования уже разработанной единой платформы резюме, тем самым было достигнуто полное выполнение поставленных задач.

В качестве объектов анализа научной литературы выступили статьи по темам разработки клиент-серверного приложения средствами библиотек React 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 Разница между React JS и React Native [Электронный ресурс]. — URL: <https://itanddigital.ru/react> (дата обращения 23.12.2023). Загл. с экр. Яз. рус.
- 2 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 24.12.2023). Загл. с экр. Яз. рус.
- 3 Ершов, Т. А. Обзор современных технологий разработки для web-приложений / Т. А. Ершов // *E-Scio*. — 2023. — № 8 (83). — С. 38–42.
- 4 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.
- 5 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.
- 6 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 24.12.2023). Загл. с экр. Яз. рус.
- 7 Файн, Я. Angular и TypeScript. Сайтостроение для профессионалов / Я. Файн, А. Моисеев. — Санкт-Петербург: Питер, 2022.
- 8 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.
- 9 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.
- 10 Masse, M. REST API design rulebook / M. Masse. — O'Reilly Media, 2011. — С. 94.

- 11 API HeadHunter. Общая информация [Электронный ресурс]. — URL: <https://github.com/hhru/api/blob/master/docs/general.md> (дата обращения 20.12.2023). Загл. с экр. Яз. рус.
- 12 Разин, С. А. Что должен знать начинающий Node.js разработчик / С. А. Разин // *E-Scio*. — 2020. — Т. 9, № 48.



## ПРИЛОЖЕНИЕ А

### 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 getFormattedStringFromDays(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             {getFormattedStringFromDays(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 Б

### 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 );}
```