

СОДЕРЖАНИЕ

Введение.....	4
1 Анализ технического задания.....	5
2 Выбор и обоснование языков программирования.....	8
3 Проектирование сайта и его интерфейса.....	10
Заключение	13
Список используемой литературы	15
Приложение А Листинг кода	16

ВВЕДЕНИЕ

Данный курсовой проект посвящен проектированию интерфейса системы обработки резюме. В настоящее время системы обработки резюме становятся неотъемлемой частью подбора персонала. Они обеспечивают удобный и эффективный способ сбора, оценки и анализа резюме соискателей для рекрутеров и работодателей.

Цель проекта заключается в разработке интуитивно понятного интерфейса, который оптимизирует процесс обработки резюме. Проект включает в себя анализ требований, проектирование пользовательского интерфейса, разработку механизмов автоматической обработки резюме, интеграцию с базой данных и тестирование системы.

В проекте будут использоваться современные технологии и инструменты, такие как языки программирования, фреймворки и библиотеки, специализированные на обработке и анализе текстовых данных, создании пользовательских интерфейсов и взаимодействии с базами данных.

Ожидаемым результатом проекта является разработка эффективного интерфейса системы обработки резюме, ускоряющего процесс подбора персонала и упрощающего работу рекрутеров.

1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Исходя из технического задания на курсовое проектирование, необходимо разработать интерфейс системы обработки резюме. В процессе проведения анализа, мной было проведено проектирование структуры системы, выбор основного стека технологий, а также начальные этапы дизайна макета веб-приложения.

После успешных первых шагов в анализе, был сделан вывод, что для создания решения, которое будет закрывать все главные потребности отдела подбора кадров, оно должно обладать следующим функционалом:

- Загрузка и скачивание резюме: Пользователи могут загружать резюме в формате CSV в систему, а также извлекать резюме интересующих их кандидатов в формате CSV из системы. Приложение обрабатывает и анализирует содержимое файла, извлекает соответствующие данные и хранит их в базе данных для дальнейшего использования.

- Удаление резюме: Пользователи имеют возможность удалять резюме из системы. Это позволяет управлять списком резюме и поддерживать актуальность данных.

- Поиск кандидатов: Приложение предоставляет возможность поиска кандидатов по различным критериям, таким как навыки, опыт работы, образование и другие параметры. Это помогает рекрутерам быстро и эффективно находить подходящих соискателей для вакансий.

- Обработка критических ситуаций: Приложение реализует обработку критических ситуаций, таких как ошибки при загрузке резюме, сбои в сети или другие непредвиденные события. В случае возникновения таких ситуаций система предоставляет соответствующие сообщения и предлагает решения или восстановление работы.

Для представления общей картины рынка таких сервисов был проведен анализ двух основных конкурентов: LinkedIn Talent Hub и Greenhouse.

LinkedIn Talent Hub (рисунок 1.1) является одним из ведущих конкурентов на рынке систем управления кадровыми ресурсами. Он предоставляет широкий спектр функциональности, включая загрузку и обработку резюме, поиск соискателей, взаимодействие с кандидатами и отслеживание процесса найма. Однако наше приложение предлагает более легковесный и простой в использовании интерфейс, а также более гибкую настройку функционала под конкретные потребности пользователей.












Global pipeline - All (163)			
Candidate	Project	Stage	
 Lois Silva Enterprise Account Manager	Account Manager in Chicago Updated 2hrs ago	Offer Offer extended	***
 Kyle Griffin Account Manager	Account Manager in Chicago Updated 2hrs ago	Interview Onsite Interview	***
 Laura Ball Enterprise Solutions Account Manager	Account Manager in Chicago Updated 2hrs ago	Screen Phone Screen 2	***
 Rhoda Haynes Marketing Specialist at Waterbnb	Senior Marketing Specialist in New York Updated 2hrs ago	Screen Phone Screen	***
 Ina Ball Account Manager	Account Manager in Chicago Updated 2hrs ago	Interview Onsite Interview	***
 Isabelle Barker Product Manager at Freshings	Product Managers in Seattle Updated 2hrs ago	Interview Product Challenge Test	***
 Maurice Martinez Product Manager at SpaceY	Product Managers in Seattle Updated 2hrs ago	Source Replied	***
 Josephine Medina Marketing Specialist at Waterbnb	Senior Marketing Specialist in New York Updated 2hrs ago	Source Contacted	***
 Frank Freeman Contract Marketing Agent	Senior Marketing Specialist in New York Updated 2hrs ago	Interview Onsite Interview	***
 Beulah Lawrence Associate Marketing Specialist	Senior Marketing Specialist in New York Updated 2hrs ago	Screen Phone Screen	***
 Emily Cobb Associate Account Manager	Account Manager in Chicago Updated 2hrs ago	Screen Phone Screen	***

Рисунок 1.1 – Сервис LinkedIn Talent Hub

Greenhouse (рисунок 1.2) является еще одним значимым конкурентом в сфере управления трудоустройством и набора персонала. Он также обладает функциями загрузки и обработки резюме, поиска соискателей, проведения интервью и управления процессом найма. Наше приложение отличается от *Greenhouse* более простым и интуитивно понятным пользовательским интерфейсом, а также предлагает более быструю и отзывчивую работу благодаря выбранному стеку технологий.

greenhouse Recruiting ▾ Jobs Candidates CRM Reports Integrations Add ▾ 🔍 ⚙️ ? Hi Greenhouse ▾				
Configure				
<ul style="list-style-type: none"> Organization Users Permission Policies Email Settings Notifications Email Templates Social Templates Offer Templates Order History Job Boards & Posts Custom Options Agencies Inclusion Tools Approvals Dev Center 	Greenhouse Agency Portal			
	Invite an agency to submit candidates directly into your database. Agencies will only see if a submitted candidate is "Active", "Rejected", or "Hired". They will NOT be able to view anything else about your candidates.			
	PLEASE NOTE: After you invite an agency, you still have to assign them to jobs. Agencies can only submit candidates to jobs which they have been assigned.			
	Add Agency Add Recruiter		<input type="checkbox"/> Include inactive agencies/recruiters <input type="checkbox"/> Include closed jobs	
	Agency / Recruiter	Assigned Jobs	Candidates	Active
	Best Talent Agency	2	2	2
	Elliot Holman	0	0	0
	Andre Koch	Invite sent	1	1
	Adam Gibson	0	0	0
	Arumi Lawson	1	1	1
	Hired			

Рисунок 1.2 – Сервис Greenhouse

Несмотря на то, что LinkedIn Talent Hub и Greenhouse являются уважаемыми и широко используемыми решениями на рынке, наше приложение имеет свои преимущества. В отличие от этих конкурентов, наше приложение является более легковесным и удобным в использовании благодаря применению библиотеки React и фреймворка Material-UI. Мы также выбрали библиотеку `ku` для выполнения HTTP-запросов на backend, что позволяет улучшить производительность и быстродействие системы. Кроме того, использование менеджера состояний Zustand обеспечивает простоту и эффективность управления состоянием приложения.

Наше приложение обладает такими преимуществами, как интуитивный интерфейс, быстрая обработка резюме, точный поиск кандидатов и обработка критических ситуаций. Оно предоставляет пользователям удобный и эффективный инструмент для обработки резюме и управления процессом найма.

Анализ конкурентов позволяет нам определить свои конкурентные преимущества и нацелиться на удовлетворение потребностей пользователей, которые ищут более простое, быстрое и гибкое решение для обработки резюме.

Стоит отметить, что серверная часть (backend) для этого сервиса разрабатывалась мной. Это позволяет нам полностью контролировать и адаптировать серверную часть под требования проекта, обеспечивая более гибкую интеграцию с фронтендом и оптимальную производительность системы. Backend был реализован с использованием Python, FastAPI, DynamoDB, Terraform, LocalStack, boto3, которые обладают широкими возможностями для разработки веб-приложений.

Благодаря доступу к серверной части проекта, мы имеем полный контроль над функциональностью и безопасностью серверной части, а также можем быстро реагировать на изменения требований проекта или возникающие проблемы.

Эффективное взаимодействие между frontend и backend частями нашего приложения обеспечивает гармоничную работу системы обработки резюме, улучшает пользовательский опыт и обеспечивает высокую производительность и надежность всей системы.

2 ВЫБОР И ОБОСНОВАНИЕ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

При проектировании интерфейса системы обработки резюме был осуществлен тщательный выбор стека технологий, учитывающий требования проекта и обеспечивающий эффективную разработку и функциональность системы. Ниже представлены основные технологии, которые были выбраны и обоснованы:

1. JavaScript (JS): JavaScript был выбран в качестве основного языка программирования для разработки клиентской части проекта. JavaScript – это язык программирования, который используют разработчики для создания интерактивных веб-страниц. Функции JavaScript могут улучшить удобство взаимодействия пользователя с веб-сайтом: от обновления ленты новостей в социальных сетях и до отображения анимации и интерактивных карт [1].

2. React: React был выбран в качестве основного фреймворка для разработки пользовательского интерфейса. React предоставляет компонентный подход, что упрощает создание и повторное использование компонентов интерфейса. Он также обладает быстрой виртуализацией DOM и хорошей производительностью, что важно для отзывчивого пользовательского опыта [2].

3. ky: Для выполнения HTTP-запросов к backend была выбрана библиотека ky вместо Axios. Решение принято в связи с тем, что ky имеет более компактный размер, удобный синтаксис и обладает более высокой производительностью, что позволяет улучшить время отклика системы [3].

4. Zustand: В качестве менеджера состояний был выбран Zustand. Zustand является современным и легковесным менеджером состояний для React-приложений. Он обладает простым API и хорошо интегрируется с React, что позволяет удобно управлять состоянием приложения и обеспечить его целостность.

5. Material-UI: Для создания пользовательского интерфейса была выбрана библиотека компонентов Material-UI. Material-UI предоставляет готовые компоненты, следующие принципам Material Design, что обеспечивает современный и стильный внешний вид интерфейса. Она также предлагает широкий выбор кастомизации и возможность быстрой разработки интерфейса.

6. React-Toastify: Для создания уведомлений в системе была выбрана библиотека React-Toastify. Она предоставляет простой и гибкий способ создания уведомлений с помощью React-компонентов. React-Toastify позволяет легко отображать уведомления различного типа, таких как успех, ошибки или информационные сообщения, и предоставляет настраиваемые параметры для управления их внешним видом и поведением.

Выбор данных технологий обоснован следующими причинами:

- JavaScript (JS) является широко распространенным языком программирования, обладающим большим сообществом разработчиков и

богатой экосистемой инструментов, что облегчает разработку и поддержку проекта.

- React предоставляет эффективный и модульный подход к разработке пользовательского интерфейса, позволяя легко создавать и управлять компонентами, а также обеспечивает высокую производительность благодаря виртуализации DOM.

- Библиотека `ku` выбрана вместо `Axios` из-за своей легковесности, удобного синтаксиса и повышенной производительности, что важно для быстрого выполнения HTTP-запросов к backend и улучшения отзывчивости системы.

- Zustand предлагает простое и интуитивно понятное API для управления состоянием приложения, что способствует чистому и эффективному коду, а также обеспечивает легкость интеграции с React [4].

- Material-UI обеспечивает стильный и современный внешний вид пользовательского интерфейса, а также предоставляет готовые компоненты, что ускоряет разработку и обеспечивает согласованный дизайн системы.

- React-Toastify обеспечивает удобный способ отображения уведомлений, что повышает пользовательский опыт и обеспечивает визуальную обратную связь для пользователей системы.

Выбранный стек технологий позволяет разработчикам эффективно реализовывать функциональность системы обработки резюме, обеспечивать высокую производительность и создавать удобный и современный интерфейс для пользователей.

3 ПРОЕКТИРОВАНИЕ САЙТА И ЕГО ИНТЕРФЕЙСА

При проектировании сайта и его интерфейса для нашего приложения по обработке резюме я уделял особое внимание удобству использования и визуальной привлекательности. Я стремился создать чистый и современный дизайн, используя Material-UI для разработки стильных компонентов интерфейса. Цветовая палитра была выбрана с учетом удобочитаемости и контрастности.

Мой главный приоритет был обеспечить удобство загрузки резюме в формате CSV и интуитивный поиск кандидатов по различным критериям. Был разработан простой и понятный интерфейс для управления резюме, включая возможность удаления и скачивание информации в CSV формате.

Макет сайта был разработан с акцентом на простоту использования и навигацию. Я создал удобное главное меню, которое обеспечивает быстрый доступ к основным функциям приложения.

Для удовлетворения различных устройств и экранов мы обеспечили реактивность и адаптивность интерфейса. Он динамически адаптируется к разным размерам экранов и обеспечивает оптимальное отображение на различных устройствах.

Я проконсультировался со специалистами в области frontend-разработки и UX/UI дизайна, чтобы улучшить интерфейс и общий пользовательский опыт. Результатом является интуитивно понятный и эстетически приятный интерфейс, который эффективно поддерживает пользователей при обработке резюме.

На главной странице сайта (рисунок 3.1) размещен текст-описание сервиса и все необходимые кнопки для быстрого доступа ко всем страницам сайта.

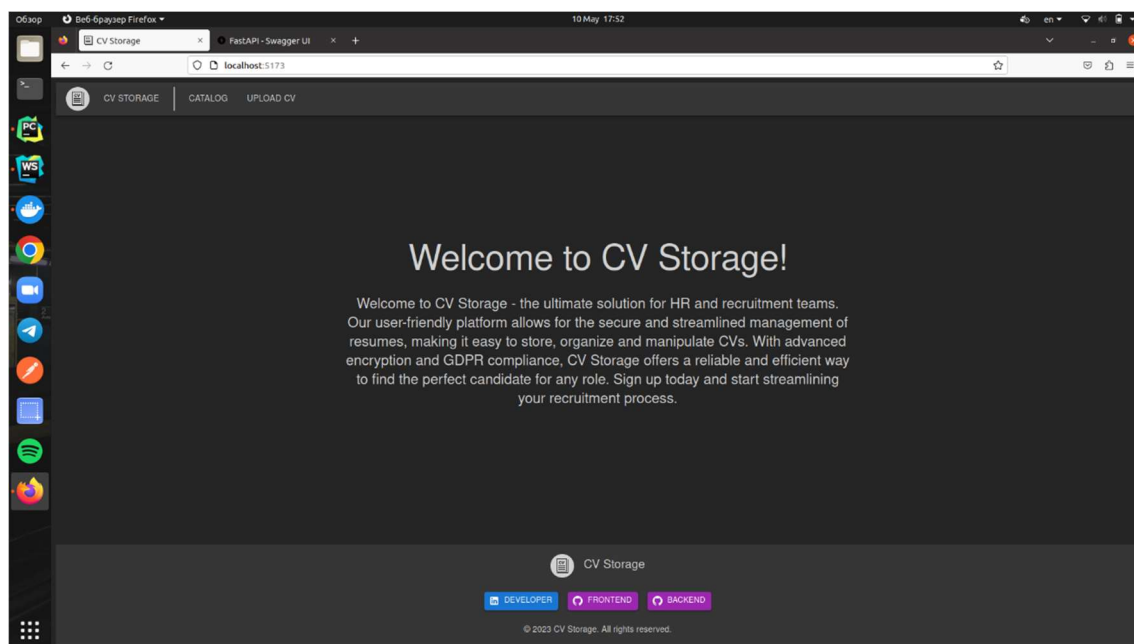


Рисунок 3.1 – Главная страница сайта «CV Storage»

Хедер сайта (рисунок 3.2) содержит логотип сайта, ссылку на главную страницу, ссылку на страницу «Каталог» и ссылку на страницу загрузки резюме.



Рисунок 3.2 – Хедер сайта «CV Storage»

Футер сайта (рисунок 3.3) содержит логотип сайта, ссылки на репозитории с исходным кодом клиентской и серверной части, ссылку на профиль разработчика в сети LinkedIn и авторские права разработчика.

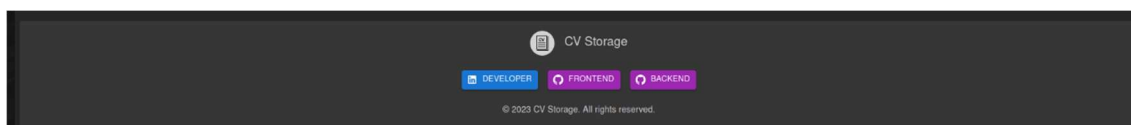


Рисунок 3.3 – Футер сайта «CV Storage»

Страница каталога (рисунок 3.4) представляет из себя список, на котором отображено 10 резюме. Возле каждого можно заметить кнопки быстрого функционала: открытие полного резюме, скачивание в формате .csv, удаление резюме из системы. Над списком находится строка поиска, благодаря которой можно выполнить поиск резюме по имени, фамилии или специальности кандидатов. Под списком находится меню пагинации для удобной навигации по страницам каталога.

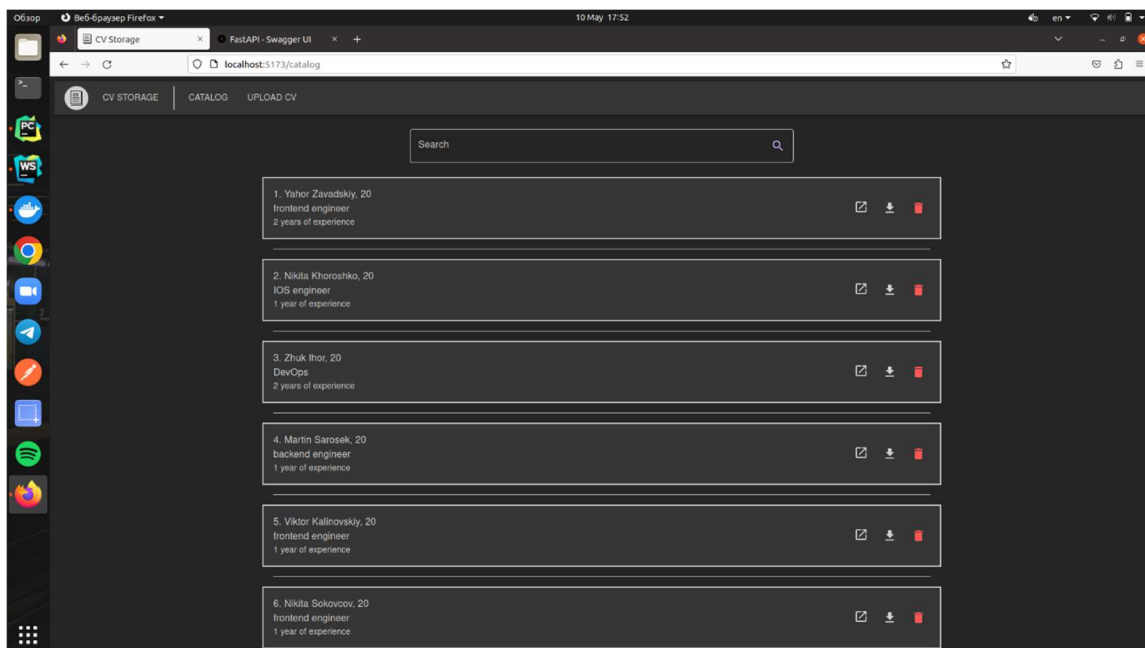


Рисунок 3.4 – Страница каталога на сайте «CV Storage»

Страница конкретного кандидата (рисунок 3.5) позволяет просматривать подробную информацию о кандидате в формате таблицы, скачивать резюме в формате CSV, а также удалять резюме из системы. Каждое поле в таблице, по нажатию, копирует значение в буфер обмена и сообщает об этом пользователю изменением состояния подсказки.

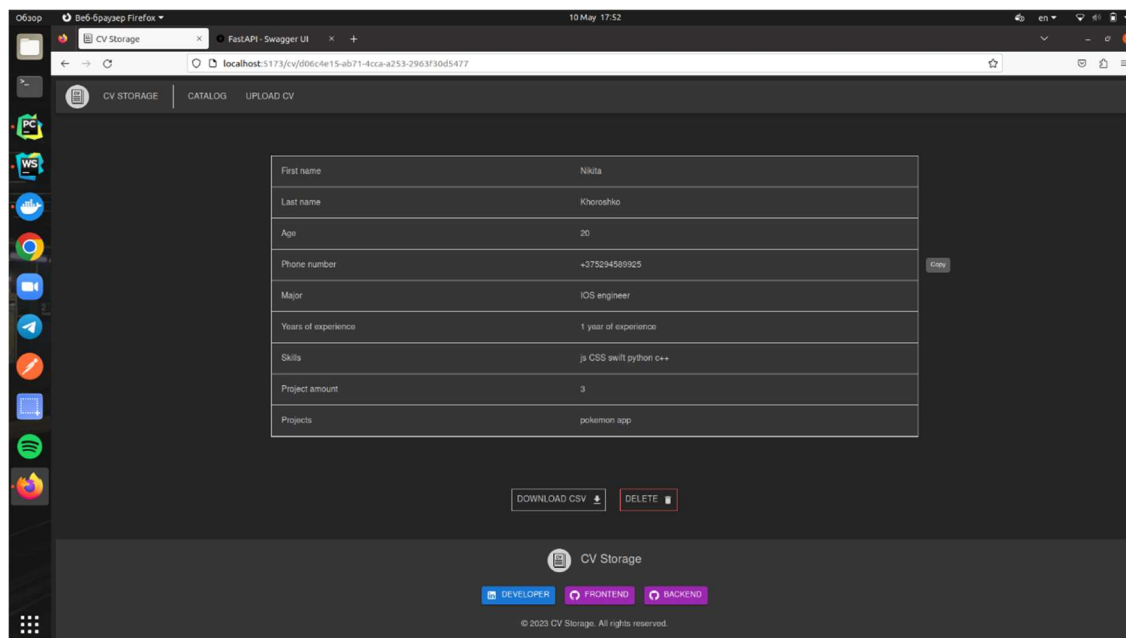


Рисунок 3.5 – Страница кандидата на сайте «CV Storage»

На странице загрузки резюме (рисунок 3.6) можно найти список полей, которые должны быть заполнены в .CSV-файле перед загрузкой, поле для загрузки файла, а также напоминание о том, что на данный момент система поддерживает исключительно CSV формат.

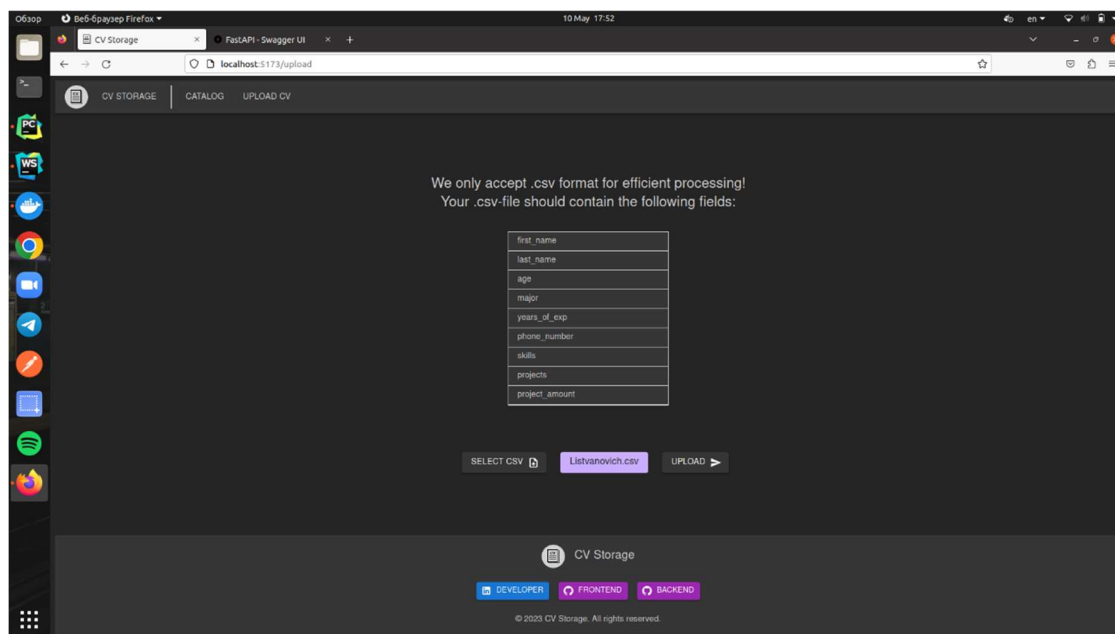


Рисунок 3.6 – Страница загрузки резюме на сайте «CV Storage»

ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта был успешно разработан и реализован интерфейс системы обработки резюме на основе языка JavaScript и фреймворка React. Проект включал в себя связь с backend-сервером, который был разработан с использованием языка программирования Python и фреймворка FastAPI. Это позволило создать полнофункциональную систему, обладающую удобным пользовательским интерфейсом и мощными возможностями обработки и анализа резюме.

Одной из ключевых функций разработанной системы является возможность загрузки резюме в формате CSV, что значительно упрощает процесс сбора и организации данных соискателей. Также была реализована функция скачивания резюме в формате CSV, обеспечивая гибкость и управление информацией в системе.

Другой важной характеристикой системы является возможность осуществлять поиск по необходимым критериям к кандидатам. Это позволяет рекрутерам быстро и эффективно отбирать соискателей, соответствующих требованиям вакансии.

В ходе разработки были предусмотрены и обработаны все критические ситуации, что обеспечивает надежность и стабильность работы системы. Были реализованы механизмы обработки ошибок, а также проверка целостности данных.

Этот проект позволил не только углубить понимание принципов проектирования интерфейсов и разработки программного обеспечения, но и применить полученные знания на практике. Результатом является готовая система обработки резюме, которая может быть успешно внедрена в реальные бизнес-процессы и принести значительные выгоды в области подбора персонала.

В заключение, разработка интерфейса системы обработки резюме является важным шагом в области управления кадровыми ресурсами. Полученные результаты демонстрируют эффективность применения технологий JavaScript, React, Python и FastAPI при создании интуитивно понятного и мощного инструмента для обработки резюме. Проект успешно достиг поставленных целей и открывает перспективы для дальнейшего улучшения и расширения функциональности системы.

Дальнейшее развитие проекта может включать следующие направления:

1. Улучшение пользовательского интерфейса: на основе обратной связи пользователей и анализа их потребностей, можно провести дальнейшую оптимизацию интерфейса, сделать его ещё более интуитивно понятным и удобным для работы.

2. Расширение функциональности: систему можно дополнить новыми возможностями, такими как автоматическое сопоставление резюме с требованиями вакансий, анализ навыков и опыта соискателей, интеграцию с внешними сервисами для проверки достоверности данных и многое другое.

3. Расширение поддерживаемых форматов резюме: помимо формата CSV, можно добавить поддержку других популярных форматов, например, PDF, DOCX и TXT, чтобы обеспечить более широкую совместимость с различными источниками резюме.

4. Улучшение производительности и масштабируемости: при работе с большим объемом данных необходимо оптимизировать производительность системы и обеспечить ее масштабируемость, чтобы она могла эффективно обрабатывать резюме даже при высокой нагрузке.

5. Развитие мобильной версии: в современном мобильном мире важно предоставить возможность удобного доступа к системе с мобильных устройств. Разработка мобильной версии интерфейса позволит пользователям использовать систему на любом устройстве.

В целом, разработка интерфейса системы обработки резюме представляет собой важный этап в улучшении процесса подбора персонала. Проект, основанный на JS&React, Python&FastAPI, успешно реализовал ключевые функции системы, обеспечивая удобство, надежность и эффективность работы с резюме. Благодаря полученным результатам, система готова к дальнейшему применению в реальных условиях и может быть адаптирована под конкретные потребности предприятий и организаций.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Amazon Web Services [Электронный ресурс]. – Режим доступа: <https://aws.amazon.com/ru/what-is/javascript/>
2. Алекс Бэнкс, Ева Порселло. React: Современные шаблоны для разработки приложений / Алекс Бэнкс, Ева Порселло. – Sebastopol: O'Reilly, 2023. – 320 с.
3. Github – ky [Электронный ресурс]. – Режим доступа: <https://github.com/sindresorhus/ky#readme>
4. Habr – React: Zustand State Manager [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/companies/timeweb/articles/646339/>

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг кода

Файл Catalog.jsx

```
import React, { useEffect } from "react";
import { Typography } from "@mui/material";

import { useStore } from "../store";
import { CVStack } from "../components/CV_Components";
import { Loader } from "../components/Loader";

export const Catalog = () => {
  const { cvs, searched_cvs, loading } = useStore();

  useEffect(() => {
    useStore
      .getState()
      .fetchCVs()
      .then((amount) => console.log(`Fetched ${amount} CVs`));
  }, []);

  if (loading) {
    return <Loader />;
  }

  return (
    <>
      {cvs.length ? (
        <CVStack cvs={searched_cvs} sx={{ flex: "10 1" }} />
      ) : (
        <Typography
          variant="h4"
          sx={{
            py: 3,
            flex: "10 1",
            justifyContent: "center",
            textAlign: "center",
          }}
        >
          CV Storage is now empty!
        </Typography>
      )}
    </>
  );
};
```

Файл CVPage.jsx

```
import React, { useEffect, useState } from "react";

import { useParams } from "react-router-dom";
import { Stack } from "@mui/material";

import { get_single_cv } from "../API";
import {
  DeleteTextButton,
  DownloadCSVTextButton,
} from "../components/UI/Buttons";
import { CVTable } from "../components/CV_Components";
```

```

import { Loader } from "../components/Loader";
import { useStore } from "../store";

export const CVPage = () => {
  const { cv_id } = useParams();
  const [person, setPerson] = useState({});
  const { loading, setLoading } = useStore();

  useEffect(() => {
    const loadData = async () => {
      setLoading(true);

      // API-on version
      const response = await get_single_cv(cv_id);
      setPerson(response.data);

      // API-off version
      // setPerson(single_cv_example);

      setLoading(false);
      return cv_id;
    };
    loadData().then((id) => console.log(`Fetched page for ID = ${id}`));
  }, []);

  if (loading) {
    return <Loader />;
  }

  return (
    <>
      <CVTable cv={person} />
      <CV_Links cv_id={cv_id} person={person} />
    </>
  );
};

const CV_Links = ({ cv_id, person }) => {
  const filename = `CV_${person.last_name}`;
  return (
    <Stack
      direction={"row"}
      spacing={3}
      sx={{ flex: "2", alignItems: "center" }}
    >
      <DownloadCSVTextButton cv_id={cv_id} filename={filename} />
      <DeleteTextButton cv_id={cv_id} />
    </Stack>
  );
};

```

Файл UploadPage.jsx

```

import React from "react";
import { Box, Typography } from "@mui/material";

import { TableFields, UploadBar } from "../components/UploadBar";

export const UploadPage = () => {
  return (
    <Box
      sx={{
        display: "flex",

```

```

        flexDirection: "column",
        alignItems: "center",
        justifyContent: "space-between",
      }}
    >
    <Typography variant="h5" align="center" color="#d2d2d2" paragraph>
      We only accept .csv format for efficient processing!
      <br /> Your .csv-file should contain the following fields:
    </Typography>
    <TableFields />
    <UploadBar />
  </Box>
);
};

```

Файл AppRouter.jsx

```

import React from "react";
import { Box } from "@mui/material";
import { Navigate, Route, Routes } from "react-router-dom";
import { ToastContainer } from "react-toastify";

import { routes } from "../router";
import { Footer } from "../components/Footer";
import { Header } from "../components/Header";

export const AppRouter = () => {
  return (
    <>
      <Box
        sx={{
          display: "flex",
          flexDirection: "column",
          minHeight: "98.3vh",
          justifyContent: "space-between",
        }}
      >
        <Header sx={{ flex: "2" }} />
        <Routes>
          {routes.map((route) => (
            <Route
              path={route.path}
              element={route.component} />
            </Route>
          ))}
          <Route path="*" element={<Navigate to="/" replace />} />
        </Routes>
        <Footer sx={{ flex: "2" }} />
      </Box>

      <ToastContainer pauseOnFocusLoss pauseOnHover />
    </>
  );
};

```

Файл store.js

```

import { create } from "zustand";
import { cvs, get_all_cvs } from "../API";

export const useStore = create((set, get) => ({

```



```

    cvs: [],
    searched_cvs: [],
    loading: false,
    page: 1,
    cv_per_page: 10,
    number_of_pages: 1,

    async fetchCVs() {
      set({ loading: true });

      ///// API-on version
      const all_cvs_response = await get_all_cvs();
      set({ cvs: all_cvs_response });
      set({ searched_cvs: all_cvs_response });

      ///// API-off version
      // set({ cvs });
      // set({ searched_cvs: cvs });

      const page_amount = Math.ceil(get().cvs.length / get().cv_per_page);
      set({ number_of_pages: page_amount });

      // Case: user deletes last cv on page
      // Behavior: we're moving user to the previous page
      if (get().page > get().number_of_pages) {
        set({ page: get().page - 1 });
      }

      set({ loading: false });
      return get().cvs.length;
    },

    async searchCVs(str) {
      set({ loading: true });

      str = str.toLowerCase();
      const searched_cvs = get().cvs.filter(
        (cv) =>
          (
            cv.first_name.toLowerCase() +
            " " +
            cv.last_name.toLowerCase()
          ).includes(str) || cv.major.toLowerCase().includes(str)
      );
      const page_amount = Math.ceil(searched_cvs.length /
get().cv_per_page);
      set({ searched_cvs: searched_cvs });
      if (str !== "") {
        set({ page: 1 });
      }
      set({ number_of_pages: page_amount });

      set({ loading: false });
    },

    setPage(newPage) {
      set({ page: newPage });
    },

    setLoading(flag) {
      set({ loading: flag });
    },
  });
});

```

Файл Header.jsx

```
import React from "react";
import { Stack, AppBar, Divider } from "@mui/material";

import { Logo } from "../Header";
import { HeaderButton } from "../UI/Buttons";

export const Header = () => {
  return (
    <AppBar position="relative">
      <Stack
        direction={"row"}
        spacing={2}
        justifyContent={"left"}
        sx={{ py: 1, bgcolor: "#363636", alignItems: "center", width: 1
      }}
    >
      <Divider orientation="vertical" variant="middle" flexItem />
      <Logo />
      <HeaderButton text={"CV Storage"} link="/" />
      <Divider
        orientation="vertical"
        variant="middle"
        flexItem
        sx={{ bgcolor: "#d2d2d2", borderRightWidth: 2 }}
      />
      <HeaderButton text={"Catalog"} link="/catalog" />
      <HeaderButton text={"Upload CV"} link="/upload" />
    </Stack>
  </AppBar>
);
};
```

Файл Footer.jsx

```
import React from "react";
import { Box, Container, Stack, Typography } from "@mui/material";

import { Copyright, FooterLinks } from "../Footer";
import { Logo } from "../Header";

export const Footer = () => {
  return (
    <Box component="footer" sx={{ bgcolor: "#363636" }}>
      <Container maxWidth="lg" sx={{ py: 2 }}>
        <Stack direction={"row"} spacing={2} sx={{ alignItems: "center"
      }}>
          <Logo />
          <Typography variant="h6" align="center" gutterBottom>
            CV Storage
          </Typography>
        </Stack>
        <FooterLinks />
        <Copyright />
      </Container>
    </Box>
  );
};
```