Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ярославский государственный университет им. П.Г. Демидова»

Кафедра информационных и сетевых технологий

Сдано на кафедру
«_____»_____ 2020 г.
Заведующий кафедрой,
к. ф.-м. н.

_____ Д. Ю. Чалый

Выпускная квалификационная работа

Разработка клиентской части системы для автоматизации процесса рекрутирования сотрудников

по направлению
09.03.03 Прикладная информатика

Научный руководитель
стар. преподаватель

_____ Н. В. Легков
«_____»_____ 2020 г.

Студент группы ПИЭ-41БО

_____ О. С. Гаршина
«_____»_____ 2020 г.

Ярославль, 2020

Реферат

Объем 13 с., 4 гл., 1 рис., 0 табл., 1 источников, 0 прил.

Ключевые слова и выражения: **react, рекрутер, автоматизация, HR, резюме, front-end, JavaScript**

Целью данной работы является разработка клиентской части приложения - HR-CV Portal, который оптимизирует работу рекрутеров при создании резюме.

В работе проведён анализ потребностей клиента. Также сформированы требования к приложению, определены технологии для разработки, отвечающие поставленным требованиям. В результате работы был получен опыт в сфере front-end разработки, а конечный продукт передан клиенту для использования и получил положительные отзывы и обратную связь для наращивания функционала в дальнейшем.

Содержание

Введение

Несмотря на то, что мы живем в век технологий и автоматизации есть еще много аспектов, требующих алгоритмов, которые не сможет имитировать машина. Такие вещи обычно требуют психологических навыков, индивидуального подхода и нажитого опыта.

В дипломной работе рассматриваются проблемы рекрутеров компании Akvelon при бюрократический деятельности, а именно проблемы при работе с огромным колчеством резюме, которые надо создавать с нуля, редактировать и поддерживать в актуальном состоянии.

Заполнение резюме формата комапнии Akvelon может занимать у сотрудника от часа до четырех часов. Как показал опрос клиента, наибольшей проблемой является время, потраченное на копирование информации из одного места в другое, орфографические ошибки кандидатов, правки съехавшей разметки в word-документе.

В связи с этим, было поставлена задача создать web-приложение, которое бы являлось централизованным хранилищем всех резюме компании и цель которого — сделать процесс заполнения данного документа менее рутинным и медленным.

1. Теория

2. О задаче

2.1. Постановка задачи

Требуется создать web-приложение, которое упрощает создание и обновление резюме кандидатами и работниками компании Akvelon, а также решает многие проблемы рекрутеров, оптимизируя их работу и тем самым сокращая время, проведенное над редактированием документов.

2.2. Требуемый функционал

1. Возможность создания, копирования, редактирования и архивирования резюме;
2. Наличие базы данных, в которой бы хранились названия компаний, институтов; проектов, навыков, персональных результатов и сфер ответственности;
3. Автоматическое заполнение перечисленных данных в поля резюме - всплывающие подсказки и поиск по ним;
4. Возможность пополнения этой базы данных как обычными пользователями так и администраторами сайта;
5. Модерация добавленных данных администраторами в один клик;
6. Подобие папок с проектами, на которые можно назначить кандидатов и производить поиск по имени и позиции;
7. Скачивание резюме в формате .docx, стилизованное под стандартное резюме Аквелона;
8. Заполнение общей информации о кандидате с помощью подсказок с логическими выделенными словосочетаниями; которые превращаются в подобие шаблона при их выборе. Подсказки должны предлагаться в случайном порядке, чтобы повысить уникальность текста в резюме;
9. Пользователь приложения должен иметь возможность указать свою роль на проекте, для которого создается резюме; Смена этой роли должна вызвать автоматическую пересортировку данных, чтобы наиболее актуальные для позиции умения находились выше остальных;
10. Сайт нужно создать в стиле Аквелона, придерживаясь дизайна других сервисов данной компании;
11. Возможность дать другим пользователям права модератора;
12. Блокировка и удаление пользователя;
13. Всплывающие уведомления об ошибках и прочей информации для пользователя;

14. Интерфейс для отслеживания прогресса работы приложения;

2.3. Используемые программные средства

Исходя из того, что требуется написать клиентскую часть приложения, для разработки были выбраны следующие программные средства:

- VSCode для разработки и отладки приложения;
- JavaScript в качестве основного языка программирования;
- GitLab для управления репозиторием кода для Git;
- MobX для управления состоянием приложения;
- Axios для взаимодействия с API;
- React.js для создания интерфейса;
- Material UI для создания единого стиля компонентов;
- Less для корректировок стиля Matreial и для создания собственного.
- Jest и Enzyme для написания unit-тестов.

3. Решение задачи

3.1. Создание базовой архитектуры

В компании мне предоставили готовый шаблон со структурой, где уже подключен Webpack, настроено несколько правил ESLint для поддержания кода чистым и более приятным глазу. Для начала разработки была релизована следующая структура проекта в директории src:

```
/
├── components
├── containers
├── services
│   ├── action.notify
│   ├── toast.notify
│   ├── stores
│   ├── request.services
│   └── validator
├── app.js
├── app.less
├── constants.js
├── index.html
├── muitheme.js
├── router.js
└── variables.less
```

- Папка components предназначена для react-компонентов для многоразового использования, непривязанных к какому-либо контексту, желательно максимально абстрактных.
- Containers - каталог для логически разделенных папок, содержащих в себе компоненты конкретных страниц.
- Services - папка для сервисов, которые отвечают за реализацию кода, независимого от внешнего окружения. В данном приложении понадобились сервисы для логики полос прогрузки данных, появления уведомлений, взаимодействия с API, валидации, и действий с observable-состаяниями MobX-а.
- index.html - точка входа приложения, в нем описываются подключения стилей и скрипта для рендера.
- index.js указывает, в какую область html документа будет проецироваться DOM-дерево и рендерит app.js.
- app.js содержит компоненты-провайдеры, отвечающие за авторизацию, инициализацию MobX stores, стилей-muitheme и перенаправления на страницы.
- app.less - в этой файле написаны общие стили, которые используются прак-

тически во всех компонентах.

- constants.js - переменные, которые используются в разных местах программы по типу предложений, коэффициентов, регулярных выражений.
- muitheme.js - файл, позволяющий задать конфигурации Material UI стилей.
- router.js - компонент-маршрутизатор, определяет какой обработчик надо вызвать для конкретного маршрута.
- variables.less - содержит палитру именных основных цветов сайта. Файл служит для удобства, чтобы было проще ориентироваться на название переменной, а не на HEX или RGB коды.

3.2. Создание сервиса запросов, авторизация

Первым делом предстояло как-то идентифицировать пользователей, для чего был написан компонент-обертка над app.js. В классе AuthorizaionProvider в конструкторе задается состояние isAuthorized, равное false. При каждом первоначальном рендеринге (монтровании) этого компонента вызовется функция, отвечающая за авторизацию. Для этого появился первый сервис в папке services - request.service со следующей структурой:

```
request.service
├── api
│   └── auth.js
└── index.js
```

В auth.js

3.3. Регистрация и вход на сервис

Клиент поставил условие о том, что страницы, связанные с авторизацией должны быть выполнены в таком же стиле, что и сайт компании для оценки рабочего времени, написанный на Vue.js. Но должна быть возможность входа с любой почтой, а не с доменным именем.

Перед те

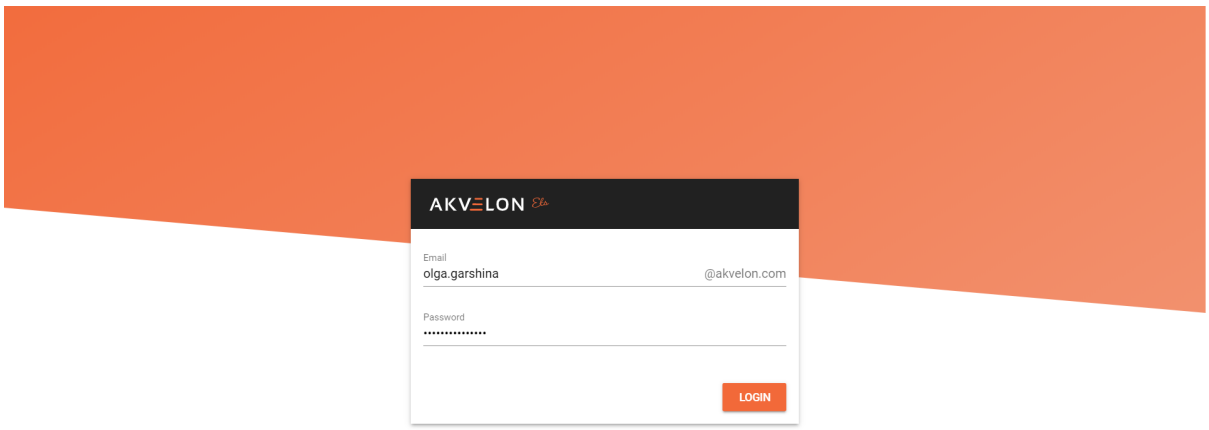

Рис. 1 — Страница входа на ets.akvelon.com

3.4. Компиляция умных контрактов

3.5. Развертывание умных контрактов в локальной сети

3.6. Развертывание умных контрактов в удаленных сетях

3.7. Отладка умных контрактов

3.8. Визуальный интерфейс для взаимодействия с развернутыми умными контрактами

4. Результаты решения задачи

В результате решения задачи было получено расширение для VSCode

Заключение

Список литературы

[1] Visual Studio Code - Code Editing. Redefined URL: https://code.visualstudio.com (дата доступа: 09.06.2020)