МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Политехнический институт: Заочный
Кафедра «Системы автоматического управления»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
д.т.н., профессор
_____/ В.И. Ширяев
«____» _____ 2019 г.

Система по автоматизации и оптимизации бизнес-процессов "Робот-Рекрутер" по найму и переводу сотрудников на ОАО "ММК"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ – 09.03.01.2019.307.00 ПЗ ВКР

Руководитель работы
доцент каф. САУ, к.т.н.
_____/ В.О. Чернецкий
«____» _____ 2019 г.

Автор работы
студент группы ПЗ-597
_____/ А.В. Каменев
«____» _____ 2019 г.

Нормоконтролер
доцент каф. САУ, к.т.н.
_____/ В.О. Чернецкий
«____» _____ 2019 г.

Челябинск 2019

АННОТАЦИЯ

Каменев А.В. Система по автоматизации и оптимизации бизнес-процессов "Робот-Рекрутер" по найму и переводу сотрудников на ОАО "ММК". – Челябинск: ЮУрГУ, ПИ: Заочный; 2019, 54 с., 20 ил., библиогр. список – 13 наим., 15 листов слайдов презентации ф.А4.

Разработана система по автоматизации и оптимизации бизнес-процессов по найму и переводу сотрудников. Система позволяет расширить воронку подбора персонала и привлечь более квалифицированные и мотивированные кадры. Спроектирована база данных системы, реализован сайт с информацией о вакансиях, сайт и мобильное приложение для проведения видеоинтервью, система распознавания документов государственного образца, брокер сообщения для безопасного функционирования системы, информационный киоск для заполнения резюме и получения информации о вакансиях.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Как известно, кадры решают все. Сегодня, в условиях непрерывного усовершенствования и вместе с тем усложнения производственного оборудования, трудно переоценить важность квалифицированных специалистов. Правильно обученные кадры, соответствующие актуальным и перспективным требованиям производства, всегда были и будут залогом стабильной и эффективной работы предприятия.

В крупных компаниях существует острая потребность в найме большого количества персонала. На выполнение рутинных задач: поиск, отбор и первичный контакт с соискателями тратится много времени. Одним из путей решения данной проблемы является внедрение в бизнес-процессы по поиску и найму сотрудников роботов-рекрутеров, позволяющих автоматизировать рутинные задачи. Группа компаний ПАО «ММК» не является исключением в этом вопросе. Поэтому руководством компании было принято решение о разработке и внедрении автоматизированной системы «Робот-Рекрутер», позволяющей на новом качественном уровне вести отбор персонала, отвечающего всем современным требованиям.

Целями проекта комплексной автоматизированной системы «Робот-Рекрутер», являются:

- Цифровизация процесса отбора персонала
- Транспарентность отбора персонала
- Открытый и равный доступ к информации о вакансиях
- Привлечение наиболее квалифицированного и мотивированного персонала
- Оперативный подбор персонала
- Формирование и развитие HR-бренда ПАО «ММК».

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Обзор существующих аналогичных решений

Эффективный найм персонала – это когда сотрудники, приходящие в компанию, имеют высокую мотивацию и квалификацию. Эту задачу невозможно решить без должного уровня автоматизации кадровой службы, автоматизация бизнес-процессов найма сотрудников снижает нагрузку на специалистов управления кадров, делает процесс найма транспарентным и позволяет отобрать для очного собеседования кандидатов с максимально подходящим уровнем компетенций. Специалистам кадровой службы больше нет необходимости перебирать и отсеивать изначально неподходящих резюме, в следствии чего они больше времени уделяют подбору компетентных работников. Это особенно важно при поиске высококвалифицированных сотрудников.

На текущий момент на российском рынке широко представлены следующие компании, занимающиеся в сфере автоматизации бизнес-процессов, связанных с наймом сотрудников:

- Робот Вера;
- AgileBot.

1 Робот Вера.

Наиболее известное решение на российском рынке. Робот Вера осуществляет поиск резюме, по требованиям указанным в вакансии, обзвон выбранных кандидатов, задает вопросы, анализирует ответы и отвечает на вопросы соискателей, проведение видеоинтервью, распознает эмоции кандидата во время видео собеседования и присылает видео.

2 AgileBot

Бот формирует первичных список кандидатов по заданным параметрам, приглашает кандидатов пройти онлайн-собеседование, автоматически проверяет компетентность каждого кандидата, предлагает лучшие резюме менеджеру.

Каждое из этих решений имеет как преимущества, так и недостатки, но все их объединяет одно, все эти решения находятся на уровне стартапов и все еще очень «сырые». Онлайн-собеседование в представлении AgileBot это простой чат-бот, в общении с которым пользователю не предоставляется свободы действия. На сайте робота Веры, есть возможность заполнить форму, чтобы Вера перезвонила, и есть возможность поговорить с роботом онлайн. Ни одни из этих механизмов у меня так и не заработал. Робот мне не перезвонил, поговорить на сайте так же не получилось. Ознакомившись с отзывами, мы пришли к выводу, что качество

распознавания голоса при звонке робота Веры очень низкое, часты ошибки и непонимание собеседника.

Помимо всего прочего, у ПАО «ММК», есть специфичные бизнес-процессы, при найме сотрудников. Ни один из сервисов не предоставляет такого функционала, так же они не могу предоставить полную интеграцию с корпоративной информационной системой (КИС), это невозможно из-за соображения безопасности хранящихся в ней данных. Это все послужило толчком к разработке собственной системы, полностью отвечающей требованиям управления кадров группы компаний ПАО «ММК».

1.2 Обзор сетевых архитектур

Для разработки системы необходимо выбрать сетевую архитектуру. В зависимости от выбранной архитектуры необходимо обозначить элементы сети, нужное программное обеспечение и методы кодирования. Наиболее распространенными являются следующие типы архитектуры:

- одноранговая архитектура;
- архитектура терминал-главный компьютер;
- архитектура клиент-сервер.

1 Одноранговая архитектура

Одноранговые сети – это малые сети, в которых любая выбранная вычислительная машина может выполнять функции и файлового сервера, и рабочей станции. В одноранговых сетях файловое хранилище любой рабочей станции может быть общим, для этого необходимо использовать службы удаленного доступа. Для файлов также можно установить защиту данных. [11] На рисунке 1.1 представлена обобщенная схема одноранговой сети.

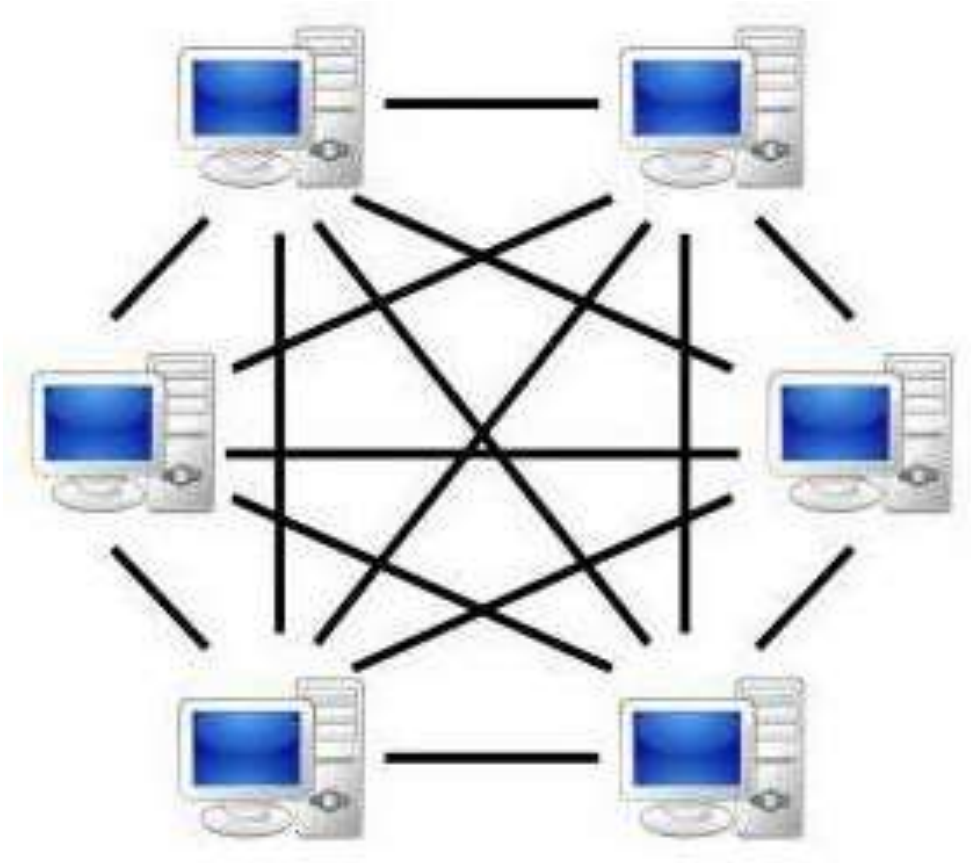

Рисунок 1.1 – Схема одноранговой сети

Преимущества одноранговых сетей:

- простая установка и настройка сети;
- низкая стоимость;
- небольшое количество оборудования;
- нет необходимости в обучении системного администратора;
- нет зависимости работоспособности вычислительных машин от выделенного сервера;

Недостатки одноранговых сетей:

- поддержка малого количества пользователей;
- сетевая безопасность реализуется на каждой вычислительной машине отдельно;
- отсутствие централизованного администрирования;
- снижение производительности при одновременном обращении к разделяемому ресурсу.

2 Архитектура «Терминал - главный компьютер»

Концепция архитектуры заключается в обработке данных одним компьютером или группой главных компьютеров. [11]

Для построения сети необходимо использовать два типа оборудования:

- главный компьютер, на котором будут осуществляться операции с данными, а также управление сетью;
- терминалы, на которых будет осуществляться передача команд на выполнение операций с указанными данными, создание сеансов и получения результатов;

Связь главного компьютера и терминалов производится через модуль передачи данных. На рисунке 1.2 представлена обобщенная схема архитектуры «Терминал – главный компьютер».

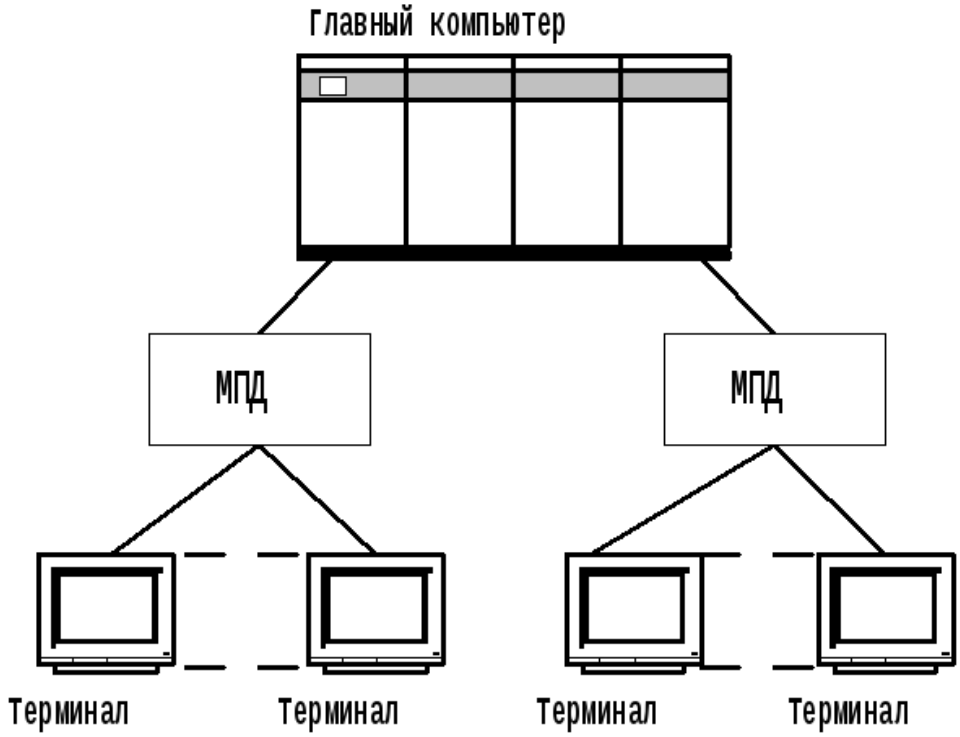

Рисунок 1.2 – Схема одноранговой сети

3 Архитектура «Клиент-сервер»

Для разработки была выбрана модель архитектуры «Клиент-сервер». Особенностью данного типа архитектуры является разделение нагрузки между поставщиками услуг (серверами) и заказчиками услуг (клиентами). Клиенты и серверы – это программное обеспечение, расположенное на одной вычислительной машине или же распределенное по разным машинам. [11]

Сервер предназначен для сложных операций ввода, хранения, обработки и модификации данных, а также для обращения к базе данных. Результатом

действий сервера является сообщение сети об успешном проведении операции или об ошибке. Серверы могу обрабатывать несколько обращений от клиентов к одному и тому же файлу одновременно. Эта особенность позволяет ускорить работу используемых приложений. Для выполнения запросов от нескольких клиентов сервер обычно размещается на специально выделенной вычислительной машине, чья производительность должна соответствовать предполагаемой нагрузке.

Процесс, отправляющий запрос, называется клиентом. Клиентом может быть, как программа, так и пользователь. Чаще всего это отдельные рабочие станции, использующие ресурсы сервера, отправляющие запросы на взаимодействие, а также предоставляющие удобные интерфейсы пользователя для процедуры взаимодействия с системой или сетью. На рисунке 1.3 представлена обобщенная схема архитектуры «Терминал – главный компьютер».

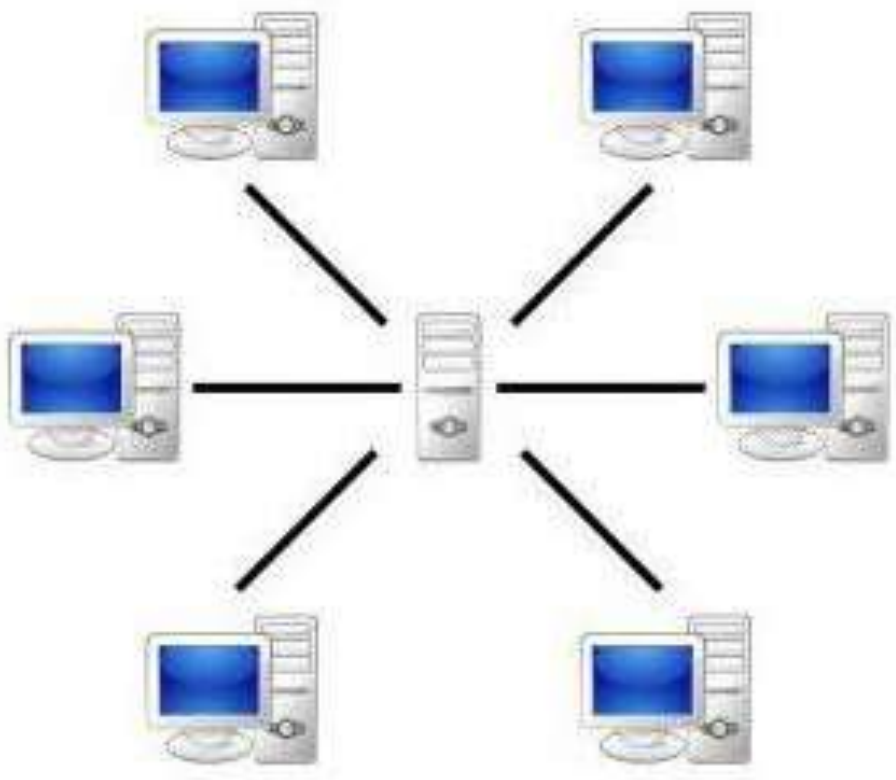

Рисунок 1.3 – Схема клиент-серверной архитектуры

Преимущества клиент-серверной архитектуры

- снижение требований к компьютерам - клиентам в том случае, если все вычисления выполняются на отдельном сервере;
- контроль полномочий на стороне сервера и настройка защиты для допуска к нужным данным только клиентов, чьи права соответствуют уровню доступа;
- отсутствие дублирования кода сервера клиентами.

Недостатки клиент-серверной архитектуры

- достаточно высокая стоимость производительного сервера;
- для поддержки работы системы чаще всего необходимо обучение отдельного специалиста - системного администратора;
- нарушение работоспособности сервера, а также низкая производительность сервера влияют на всю вычислительную сеть.

1.3 Обзор архитектур СУБД

Помимо клиентской и серверной части система «Робот-рекрутер» также должна включать в себя базу данных.

База данных - структурированная определенным образом совокупность данных, хранящихся и обрабатывающихся в соответствии с некоторыми правилами. Существует несколько типов баз данных: [3]

- иерархические. Иерархическая база представляет собой схему, имеющую объекты различных уровней (дерево). Создаются родительские и дочерние элементы, реализованы принципы наследования и группировки по типу сохраненной информации.
- сетевые. Сетевая база данных также строится по принципу иерархии, однако данный тип отличается наличием большего количества связей: один дочерний элемент может быть связан с несколькими родительскими. Недостатком сетевых баз данных является сложная система связей при хранении больших массивов, что снижает эффективность использования базы.
- реляционные. Реляционные базы данных являются одним из самых распространенных типов баз. Реляционная база данных представляет собой отдельную таблицу, доступ к которой выполняется посредством обращения к строке, столбцу или ячейке. Обращение может происходить напрямую, а также с помощью системы управления базой данных и языка запросов.

Таблица содержит в себе тип данных, порядковый номер, параметр, текст и т.п.

Для разработки системы «Робот-рекрутер» использована реляционная база данных, поскольку данный тип является самым эффективным и простым в области обработки поисковых запросов.

1.4 Обзор шаблонов проектирования архитектуры приложения

Для того чтобы снизить трудовые и временные, а равно и финансовые затраты на разработку сложного программного обеспечения, необходимо придерживаться общепринятых подходов к разработке программных продуктов. Одним из таких подходов являются шаблоны проектирования. Благодаря шаблонам проектирования снижается риск допущения ошибок, облегчается коммуникация между программистами, позволяя ссылаться на общепринятые конструкции.

Существует несколько шаблонов проектирования архитектуры приложения. Из них наиболее известными являются: [2]

- MVC
- MVVM

Для построения общей архитектуры приложения целесообразнее использовать архитектуру MVC (Model-View-Controller) – Данная схема позволяет разделить данные приложения, интерфейса пользователя и бизнес логики на три отдельных компонента: модель, представление и контроллер — такая схема позволяет осуществлять модификацию каждого элемента в отдельности, не затрагивая остальные. Модульная структура делает систему масштабируемой и распределенной.

Модель (Model) в зависимости от команды контроллера предоставляет или изменяет данные.

Представление (View) представляет из себя пользовательский интерфейс и отвечает за отображение данных пользователю.

Контроллер (Controller) содержит в себе бизнес логику и логику подготовки и отображения модели данных.

Использование концепции MVC дает ряд преимуществ, весомым из которых является — разделение пользовательских интерфейсов от модели данных и бизнес логики управления этими данными.

Поддержка различных типов пользователей, которые используют различные типы устройств является общей проблемой наших дней. Предоставляемый интерфейс должен различаться, если запрос приходит с персонального

компьютера или с мобильного телефона. Модель возвращает одинаковые данные, единственное различие заключается в том, что контроллер выбирает различные виды для вывода данных.

Концепция MVC, в значительной степени, уменьшает сложность больших приложений, позволяя сделать код более структурированным, что позволяет облегчить поддержку, тестирование и повторное использование решений.

MVVM (Model-View-ViewModel) шаблон наиболее применим для реализации пользовательских интерфейсов, когда необходимо выполнить двухстороннюю привязку данных. [5]

Шаблон MVVM делится на три части:

Модель (так же, как в классической MVC) представляет из себя фундаментальные данные, необходимые для работы приложения.

Представление — интерфейс пользователя (web интерфейс). Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых моделью представления. В случае, если в модели представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и представление, в свою очередь, запрашивает обновлённое значение свойства из модели представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, представление вызывает соответствующую команду, предоставленную моделью представления.

Модель представления — с одной стороны, абстракция представления, а с другой — обёртка данных из модели, подлежащие связыванию. То есть, она содержит модель, преобразованную к представлению, а также команды, которыми может пользоваться представление, чтобы влиять на модель.

1.5 Обзор средств разработки

Для того чтобы повысить эффективность разработки программных продуктов принято использовать интегрированную среду разработки (IDE).

Среда разработки включает в себя:

- текстовый редактор,
- компилятор и/или интерпретатор,
- средства автоматизации сборки,
- отладчик.

Использование IDE для разработки программного обеспечения является прямой противоположностью способу, в котором используются несвязанные инструменты, такие как текстовый редактор, компилятор, и т.п. Интегрированные

среды разработки были созданы для того, чтобы максимизировать производительность программиста благодаря тесно связанным компонентам с простыми пользовательскими интерфейсами. Это позволяет разработчику сделать меньше действий для переключения различных режимов, в отличие от дискретных программ разработки. [1]

Существует большое количество IDE от разных производителей и для разных целей. Все из них имеют свои преимущества и недостатки, некоторые из них распространяются под коммерческой лицензией, другие распространяются свободно, третьи имеют условно бесплатную лицензию (с ограничениями).

Наиболее широкую известность получили интегрированные среды разработки от компании Майкрософт.

Для разработки на стеке технологий .Net нами была выбрана IDE Microsoft Visual Studio.

Microsoft Visual Studio — это пул программных продуктов от компании Microsoft, включающих, непосредственно, интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Поставляемые программные продукты позволяют охватить полный стек технологий, с их помощью можно реализовать как простые консольные приложения, так и приложения с графическим интерфейсом, начиная с классических Windows Forms и заканчивая современным WPF, также возможно разработка веб-сайтов, веб-приложений, веб-служб. А благодаря технологии .Net Core, программные продукты разработанные, с использованием стека технологий .Net больше не «привязаны» к операционной системе Windows. В настоящий момент эти продукты успешно могут функционировать на всех операционных системах, для этого достаточно установить net core sdk.

Visual Studio постоянно совершенствуется и раз в два года выходит новая версия, но все они включают базовый набор инструментов, таких как редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода, что в значительной степени ускоряет разработку и минимизирует вероятность допущения ошибки, потому что IntelliSense имеет возможность генерации часто используемых конструкций кода, а анализатор кода подскажет не только ошибки в коде и возможные варианты их устранения, но рекомендации как сделать код безопасным. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Так же в наборе инструментов присутствует профайлер, который позволяет провести анализ кода в runtime, выявить какие участки кода занимают больше всего процессорного времени, наиболее часто вызываются или занимают большой объем оперативной памяти или стека. Профайлер позволяет проанализировать

приложение и сделать его оптимальным по использованию ресурсов. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения.

Для разработки веб приложений наиболее применим редактор исходных кодов Visual Studio Code.

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

Visual Studio Code основан на Electron — фреймворк, позволяющий с использованием Node.js разрабатывать настольные приложения, которые работают на движке Blink. Несмотря на то, что редактор основан на Electron, он не использует редактор Atom. Вместо него реализуется веб-редактор Monaco, разработанный для Visual Studio Online.

Для разработки пользовательских интерфейсов на стеке технологий Oracle, одним из вариантов является Oracle JDeveloper.

Oracle JDeveloper — бесплатная интегрированная среда разработки программного обеспечения, разработанная корпорацией Oracle. Предоставляет возможность для разработки на языках программирования Java, JavaScript, BPEL, PHP, SQL, PL/SQL и на языках разметки HTML, XML. JDeveloper покрывает весь жизненный цикл разработки программного обеспечения от проектирования, кодирования, отладки, оптимизации и профилирования до его развёртывания.[4]

Производитель отмечает в качестве основной задачи среды — максимальное использование возможностей визуального и декларативного подхода к разработке программного обеспечения в дополнение к удобной среде кодирования. Oracle

JDeveloper интегрирована с Oracle ADF — Java EE-каркасом для создания коммерческих приложений на Java.

Выводы по главе один

В теоретической части пояснительной записки были рассмотрены варианты уже реализованных систем «Робот-рекрутер», выявлены их преимущества и недостатки. Поставлена задача о создании собственной системы, отвечающей необходимым требованиям для удобства создания, реализации и использования управлением кадров и кандидатами. Выбрана архитектура построения системы – «Клиент-сервер», также выбран тип базы данных – реляционная база данных, в качестве шаблона проектирования всей системы в целом выбрана MVC, в качестве шаблона проектирования web интерфейсов выбран шаблон MVVM. В качестве сред разработки были выбраны Microsoft Visual Studio, Visual Studio Code и Oracle JDeveloper.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

Система «Робот-Рекрутер» содержит следующие модули компоненты:

- Клиентская часть сотрудника кадров
- Клиентская часть сотрудника отдела труда и занятости
- Информационный киоск
- Сайт «Вакансии»
- Система проведения видеоинтервью
- Система распознавания документов государственного образца
- Брокер сообщений

2.1 Клиентская часть сотрудника кадров

На рисунке 2.1 можно увидеть пользовательский интерфейс клиентской части сотрудника кадров. Клиентская часть сотрудника кадров построена с использованием Oracle Jdeveloper и предоставляет доступ к уже отфильтрованной по требованиям, базе откликов. Пользовательский интерфейс предоставляет доступ к таким возможностям как поиск и сортировку по активным вакансиям, в выбранной вакансии отображается список требований, список необходимых документов, штатная позиция, цех и подразделение куда требуется сотрудник, у каждой вакансии есть возможность просмотреть список откликов, по каждому из них можно открыть сформированное резюме, отображающее исчерпывающую информацию о соискателе с приложением сканов документов, которые вложил пользователь, есть функциональность фильтрации соискателей по определенным признакам, интересующему кандидату можно отправить приглашение на очное собеседование или приглашение на прохождение видеоинтервью. После успешного собеседования, сотрудник кадров может выбрать лучшего кандидата и инициировать процедуру трудоустройства, для этого он устанавливает статус «кандидат», выбранному соискателю, в этом случае, остальным соискателям отправляется смс уведомление, о том что вакансия закрыта, но все эти соискатели остаются в базе кандидатов и при открытии этой вакансии вновь или вакансии с похожей профессией или штатной позицией, они сразу будут отображаться в списке кандидатов, тем самым мы не теряем заинтересованных в трудоустройстве соискателей.

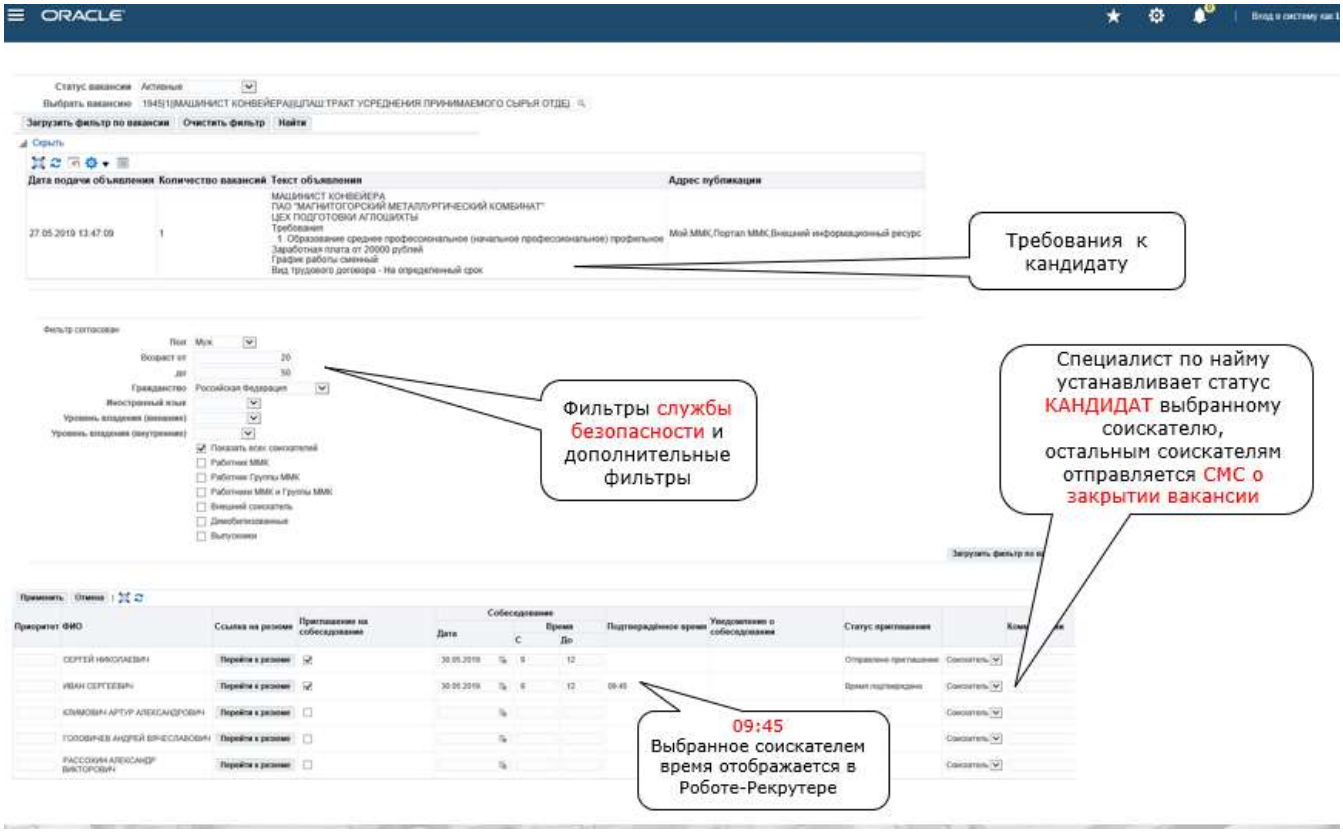

Рисунок 2.1 – Пользовательский интерфейс «Робота-Рекрутера»

Для того чтобы пригласить соискателя на очное собеседование, сотруднику кадров необходимо выбрать дату собеседования, установить временной интервал, в который возможно проведение собеседования и нажать кнопку «Отправить SMS». Выбранному кандидату отправится информационное смс сообщение, содержащее информацию о том, что его пригласили на очное собеседование с указанием вакансии. Помимо информации о вакансии, текс смс будет содержать ссылку на страницу сайта, где у пользователя есть возможность выбрать удобное для него время в предложенном интервале.

На рисунке 2.2 можно увидеть пользовательский интерфейс страницы подтверждения времени. После подтверждения времени пользователем, эта информация отобразится в интерфейсе робота-рекрутера, а самому пользователю придет информационное смс сообщение, которое будет дублировать выбранное пользователем время и дату.

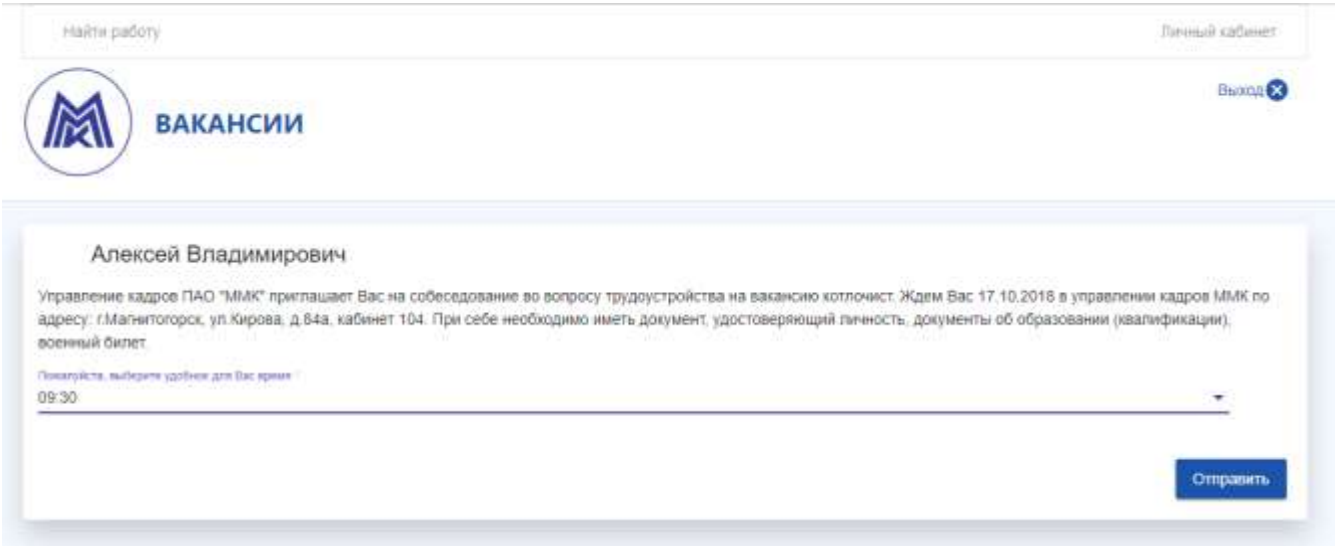

Рисунок 2.2 – Страница подтверждения времени

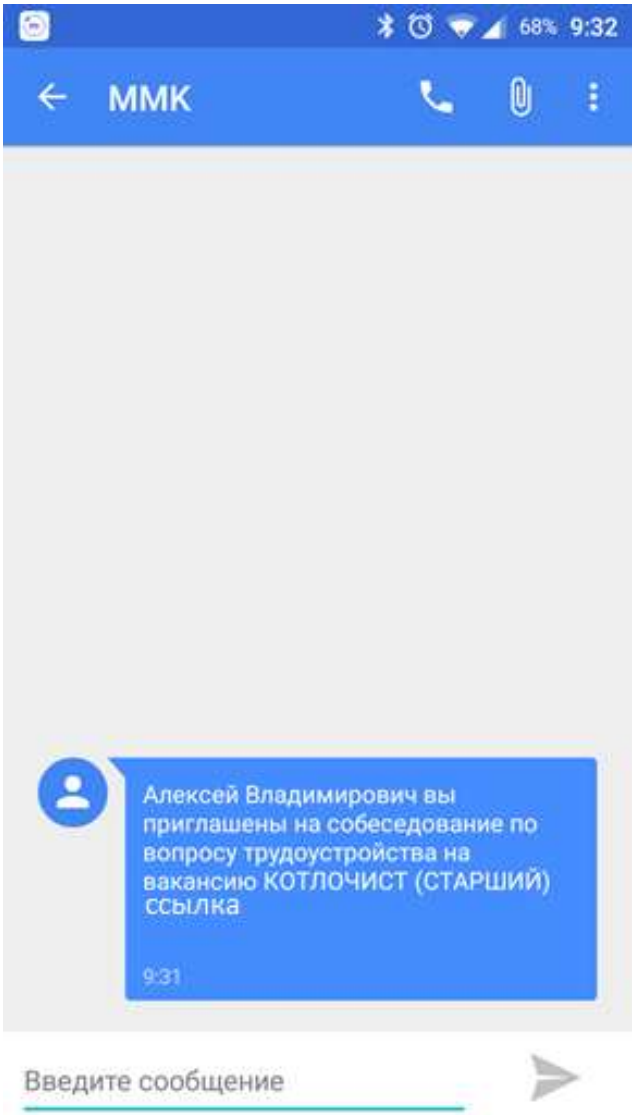

Рисунок 2.3 – Приглашение на собеседование

2.2 Клиентская часть сотрудника отдела труда и занятости

Перед тем как соискатель сможет оставить отклик на вакансию, эту вакансию необходимо сформировать и опубликовать. Этим занимаются специалисты отдела труда и занятости (ОТиЗ), непосредственно в подразделениях. Они формируют заявку на комплектование, указывают штатную позицию, дополнительные требования, список необходимых документов, после чего руководителям отправляется уведомление на электронную почту о размещении вакансии, а сама вакансия размещается в публичном доступе.

На рисунке 2.4 можно увидеть интерфейс пользователя клиентской части специалиста труда и занятости

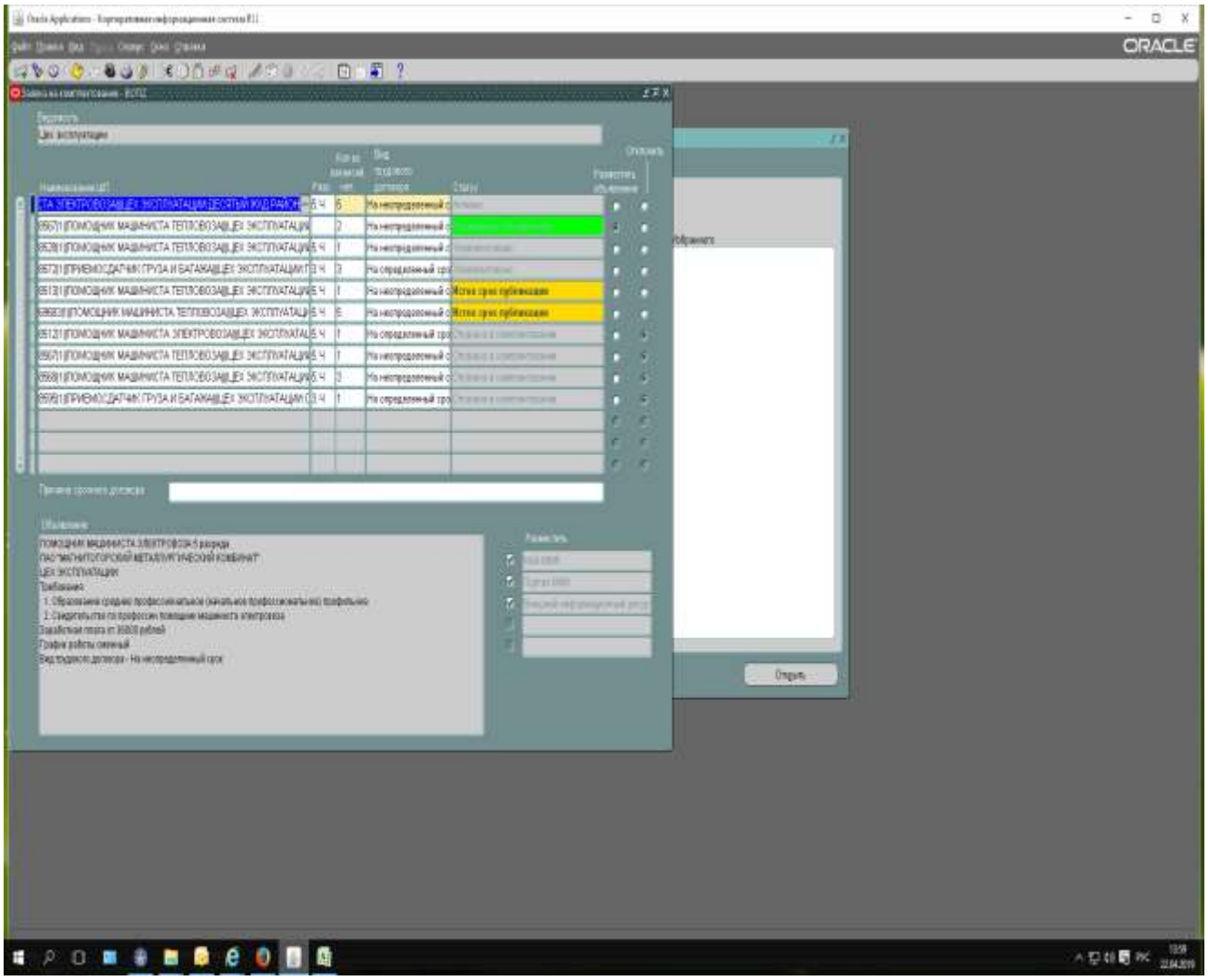

Рисунок 2.4 – Пользовательский интерфейс специалист ОТиЗ

Пользовательский интерфейс специалиста ОТиЗ построен с использованием технологии Oracle Forms

Oracle Forms — это программный продукт корпорации Oracle, предназначенный для создания приложений баз данных, в котором создаются модули форм. Помимо этого, он включает в себя оболочку для разработки меню и библиотечных модулей PL/SQL. [4]

Oracle Forms получают доступ к базе данных Oracle и генерируют экранные формы отражающие данные базы. Форма в исходнике (*.fmb) компилируется в выполнимую форму (*.fmx) которая может запускаться независимо из модуля запуска форм. Форма используется для отображения и редактирования данных в приложениях, управляющих базой данных. Некоторые элементы графического интерфейса пользователя GUI такие как кнопки, меню, области прокрутки и графики могут быть размещены на форме. [7]

С помощью Oracle Forms можно создать небольшое приложение, которое позволит пользователю просматривать или изменять данные БД Oracle. Все данные приложения хранятся в БД, имеющей известную заранее структуру, поэтому формы создаются с использованием структуры уже готовых таблиц БД.

Как и во всех обычных интерфейсах, управляемых событиями, ПО выполняет функции, привязанные к событиям, называемые триггерами, вызывает триггеры, которые:

- автоматически отрабатывают в определенные моменты работы с записями,
- обрабатывают нажатие клавиш и движение мышкой.

При инициализации триггера определяется то, что будет выполнено данной операцией. В связи с этим программирование Oracle Forms заключается в том, чтобы надлежащим образом изменить начальные действия триггеров. Некоторые триггеры, предусмотренные программистом, заменяют стандартные действия, тогда как другие — расширяют их.

В результате применения подобного подхода, становится возможным создание нескольких стандартных форм, которые бы реализовывали полную функциональность базы данных, пока не содержится вообще никакого кода, написанного программистом.

2.3 Информационный киоск

Для большего охвата кандидатов, в управлении кадров ПАО «ММК», установлен информационный киоск. Средствами информационного киоска можно зарегистрироваться в системе, заполнить резюме и отсканировать документы.

Сам киоск был заказан у компании BM Group «Фабрика инноваций», специализирующейся на разработке информационных киосков и интерактивных стендов.

BM Group «Фабрика инноваций» является лидером Российского рынка по производству интерактивного оборудования и первым заводом интерактивного оборудования в России. Более 4 лет BM Group «Фабрика инноваций» успешно реализует самые разнообразные проекты от точечных поставок интерактивных развивающих столов в дошкольные учреждения до сложного комплексного оснащения объектов Государственных масштабов. Так в 2018 году завод компании BM Group «Фабрика инноваций» выпустил более 2000 интерактивных устройств для бизнеса, Государственных учреждений и объектов инфраструктуры городов России.

С нашей стороны было разработаны дизайн-эскиз информационного киоска, исходя из требуемого функционала, рассчитаны необходимые вычислительные мощности, подобраны требуемое оснащение и компоненты информационного киоска, так же было разработано техническое задание.

Корпус киоска выполнен из композитного алюминия и брендирован в стилистике группы компаний ПАО «ММК» исходя из рекомендаций, указанных в брендбук ПАО «ММК». Киоск оснащен сенсорным экраном, выполненным в антивандальном исполнением. Все внутренние компоненты информационного киоска подключены через источник бесперебойного питания, что позволяет защитить внутренние компоненты от скачков электрического потенциала и нивелировать помехи в сети 220В, а также произвести корректное выключение информационного киоска, в случае отключения электроэнергии. В качестве вычислительных мощностей используется процессор Intel Pentium GOLD, имеющий 2 ядра и 4 потока, работающих на частоте 3.7Ггц, что дает возможность производить ресурсоемкие вычисления, которые требуются произвести при распознавании документов. Так же в киоск установлен сканер QR-кодов, планшетный сканер, принтер для печати чеков, камера и акустическая система.

Веб-приложение, размещенное в киоске, представляет из себя портированный под задачи информационного киоска сайт с вакансиями, добавлена возможность сканирования документов на планшетном сканере информационного киоска через веб-интерфейс.

Для реализации возможности инициировать сканирование и получить изображение через веб интерфейс, была разработана библиотека, написанная на языке C# и обернута в asp.net сервис, функционирующий на локальном веб сервере IIS, запущенном непосредственно на киоске. Для взаимодействия со сканером, эта библиотека использует технологию WIA.

C# (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.[1]

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ и некоторых других языков, не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

IIS (Internet Information Services) — набор сервисов от компании Microsoft для работы веб-сервера и других интернет служб. IIS устанавливается на сервер и работает с протоколами HTTP/HTTPS, POP3, SMTP, FTP, NNTP.[13]

ASP.NET (Active Server Pages для .NET) — платформа разработки веб-приложений, в состав которой входит: веб-сервисы, программная инфраструктура, модель программирования, от компании Майкрософт. ASP.NET входит в состав платформы .NET Framework и является развитием более старой технологии Microsoft ASP.

Поскольку ASP.NET основывается на Common Language Runtime (CLR), которая является основой всех приложений Microsoft .NET, разработчики могут писать код для ASP.NET, используя языки программирования, входящие в комплект .NET Framework (C#, Visual Basic.NET, J# и JScript .NET).

Программная модель ASP.NET основывается на протоколе HTTP и использует его правила взаимодействия между сервером и браузером. При формировании страницы заложена абстрактная программная модель Web Forms и на ней основана основная часть реализации программного кода.

Windows Image Acquisition (WIA) – или «Служба загрузки изображений Windows (WIA)» это модель драйверов от компании Майкрософт, а также API для Windows ME и более поздних операционных систем семейства Windows. Данная модель позволяет графическим программам взаимодействовать с оборудованием таким как сканеры, цифровые фотоаппараты и видеокамеры. Впервые

представлена в 2000 году как часть Windows ME и продолжает оставаться стандартной моделью устройств обработки изображений и API в последующих версиях Windows.[1]

Для того чтобы получить изображение со сканера, веб приложение вызывает локальный сервис, который обращается к сканеру, получает изображение и возвращает его в качестве ответа.

На рисунке 2.5 можно увидеть внешний вид информационного киоска.

Рисунок 2.5 – Внешний вид информационного киоска

2.4 Сайт «Вакансии»

Сайт с вакансиями доступен для всех желающих через сеть интернет, имеет адаптивную верстку, позволяющую ему, корректно отображаться в современных браузерах мобильных и стационарных устройств таких как телефоны, планшеты, ноутбуки или компьютеры имеющие различные разрешения экранов и операционные среды, что делает этот сайт кросс-платформенным и кросс-браузерным.

Сайт предоставляет возможность зарегистрироваться, либо авторизоваться в системе, если регистрация была проведена ранее, просмотреть список открытых вакансий, при необходимости есть возможность фильтрации данных по группе компании или профессии, так же сайт предоставляет возможность оставить отклик на любую вакансию для зарегистрированных пользователей, отредактировать свое резюме в личном кабинете, если ранее был произведен отклик.

С целью обеспечения безопасности персональных данных, система авторизации имеет двухфакторную аутентификацию по средством отправки смс кодов. А система регистрации включает в себя валидацию введенных данных и подтверждение номера телефона через смс-код.

Одной из ключевых особенностей системы «Робот-Рекрутер», является то, что система, должна отличать внутренних соискателей, те, что работали или работают в группе ПАО «ММК», от внешних. Для этого проводится поиск данных, указанных пользователем при регистрации, в корпоративной информационной системе (КИС). Для внутренних соискателей, резюме частично заполняется данными, содержащимися в КИС, а также дополнительно отображает информацию об образовании, ранее занесенную в КИС сотрудниками отдела труда и занятости ОТиЗ.

На рисунке 2.6 можно увидеть внешний вид страницы с вакансиями

Сайт реализован на фреймворке Angular 6.

Angular 6 — фреймворк с открытым исходным кодом, написанный на JavaScript и поддерживаемый компанией Google. Он представляет собой полностью переработанную версию своего популярного предшественника, AngularJS. С помощью Angular возможна разработка приложений на JavaScript (применяя синтаксис ECMAScript 5 или 6), Dart или TypeScript.[9]

Прежде всего он нацелен на разработку SPA-решений (Single Page Application), то есть одностраничных приложений.

Angular предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте

интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и так далее.

Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript.

TypeScript – язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

Разработчиком языка TypeScript является Андерс Хейлсберг, создавший ранее Turbo Pascal, Delphi и C#.

Спецификации языка открыты и опубликованы в рамках соглашения Open Web Foundation Specification Agreement (OWFa 1.0).

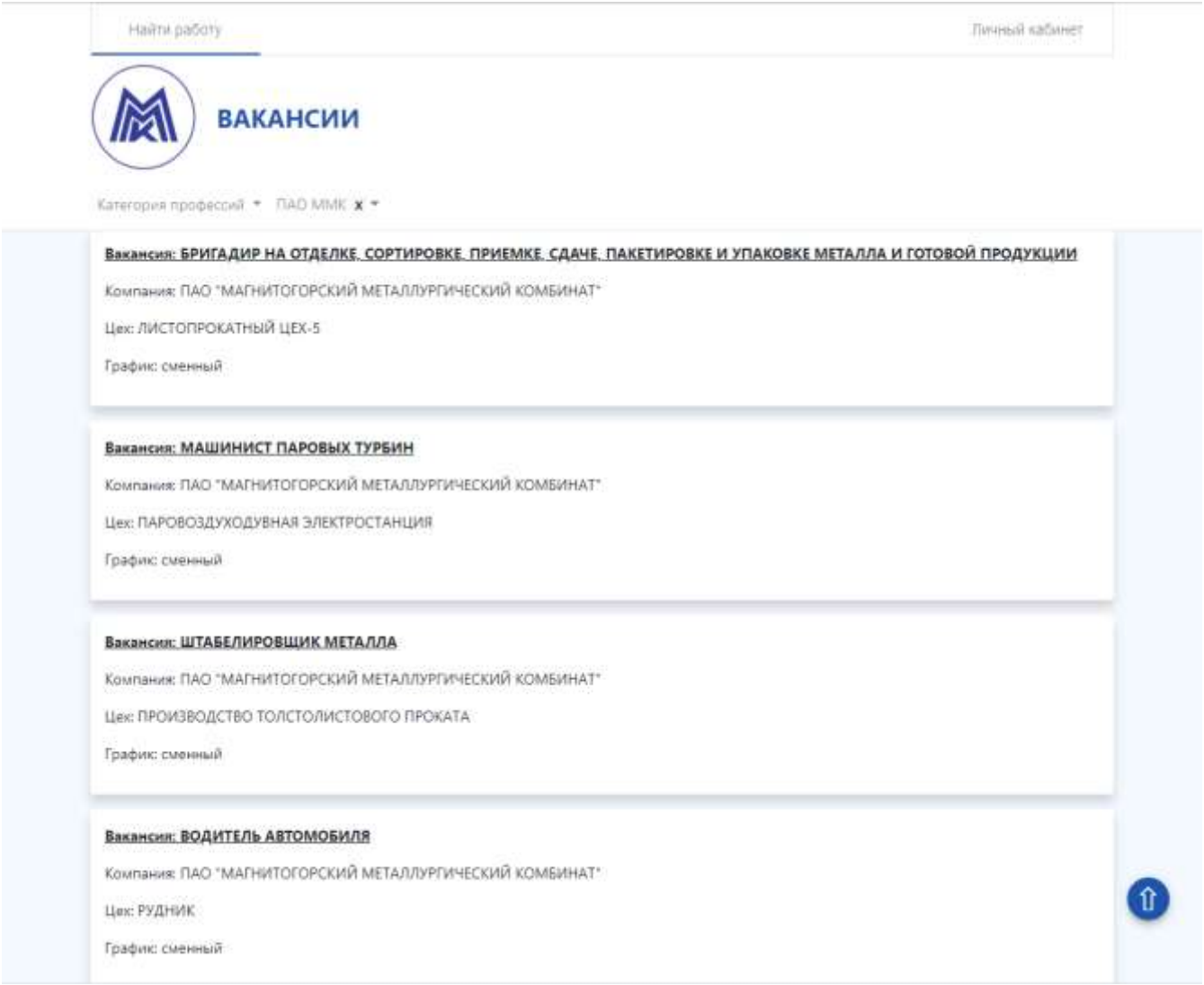

Рисунок 2.6 – Интерфейс страницы с вакансиями

TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js. Код экспериментального компилятора, транслирующего TypeScript в JavaScript, распространяется под лицензией Apache.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

Для того чтобы реализовать адаптивную верстку использовался CSS фреймворк Bootstrap

Bootstrap – это инструментарий с открытым исходным кодом для разработки с помощью HTML, CSS и JS. Используйте переменные Sass и миксины, гибкую систему сеток, множество готовых компонентов и мощных плагинов, основанных на jQuery.[10]

Благодаря его гибкой системе сеток, у разработчика появляется возможность динамически менять компоненты на странице, отображать их или скрывать в зависимости от размера экрана или его ориентации.

В качестве направления дизайна был выбран стиль Material Design – это язык дизайна для веб- и мобильных приложений, который был разработан Google в 2014 году. Material Design упрощает разработчикам настройку UI, сохраняя при этом удобный интерфейс приложений.

Преимущественно компоненты Angular Material, такие как кнопки, элементы меню, вкладки, текстовые поля и многие другие, использовались в разработке сайта.

Взаимодействие сайта с бэкендом строится по RESTful архитектуре. Этот архитектурный стиль позволяет сделать распределенный бэкенд, причем он может быть не только распределен в сети, но и быть реализован при помощи разных стеков технологий.[12]

Сам бэкенд реализован средствами oracle apex – позволяющей сделать любую процедуру PL/SQL доступной как rest – сервис.

Практически все сервисы защищены токеном, имеющим ограниченное время действия.

На текущий момент, все запросы к бэкенду идут через кастомный брокер сообщений, что также позволяет контролировать запросы к бэкенду и осуществлять безопасность данных.

На рисунке 2.7 можно увидеть внешний вид личного кабинета

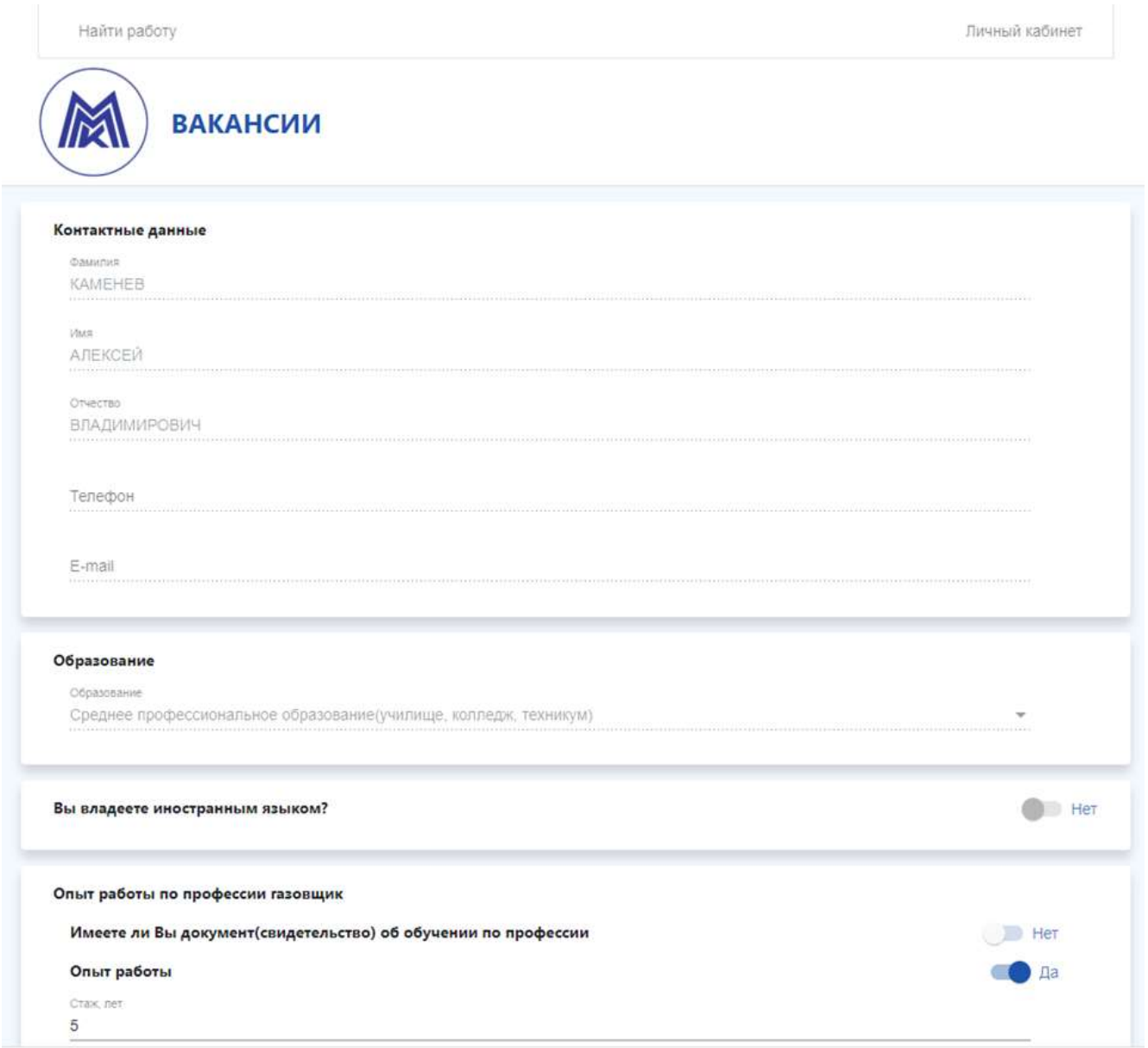

Рисунок 2.7 – Интерфейс личного кабинета сайта

2.5 Система проведения видеоинтервью

Для составления более полного представления о соискателе и для реализации возможности проведения интервью с соискателями из удаленных районов, была разработана система проведения видеоинтервью.

Система видеоинтервью состоит из административной панели, позволяющей создавать справочники с вопросами, отправлять приглашение на видеоинтервью,

просматривать записанные кандидатами видео файлы и пользовательской части, позволяющей записать видеоинтервью, просмотреть ролик и отправить его на сервер.

На рисунках 2.8, 2.9 и 2.10 можно увидеть пользовательский интерфейс административной панели видеоинтервью

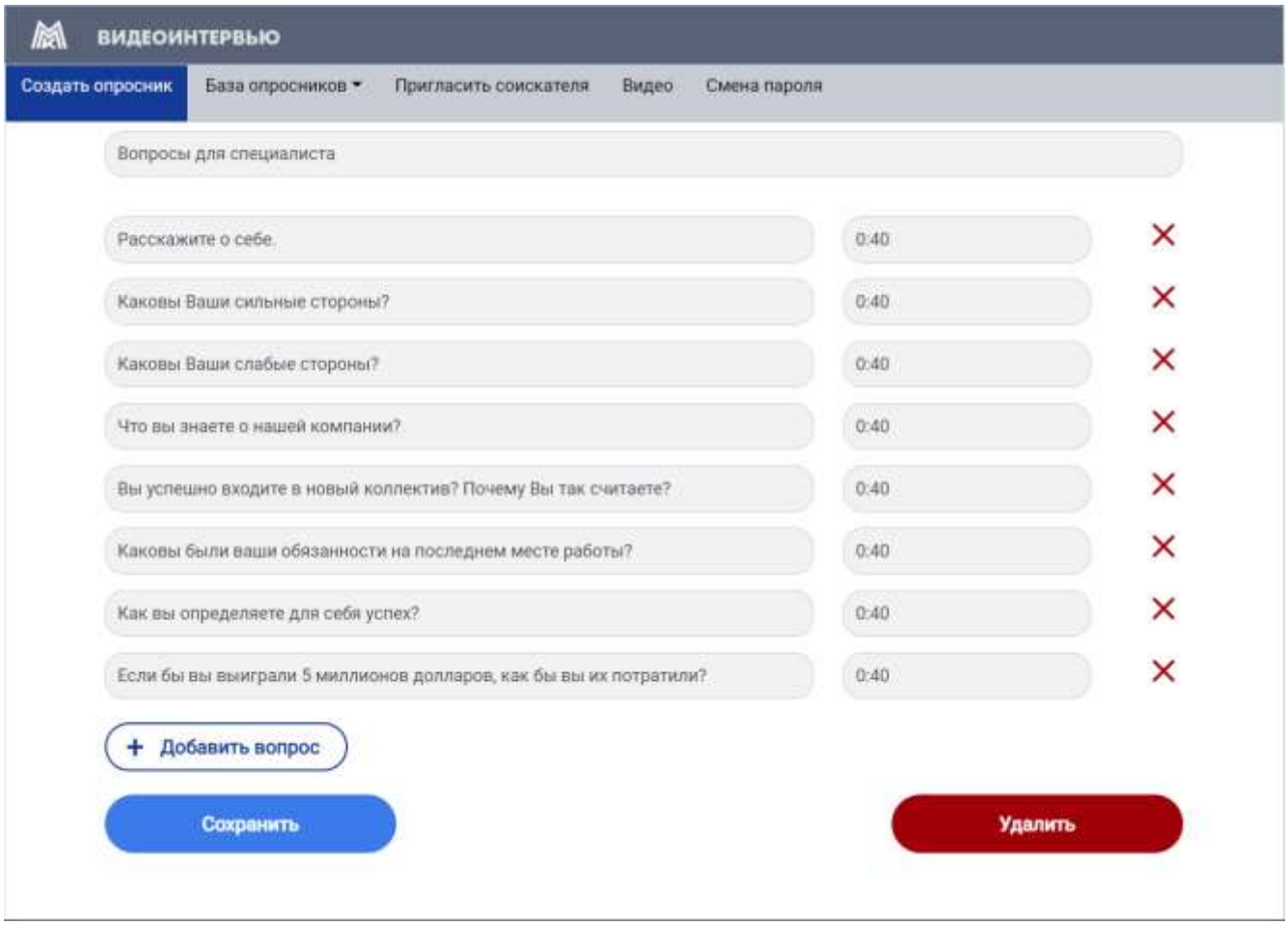

Рисунок 2.8 – Интерфейс страницы формирования опросников

После того как специалист управления кадров выберет соискателей, необходимо выбрать справочник с вопросами, на которые предстоит ответить соискателю. После того как специалист пригласит соискателя, ему придет смс, о том, что он приглашен и электронное письмо с краткими инструкциями о прохождении видеоинтервью.

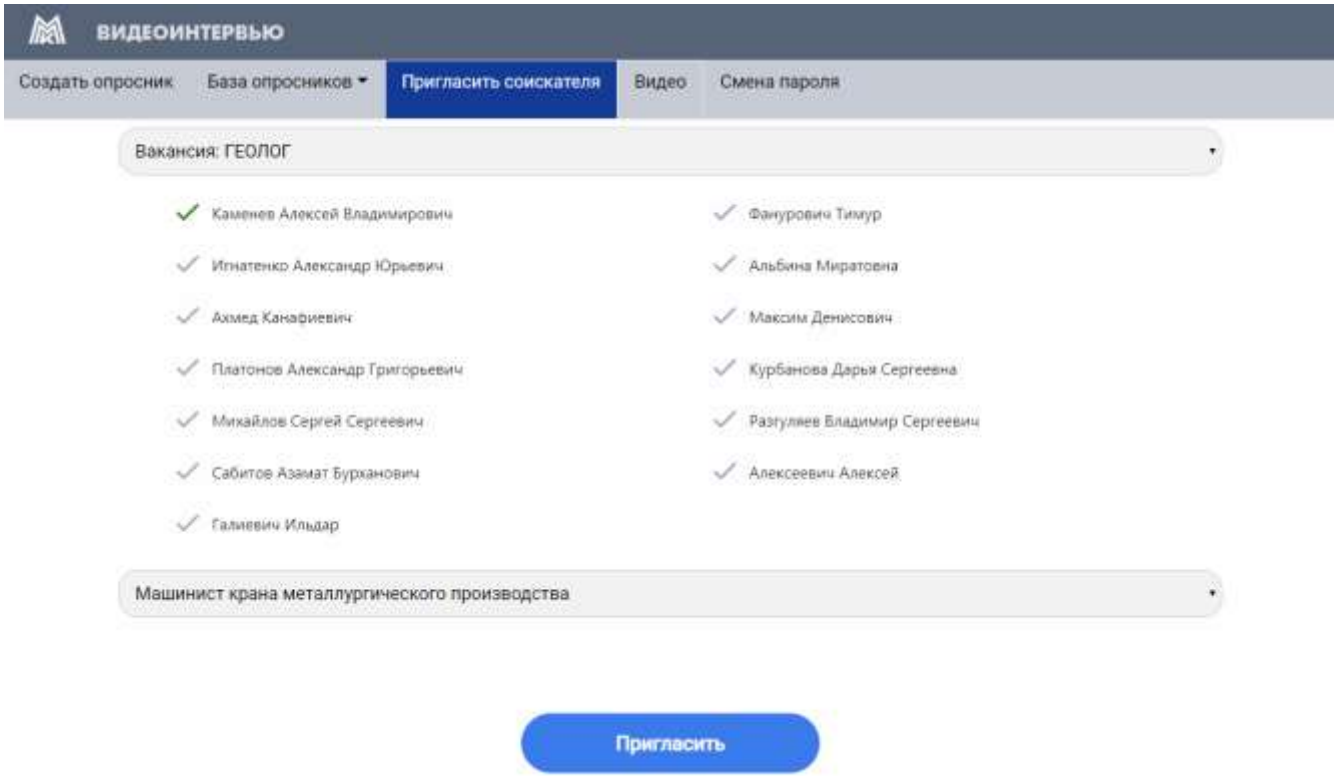

Рисунок 2.9 – Интерфейс страницы отправки приглашения

Клиентская часть видеоинтервью реализована при помощи технологии JavaScript, HTML5 и CSS. Для взаимодействия с камерой устройства, на котором запущен сайт для проведения видеоинтервью, используется интерфейс MediaDevices, предоставляемый браузерами. Этот интерфейс предоставляет доступ к потоку с камеры устройства. Этот поток записывается средствами интерфейса MediaRecorder, так же предоставляемого браузерами.[6] После чего записанные данные отправляются на сервер.

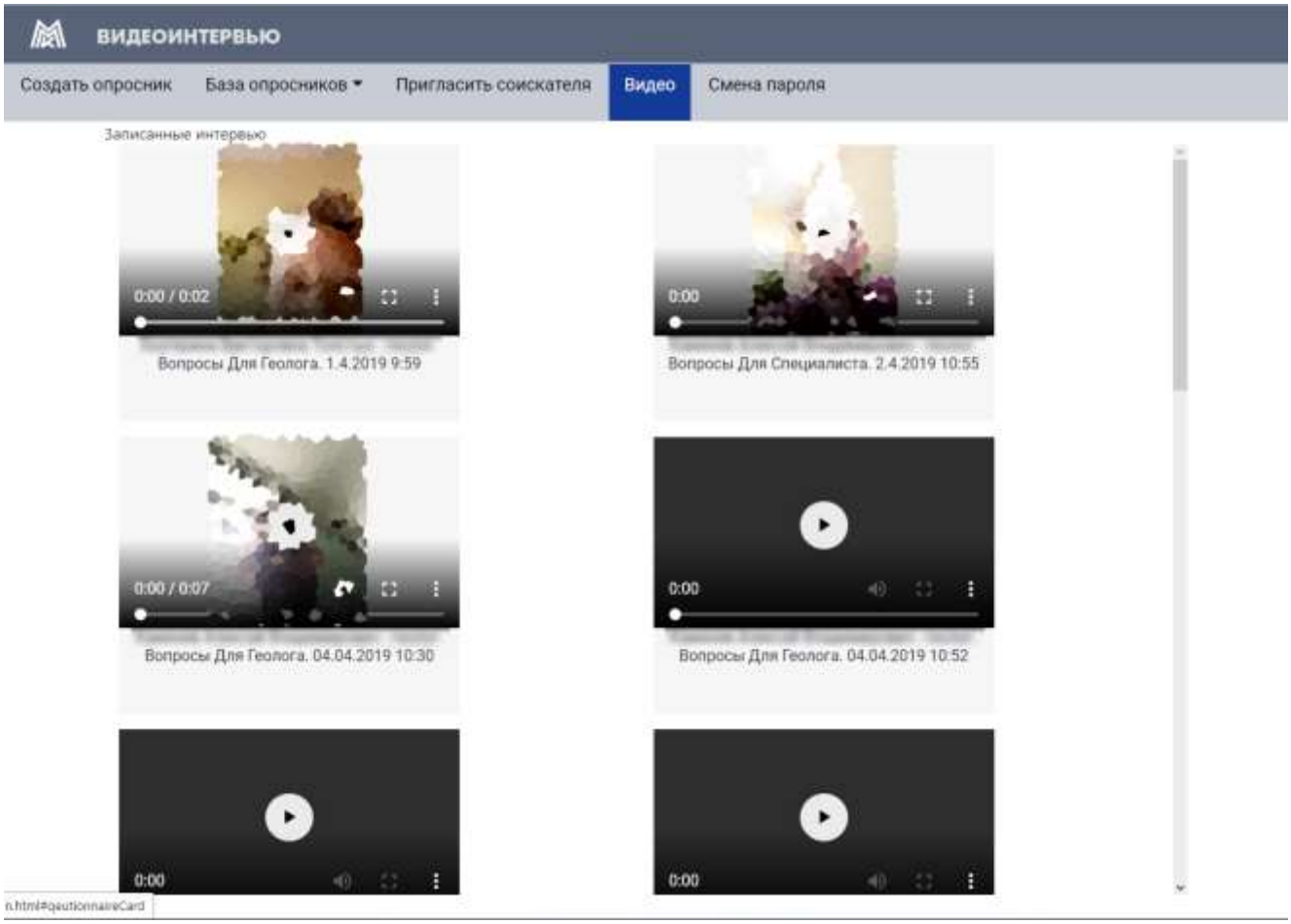

Рисунок 2.10 – Интерфейс страницы просмотра записанных интервью

Из-за того, что браузер Safari на мобильных устройствах apple не поддерживает интерфейс MediaRecorder, возникла необходимость разработать мобильное приложение для мобильных устройств на операционной системе iOS. Мобильное приложение было разработана средствами фреймворка Xamarin. Это фреймворк для кроссплатформенной разработки мобильных приложений (iOS, Android, Windows Phone) с использованием языка C# и всех возможностей среды .Net, но при этом остается полный доступ ко всем возможностям SDK платформы и родному механизму создания UI, получая на выходе приложение, которое, строго говоря, ничем не отличается от нативных и не уступает им в производительности.

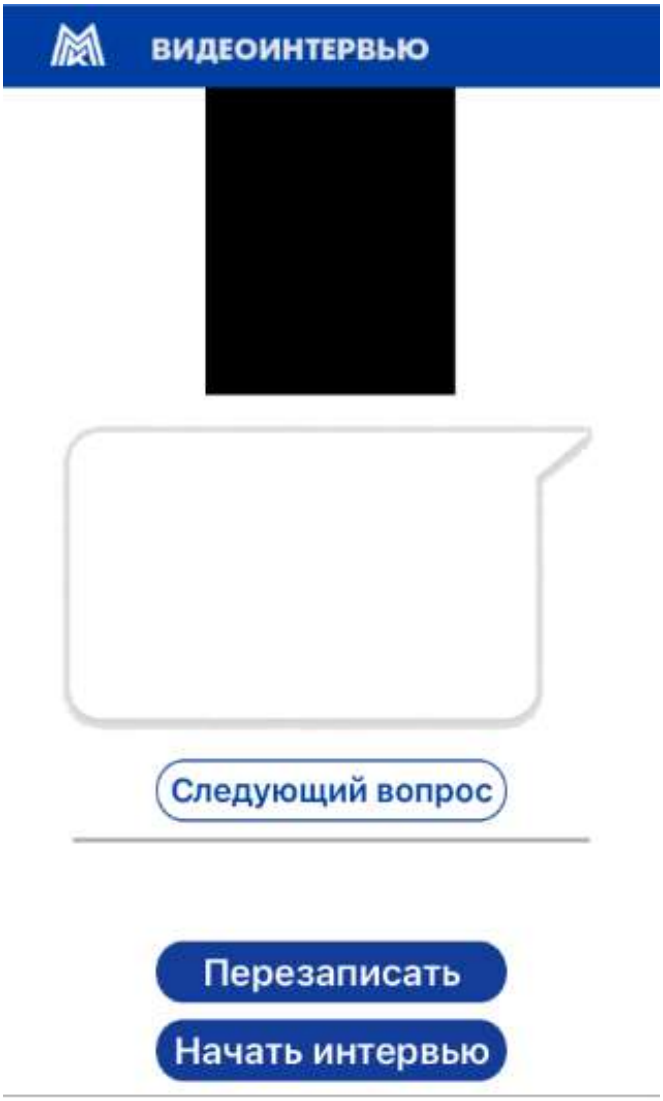

Рисунок 2.11 – Пользовательский интерфейс мобильного приложения

JavaScript проверяет UserAgent устройства, с которого открыли страницу, если это iOS, то предлагается скачать приложение с нашего сервера, установить его и воспользоваться ссылкой. Приложение регистрируется в системе как целевое для открытия ссылок определенного формата, поэтому нажав на ссылку на сайте, откроется приложение, которое позволит записать и отправить видеоинтервью.

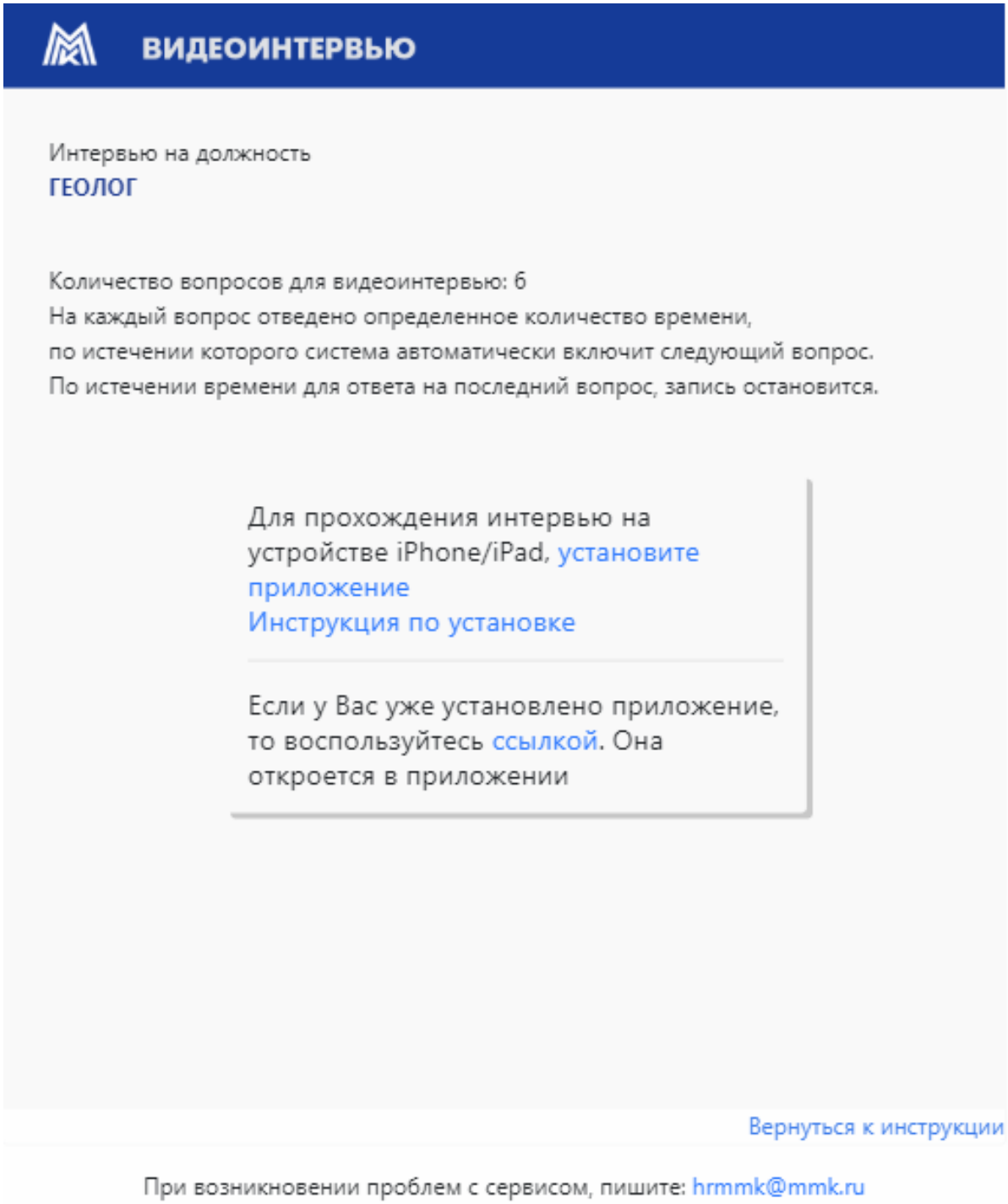

Рисунок 2.12 – Интерфейс страницы для устройст Apple

Пользовательский интерфейс для соискателей можно увидеть на рисунке 2.13. После того как пользователь перейдет по ссылке, ему будет показана страница с краткими инструкциями о процессе прохождения интервью, после того как пользователь согласится с правилами, он попадет на основную страницу прохождения видеоинтервью, на ней он должен дать разрешение на доступ к камере и микрофону, после чего станет активная кнопка начать интервью. Для каждого вопроса отведено ограниченное время, по истечении этого времени, вопрос переключится на следующий. Вопросы, на которые отвечает пользователь,

записываются титрами к видео и отображаются в панели администрирования. По истечении времени ответа на последний вопрос, запись автоматически прекращается. Пользователю предоставляется возможность его просмотреть, отправить или перезаписать.

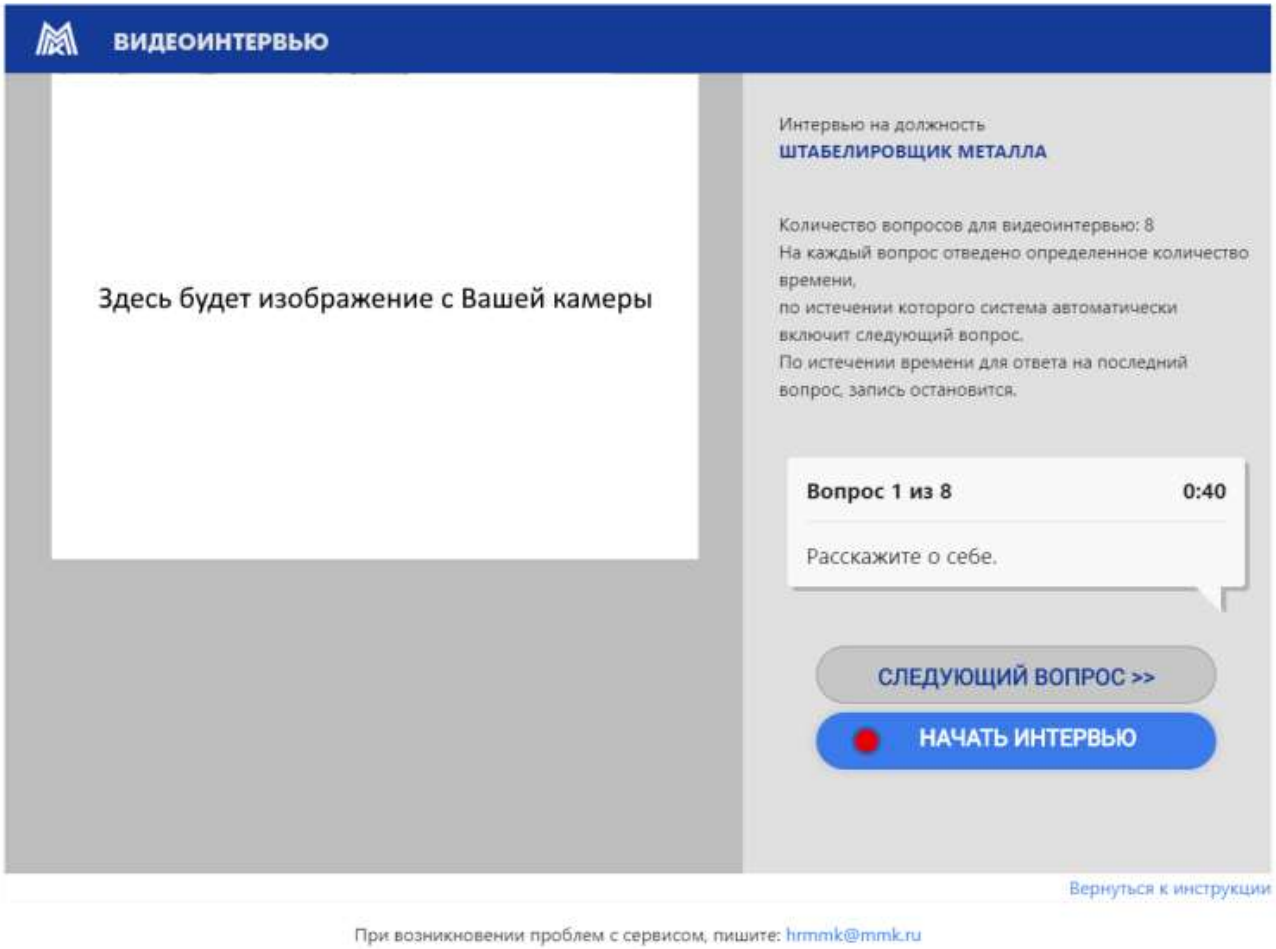

Рисунок 2.13 – Пользовательский интерфейс для соискателей

2.6 Система распознавания документов государственного образца

Система распознавания документов государственного образца, таких как, паспорт, СНИЛС и ИНН. Система реализована на языке C#. Для обработки изображения используется алгоритмы компьютерного зрения, входящие в пакет Image, фреймворка Accord.NET Framework. Непосредственно распознавание текста происходит с помощью библиотеки Tesseract OCR.

Accord.NET Framework - это среда машинного обучения .NET в сочетании с библиотеками обработки звука и изображений, полностью написанными на C#. Это полная структура для создания приложений для компьютерного зрения промышленного уровня, компьютерного прослушивания, обработки сигналов и статистики даже для коммерческого использования.

Tesseract - свободная компьютерная программа для распознавания текстов, разрабатывавшаяся Hewlett-Packard с середины 1980-х по середину 1990-х, а затем 10 лет «пролежавшая на полке». В августе 2006 г. Google купил её и открыл исходные тексты под лицензией Apache 2.0 для продолжения разработки.

Распознавание можно разделить на 3 этапа.

1 Препроцессинг изображения.

На этом этапе происходит нормализация изображения – приведение к заданному размеру, коррекция яркости и контрастности, выравнивание изображение по горизонту, поиск целевого объекта на изображении и отделение его от фона, поиск и установка правильной ротации текста. После этого необходимо изображение удалить с изображения шум, мусор и лишние элементы. В результате этого этапа на изображении останутся только значимый текст, который будет размечен и отправлен на этап распознавания.

Для того чтобы отделить значимый объект от фона, изображение бинарузется и инвертируется, после чего строится пирамида изображения.

Изображение сцены может быть представлено в различных пространственных масштабах. При этом крупные детали сцены лучше видны на изображениях с мелким (грубым) разрешением. Мелкие детали сцены проявляются только на изображениях с высоким разрешением. Изображение, представленное в нескольких масштабах, в дальнейшем называется пирамидой.

Использование пирамидальной структуры данных при сопоставлении изображений имеет две основные цели:

1. сокращение времени обработки изображений
2. определение более точных начальных приближений для обработки нижних уровней по результатам обработки верхних уровней.

Пирамида изображений представляет собой последовательность N изображений, причем каждое последующее изображение получается из предыдущего путем фильтрации и прореживания в два раза.

На рисунке 2.14 можно увидеть один из слове пирамиды изображения. На этом изображении четко видны контуры паспорта, затем используя класс BlobCounter фреймворка Accord.NET Framework, мы легко находим этот объект, получаем его координаты и переносим их с масштабированием на основное изображение, тем самым получая координаты значимого объекта, которые отделяем от фона изображения.

Рисунок 2.14 Слой пирамиды изображения

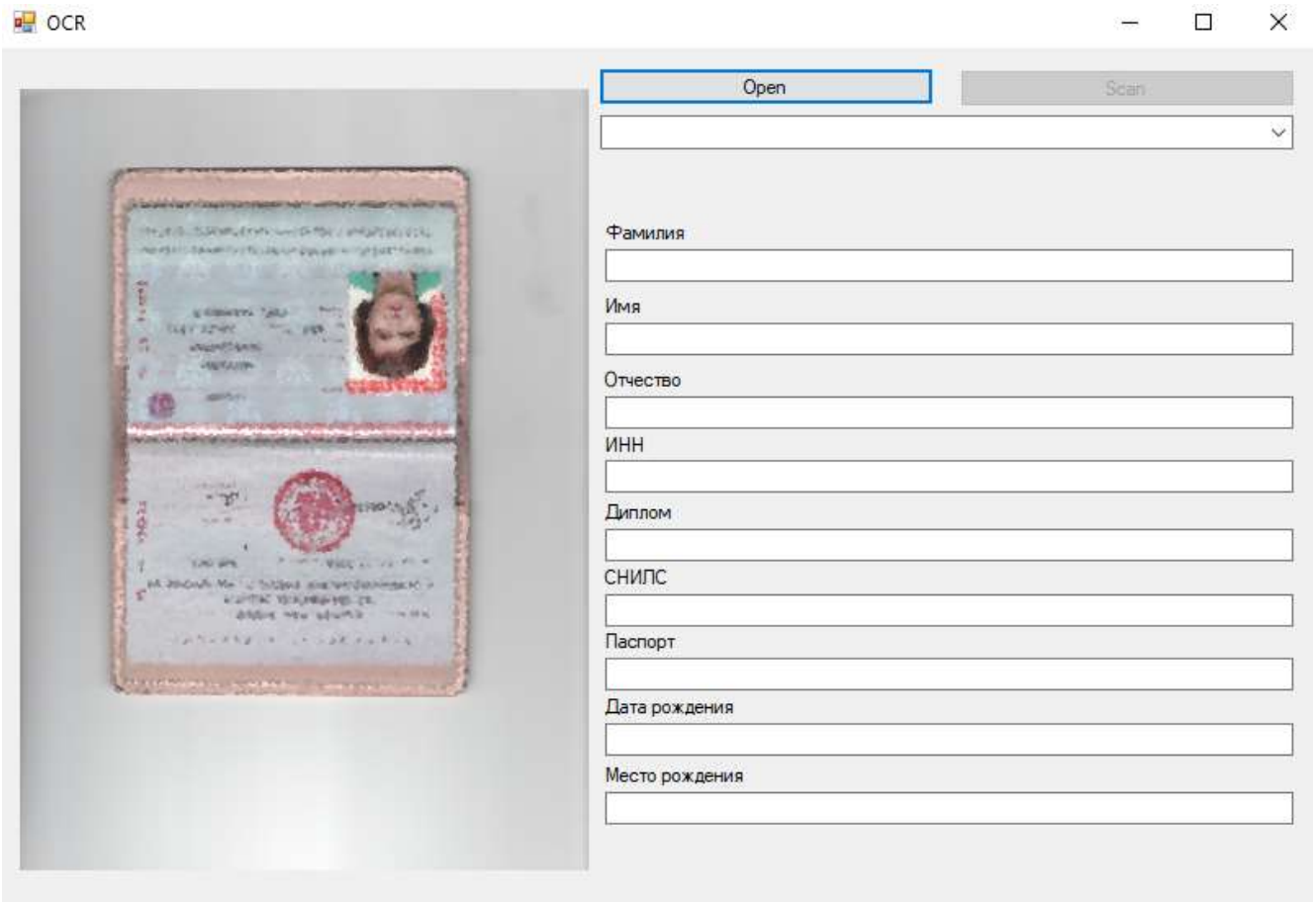

Рисунок 2.15 – Интерфейс системы распознавнаия документов

2 Распознавание

На этапе распознавания текст с изображения переводится в строковый формат и передается на этап аналитической обработки.

3 Аналитическая обработка результатов

На этом этапе производится аналитическая обработка информации, и получение из текста именованных сущностей. Именно на этом этапе мы ищем в

тексте номера СНИЛС, понимаем, что является фамилией, что именем, что датой рождения, а что местом жительства.

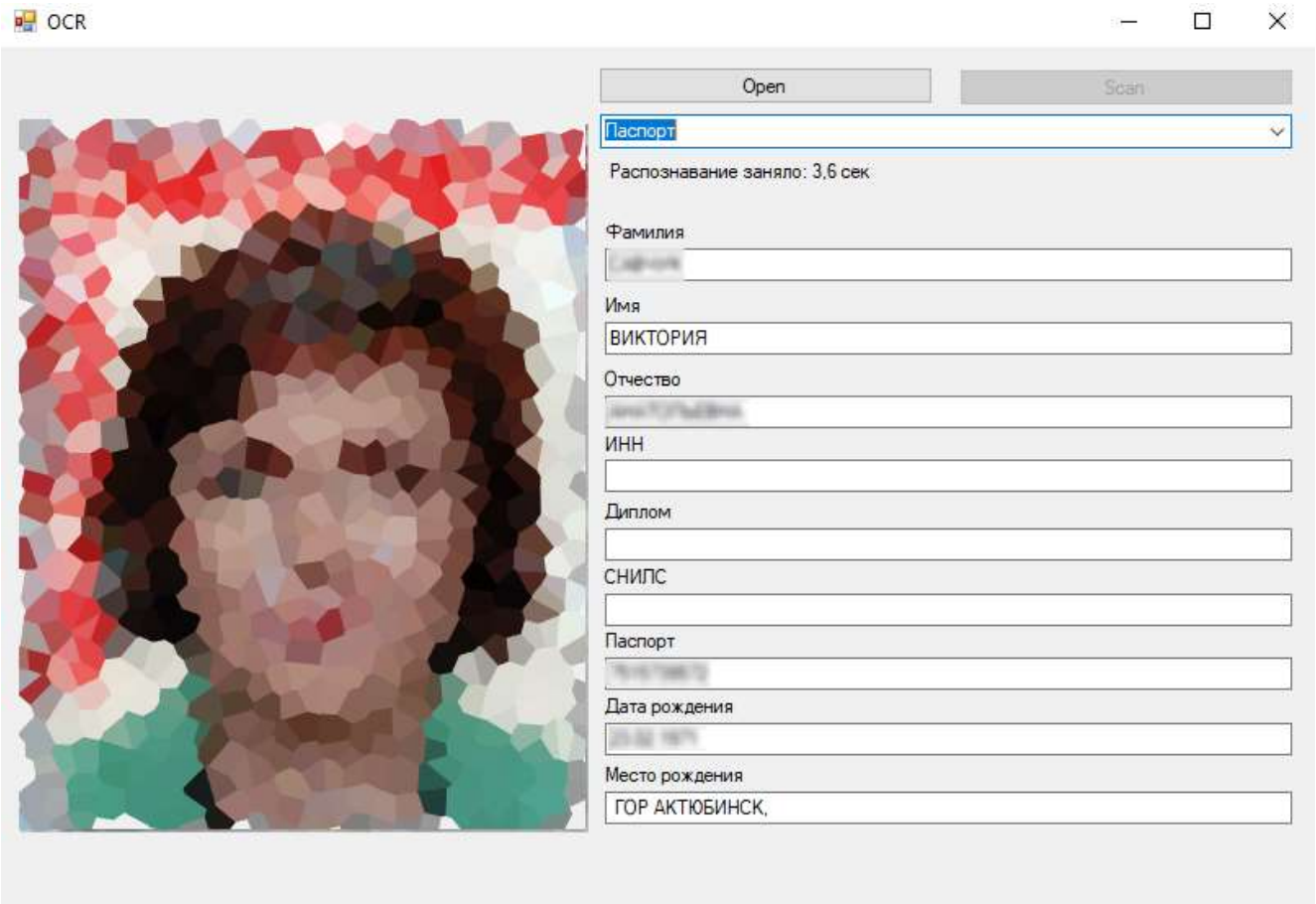

Рисунок 2.16 – Результат распознавания паспорта

2.7 Брокер сообщений

Для того чтобы система стала общедоступной через интернет и вышла за пределы корпоративной сети ПАО «ММК», возникла необходимость разместить сервер системы в демилитаризованную зону (DMZ)

DMZ — сегмент сети, содержащий общедоступные сервисы и отделяющий их от частных. В качестве общедоступного может выступать, например, веб-сервис: обеспечивающий его сервер, который физически размещён в локальной сети, должен отвечать на любые запросы из внешней сети, при этом другие локальные ресурсы (например, файловые серверы, рабочие станции) необходимо изолировать от внешнего доступа.[11]

Цель ДМЗ — добавить дополнительный уровень безопасности в локальной сети, позволяющий минимизировать ущерб в случае атаки на один из общедоступных сервисов: внешний злоумышленник имеет прямой доступ только к оборудованию в ДМЗ.

В связи с тем, что политикой информационной безопасности ПАО «ММК», запрещен доступ из DMZ в локальную сеть, возникла необходимость в предоставлении серверу в ДМЗ данных из локальной сети и при этом эти данные должны быть оперативно предоставлены.

Для того чтобы запрашивать и доставлять данные серверу в DMZ, мною было принято решение разработать брокер сообщений.

Архитектура брокера сообщений выглядит следующим образом, в зоне DMZ размещается база данных, в качестве СУБД для этих целей был выбран PostgreSQL.

К этой СУБД, размещенной в ДМЗ, подключается веб сервер IIS, так же размещенный в ДМЗ и сервер, размещенный в корпоративной сети.

На веб сервере IIS размещен REST-сервис, принимающий POST-запрос, содержащий в себе информацию об целевом адресе в локальной сети, типе запроса, в настоящий момент реализованы запросы двух типов GET и POST, параметры с которыми необходимо выполнить эти запросы, заголовки и вложенные файлы. Сервис запаковывает эти данные с специально разработанный класс, сериализует экземпляр этого класса в массив байт. Этот массив байт записывает в таблицу базы данных, как результат запроса, получает идентификатор, появление этот идентификатор он ждет в таблице с результатами, в ней он берет результат выполнения и возвращает тому, кто запрашивал данные.

На сервере в локальной сети запущен многопоточный клиент, который в несколько потоков (на текущий момент 2 потока) опрашивает таблицу с заданиями в базе данных, размещенной в DMZ. Каждый потом берет свободное задание из этой таблицы, помечает его, чтобы другой поток не взял то же самое задание, десериализует его в экземпляр класса, выполняет задание, результат записывает в таблицу с результатами.

На рисунке 2.17 можно увидеть схему информационных потоков брокера сообщений.

PostgreSQL - свободная объектно-реляционная СУБД.

Сильными сторонами PostgreSQL считаются: [8]

- высокопроизводительные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки C-совместимых модулей;
- наследование;
- легкая расширяемость.

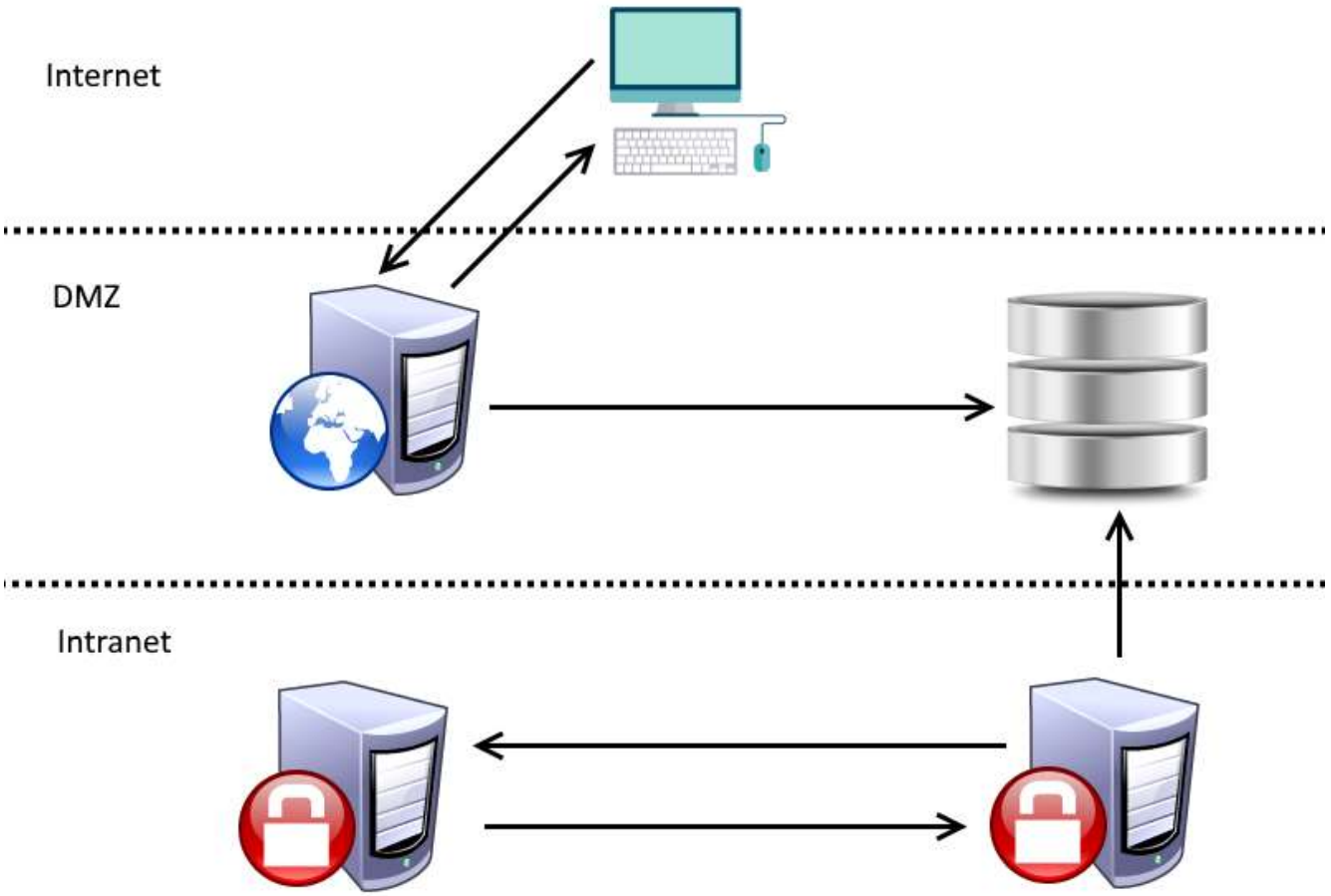

Рисунок 2.17 – Схема информационных потоков брокера сообщений

Выводы по главе два

Разработаны клиентская часть сотрудника кадров, клиентская часть сотрудника отдела труда и занятости, информационный киоск, сайт «Вакансии», система проведения видеоинтервью, система распознавания документов государственного образца, брокер сообщений.

3 РЕАЛИЗАЦИЯ СИСТЕМЫ

Реализация системы и вводы в эксплуатацию проходила в 5 этапов.

3.1 Этап бизнес анализа и формализации технического задания

Основная задача бизнес-анализа – это сделать возможным проведение изменений в организации, путем реализации выбранного решения. Решение разрабатывается с целью устранения выявленных в процессе бизнес-анализа бизнес-проблем. Понятие Решение включает в себя широкий диапазон возможных путей устранения выявленных бизнес-проблем: разработка новых или изменение существующих бизнес-процессов или бизнес-правил, оптимизация организационной структуры организации, разработка новых стратегических планов организации и т.п.

Исторически сложилось так, что бизнес-анализ был наиболее востребован и долгое время развивался в области информационных технологий. В данной области наиболее распространенным Решением является автоматизация бизнес-процессов организации, т.е. разработка информационной системы. Именно этот путь выбрало руководство ПАО «ММК».

В данном случае бизнес-аналитик отвечает за выявление бизнес-требований по отношению к бизнес-процессам и/или бизнес-правилам, которые будут автоматизированы в рамках реализации Решения.

Целями бизнес-анализа являются:

- Уменьшение затрат
- Поиск оптимальных путей решения бизнес проблем
- Постановка корректных временных рамок для реализации
- Повышение эффективности реализации проекта
- Разработка технической документации

На этом этапе я, совместно со специалистами управления кадров ПАО «ММК», разработали концепцию системы, провели сравнительный анализ с существующими продуктами на рынке, подготовили техническую документацию.

3.2 Этап проектирования и разработки

На этом этапе выделяются базовые сущности и взаимосвязи между ними. Создается основа для дальнейшего проектирования системы.

Методология проектирования соединяет в себе объектную декомпозицию, приемы представления физической, логической, а также динамической и статической моделей системы.

Во время проектирования разрабатываются проектные решения по выбору платформы, где будет функционировать система языка или языков реализации, назначаются требования к пользовательскому интерфейсу, определяется наиболее подходящая СУБД. Разрабатывается функциональная спецификация ПО: выбирается архитектура системы, оговариваются требования к аппаратному обеспечению, определяется набор орг. мероприятий, которые необходимы для внедрения ПО, а также перечень документов, регламентирующих его использование.

Этап разработки программного обеспечения организован в соответствии с моделями эволюционного типа жизненного цикла ПО. При разработке применяются экспериментирование и анализ, строятся прототипы, как целой системы, так и ее частей. Прототипы дают возможность глубже вникнуть в проблему и принять все необходимые проектные решения еще на ранних этапах проектирования. Такие решения могут затрагивать разные части системы: внутреннюю организацию, пользовательский интерфейс, разграничение доступа и т.д. В результате этапа реализации появляется рабочая версия продукта.

3.3 Внедрение системы и опытная эксплуатация

Внедрения системы обычно предусматривает следующие шаги:

- установка системы,
- обучение пользователей,
- опытная эксплуатация.

Опытная эксплуатация предполагает использование программного комплекса персоналом управления кадров на реальных данных с целью определения фактических значений количественных и качественных характеристик программного комплекса, выявления и устранения недостатков. Опытная эксплуатация проводится в соответствии с программой, в которой указываются продолжительность опытной эксплуатации, порядок устранения недостатков, перечень контрольных мероприятий (действий), при успешном выполнении которых программный комплекс признается соответствующим требованиям технического задания и готовым к передаче в опытно-промышленную эксплуатацию.

В рамках опытной эксплуатации, была проведено первичное тестирование прототипов представителями управления кадров, были выявлены замечания и пожелания, внесены коррективы в техническую документацию, концепцию и архитектуру системы.

3.4 Опытно-промышленная эксплуатация

Опытно-промышленная эксплуатация системы производится пилотной группой управления кадров, утвержденной в уставе проекта. Опытно-промышленная эксплуатация отличается от опытной эксплуатации тем, что систему будут использовать обученные конечные пользователи управления кадров в реальных бизнес-условиях.

Цель этапа опытно-промышленной эксплуатации – апробировать систему в производственных условия. На основе «Плана опытно-промышленной эксплуатации» в ходе этапа формируется «внутренний» продукт «Результаты опытно-промышленной эксплуатации», на основе которого будут подготовлены «внешние» продукты «Отчет об опытно промышленной эксплуатации» и «Протокол замечаний и доработок». «Опциональным внешним» продуктом этапа может стать «Презентация итогов опытно-промышленной эксплуатации».

Переход на стадию опытно-промышленной эксплуатации выполняется, когда:

- настроенная в рамках проекта система развернута на промышленной среде и используется опытной группой;
- истек срок опытной эксплуатации, который определен в календарном плане;
- исправлены ошибки с критичным и высоким приоритетом;
- актуализированы регламенты.

На этом этапе систему запустили в опытно-промышленную эксплуатацию среди некоторых обществ группы ПОА «ММК», был подготовлен и установлен информационный киоск в управлении кадров, собирались отзывы пользователей и специалистов управления кадров. Вносились коррективы в документацию, формализовались и устранялись замечания и пожелания заказчика.

3.5 Промышленная эксплуатация

Центральное событие этапа ввода в промышленную эксплуатацию – начало использования системы в промышленном режиме. Для этого необходимо устранить замечания, сформулированные на этапе опытно-промышленной эксплуатации. На основе «Протокола замечаний и доработок» разрабатывается «внешний» продукт и «Презентация итогов проекта».

Промышленная эксплуатация системы подразумевает переход на новый программный продукт и отказ от всех альтернативных способов работы за рамками данной системы. Этап промышленной эксплуатации системы подразумевает организацию службы технической поддержки системы.

При вводе в промышленную эксплуатацию, система была перенесена на production сервер, находящийся в DMZ, был закуплен и установлен SSL сертификат, в управлении кадров были установлены информационные стенды, освещающие систему, в отделе связи с общественностью заказана реклама системы, размещены ссылки в социальных сетях. Был реализован и внедрен брокер сообщений, система стала доступна для всех желающих.

Выводы по главе три

Проведены этапы бизнес анализа и формализации технического задания, разработки, опытной эксплуатации, опытно-промышленной эксплуатации и промышленной эксплуатации.

ЗАКЛЮЧЕНИЕ

После внедрения проекта, в системе зарегистрировано 556 пользователей из них 501 оставили отклики, трудоустроено было 80 человек. Каждый день в системе регистрируется, в среднем, 20 человек. 1500 уникальных посетителей сайта. На реализацию проекта было потрачено 5 миллионов рублей.

Цели проекта достигнуты, значительно расширена воронка подбора персонала, как следствие увеличился конкурс, что позволяет выбирать наиболее квалифицированные кадры.

На текущий момент, подготавливаются документы для реализации второго этапа проекта, который охватит еще больше бизнес-процессов, будут реализованы сервисы для выпускников, интеграция с внешними работными сайтами и т.д.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Горелов, С.В. Современные технологии программирования. Разработка Windows-приложений на языке C#. Учебник в 2 томах. Том II / С.В. Горелов.– Москва: Прометей, 2016.–378 с.

2 Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма – СпБ: Питер, 2001. — 368 с.

3 Дейт, К. Дж. Введение в системы баз данных, 8-е издание. / К. Дж. Дейт. – Москва: Издательский дом "Вильяме", 2005. — 1328 с.

4 Кайт, Т. Oracle для профессионалов / Т. Кайт. – СпБ: ООО «ДиаСофтЮП», 2003. — 672 с.

5 Купер, А. Интерфейс. Основы проектирования и взаимодействия / А. Купер, Р. Рейман.–4-е изд.–Санкт-Петербург: Питер, 2018.–720 с.

6 Лабберс, П. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений / П. Лабберс – Москва: ООО «И.Д. Вильямс», 2011. – 272с.

7 Моргунов, Е.П. Язык SQL. Базовый курс / Е.П. Моргунов – Москва: Postgres Professional, 2017. — 257 с.

8 Моргунов, Е.П. PostgreSQL. Основы языка SQL. Учебное пособие / Е.П. Моргунов.– Санкт-Петербург: БХВ-Петербург, 2018.–336 с.

9 Моисеев, А. Angular и TypeScript. Сайтостроение для профессионалов / А. Моисеев, Я. Файн. – СпБ: Питер, 2018. – 464 с.

10 Морето, С. Bootstrap в примерах / С. Морето. – Москва: ДМК Пресс, 2017. – 314 с.

11 Олифер, В. Компьютерные сети. Принципы, технологии, протоколы / В. Олифер, Н. Олифер.–5-е изд.–Санкт-Петербург: Питер, 2019.–992 с.

12 Прайс, М. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е издание / М. Прайс. – СпБ: Питер, 2018. — 640 с.

13 Фримен, А. ASP.NET Core MVC с примерами на C# для профессионалов 6-е издание / А. Фримен. – СпБ: ООО "Альфа-книга", 2017. – 992с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. ЛИСТИНГ БРОКЕРА СООБЩЕНИЙ

```
    public class BaseRepo
    {
        protected string connectionString =
"Server=127.0.0.1;port=5432;UID=postgres;PWD=password;Database=";

        public BaseRepo(string database)
        {
            connectionString += database;
        }

        protected byte[] GetBytes(object obj)
        {
            BinaryFormatter bf = new BinaryFormatter();
            using (MemoryStream ms = new MemoryStream())
            {
                bf.Serialize(ms, obj);
                return ms.ToArray();
            }
        }
    }

    public class QueueRepo : BaseRepo
    {
        NpgsqlConnection connection;
        public QueueRepo() : base("proxybase")
        {
            connection = new NpgsqlConnection(connectionString);
            connection.Open();
        }

        public async Task<string> AddRequest(CustomRequest request)
        {
            var bytes = GetBytes(request);
            long id;
            using (var cmd = new NpgsqlCommand("BEGIN; insert into public.queue (\"task\")
values (@task) RETURNING \"id\"; COMMIT;", connection))
            {
                cmd.Parameters.AddWithValue("task", bytes);
                using (var reader = cmd.ExecuteReader())
                {
                    if (reader.HasRows)
                    {
                        reader.Read();
                        id = reader.GetInt64(0);
                    }
                    else
                    {
                        throw new Exception("internal server error");
                    }
                }
            }

            string result = string.Empty;
            await Task.Run(() =>
            {
                while (true)
                {
                    using (var cmd = new NpgsqlCommand("select a.json from public.results a
where a.crt_date > current_date and a.id=@id", connection))
```

```
                    {
                        cmd.Parameters.AddWithValue("id", id);
                        using (var reader = cmd.ExecuteReader())
                        {
                            if (reader.HasRows)
                            {
                                reader.Read();
                                result = reader.GetString(0);
                                break;
                            }
                        }
                    }
                    Thread.Sleep(1);
                }
            });
            return result;
        }

        public CustomRequest GetTask(out long id)
        {
            CustomRequest result;
            while (true)
            {
                using (var cmd = new NpgsqlCommand("BEGIN; UPDATE public.queue SET
\"state\"=@state, \"upd_date\"=current_timestamp FROM (SELECT \"id\", \"task\" FROM
public.queue a where \"crt_date\" > current_date and \"state\" is null order by \"crt_date\"
limit 1 for update) AS subquery WHERE public.queue.id=subquery.id RETURNING subquery.id,
subquery.task; COMMIT;", connection))
                {
                    cmd.Parameters.AddWithValue("state", "in work");
                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.HasRows)
                        {
                            reader.Read();
                            id = reader.GetInt64(0);
                            var bytes = (byte[])reader[1];
                            using (MemoryStream memStream = new MemoryStream())
                            {
                                BinaryFormatter binForm = new BinaryFormatter();
                                memStream.Write(bytes, 0, bytes.Length);
                                memStream.Seek(0, SeekOrigin.Begin);
                                result = (CustomRequest)binForm.Deserialize(memStream);
                            }
                            break;
                        }
                    }
                }
            }
            return result;
        }

        public void SetResult(long id, string json)
        {
            using (var cmd = new NpgsqlCommand("BEGIN; insert into public.results (\"id\",
\"json\") values(@id, @json); update public.queue a set \"state\"=@state,
\"upd_date\"=current_timestamp where \"crt_date\" > current_date and a.id=@id; COMMIT;",
connection))
            {
                cmd.Parameters.AddWithValue("id", id);
                cmd.Parameters.AddWithValue("json", json);
                cmd.Parameters.AddWithValue("state", "done");
                cmd.ExecuteNonQuery();
            }
```

```
        }

        public void SetError(long id, string error)
        {
            using (var cmd = new NpgsqlCommand("BEGIN; insert into public.results (\"id\",
\"json\") values(@id, @json); insert into public.errors (\"task_id\",\"error\") values(@id,
@error); update public.queue a set \"state\"=@state, \"upd_date\"=current_timestamp where
\"crt_date\" > current_date and \"id\"=@id; COMMIT;", connection))
            {
                cmd.Parameters.AddWithValue("id", id);
                cmd.Parameters.AddWithValue("json", "error");
                cmd.Parameters.AddWithValue("error", error);
                cmd.Parameters.AddWithValue("state", "error");
                cmd.ExecuteNonQuery();
            }
        }

        public void Abort()
        {
            connection.Close();
            connection.Dispose();
            connection = null;
        }
    }

    public class ProxyController : ApiController
    {
        NLog.Logger logger = NLog.LogManager.GetCurrentClassLogger();

        [HttpPost]
        [Route("api/Proxy")]
        public async Task<HttpResponseMessage> Proxy()
        {
            var request = ReadContext();
            var result = await Task.Run(async () =>
            {
                QueueRepo repo = new QueueRepo();
                var response = await repo.AddRequest(request);
                repo.Abort();
                return response;
            });
            if (string.Compare(result, "error") == 0)
            {
                return Request.CreateResponse(HttpStatusCode.InternalServerError, "Внутренняя
ошибка сервера");
            }
            return Request.CreateResponse(HttpStatusCode.OK,
JsonConvert.DeserializeObject<object>(result));
        }

        private string GetInnerException(Exception ex)
        {
            var result = string.Empty;
            result = $"Message: {ex.Message}\r\n\r\nStackTrace:{ex.StackTrace}";
            if (ex.InnerException != null)
            {
                result +=
"**********InnerException**********\r\n\r\n";
                result += GetInnerException(ex.InnerException);
            }
            return result;
        }
```

```
        private CustomRequest ReadContext()
        {
            var form = HttpContext.Current.Request.Form;
            var URL = form["URL"];
            var rawType = form["Type"];
            var rawHeaders = form["Headers"];
            var rawBody = form["Body"];
            logger.Debug($"URL: {URL}");
            logger.Debug($"Type: {rawType}");
            logger.Debug($"Headers: {rawHeaders}");
            logger.Debug($"Body: {rawBody}");
            var headers = new Dictionary<string, string>();
            var body = new Dictionary<string, string>();
            Enum.TryParse<RequestType>(rawType, out RequestType type);

            if (!string.IsNullOrEmpty(rawHeaders))
            {
                headers = JsonConvert.DeserializeObject<Dictionary<string,
string>>(rawHeaders);
            }

            if (!string.IsNullOrEmpty(rawBody))
            {
                body = JsonConvert.DeserializeObject<Dictionary<string, string>>(rawBody);
            }

            switch (type)
            {
                case RequestType.GET:
                    return new CustomRequest(URL, type);
                case RequestType.POST:
                    var files = HttpContext.Current.Request.Files.Count > 0 ?
HttpContext.Current.Request.Files : null;
                    if (files != null)
                    {
                        List<Tuple<string, byte[]>> fileList = new List<Tuple<string,
byte[]>>();
                        for (int i = 0; i < files.Count; i++)
                        {
                            var file = files.Get(i);
                            logger.Debug($"Request have file: type: {file.ContentType};
length: {file.ContentLength}; name: {file.FileName}");
                            fileList.Add(new Tuple<string, byte[]>(file.FileName,
GetBytes(file.InputStream)));
                        }
                        return new CustomRequest(URL, type, headers.Count > 0 ? headers :
null, body.Count > 0 ? body : null, fileList);
                    }
                    else
                    {
                        return new CustomRequest(URL, type, headers.Count > 0 ? headers :
null, body.Count > 0 ? body : null, null);
                    }
                default:
                    return null;
            }
        }

        private byte[] GetBytes(Stream input)
        {
            byte[] buffer = new byte[16 * 1024];
            using (MemoryStream ms = new MemoryStream())
            {
                int read;
```

```
                while ((read = input.Read(buffer, 0, buffer.Length)) > 0)
            {
                ms.Write(buffer, 0, read);
            }
            return ms.ToArray();
        }
    }
}
```