Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Средства и технологии анализа и разработки информационных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ОНЛАЙН-СИСТЕМА ПО ПОИСКУ РАБОТЫ

БГУИР КП 1-40 01 02-02 301 ПЗ 101

Студент: гр. 572301 Абметка В. В.

Руководитель: Хомяков П. В.

Минск 2018

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc512798599)

[1 Описание процесса поиска работы 7](#_Toc512798600)

[2 Описание основных процессов поиска работы 11](#_Toc512798601)

[3 Спецификация вариантов использования системы поиска работы 17](#_Toc512798602)

[4 Информационная модель системы по поиску работы и ее описание 23](#_Toc512798603)

[5 Обоснование выбора компонентов и технологий 26](#_Toc512798604)

[5.1 Диаграмма последовательностей (sequence diagram) 29](#_Toc512798605)

[5.2 Диаграмма компонентов 31](#_Toc512798606)

[5.3 Диаграмма развертывания 32](#_Toc512798607)

[6 Модели представления системы и их описание 33](#_Toc512798608)

[6.1 Диаграмма состояний 33](#_Toc512798609)

[6.2 Описание алгоритма авторизации пользователя 34](#_Toc512798610)

[6.3 Описание алгоритма отклика кандидатом на вакансию 35](#_Toc512798611)

[7 Описание применения паттернов проектирования 38](#_Toc512798612)

[7.1 Паттерн «Одиночка» (Singleton) 38](#_Toc512798613)

[7.2 Паттерн «Команда» (Command) 39](#_Toc512798614)

[7.3 Паттерн «Заместитель» (Proxy) 40](#_Toc512798615)

[8 Руководство по развертыванию системы 42](#_Toc512798616)

[9 Результаты тестирования системы 45](#_Toc512798617)

[Заключение 62](#_Toc512798618)

[Список использованных источников 64](#_Toc512798619)

[Приложение А (Обязательное) Диаграммы классов 65](#_Toc512798620)

[Приложение Б (Обязательное) Листинг основных элементов программы 81](#_Toc512798623)

[Приложение В (Обязательное) Листинг скрипта генерации базы данных 90](#_Toc512798626)

# **ВВЕДЕНИЕ**

В условиях рыночной экономики качество персонала стало главнейшим фактором, определяющим выживание и экономическое положение организаций. Отбор работников всегда старались проводить достаточно тщательно, поскольку качество людских ресурсов во многом определяет возможности и эффективность последующего использования. В настоящее время работодатели перешли к активным методам поиска и вербовки персонала; стремятся привлечь в организацию как можно больше соискателей, удовлетворяющих требованиям, совершенствуется процедура самого отбора.

Отбор новых работников не только призван обеспечить режим нормального функционирования, но и закладывает фундамент будущего успеха организации. Актуальность исследования поиска и отбора персонала как функции управления персоналом на современном этапе заключается в том, что с переходом к рыночной экономике стало невозможным использовать старые методы поиска и отбора персонала, которые использовались в административной системе.

Управление персоналом состоит в подборе и сохранении требуемого организации персонала, его профессиональном обучении и развитии, оценке деятельности каждого из работников с точки зрения реализации целей организации, дающей возможность скорректировать его поведение, вознаграждении персонала за его усилия.

Работники кадровых служб давно ощущали потребность в более обоснованных и надежных процедурах. Повышение эффективности и надежности отбора связывается с последовательным проведением проверки деловых и личностных качеств кандидата, основанной на взаимодополняющих методах их выявления и источниках информации. Осуществляется поэтапный отбор кандидатур. Каждый раз отсеивают тех, кто обнаружил явное несоответствие предъявляемым требованиям. Одновременно применяют, по возможности, объективную оценку фактических знаний и степени; владения кандидатом необходимыми производственными навыками. Таким образом, формируется сложная многоступенчатая система проведения отбора человеческих ресурсов [1].

Целью данной курсовой работы является оптимизация поиска работы для соискателей и автоматизация процесса поиска наиболее подходящих кандидатов на должности для сотрудников HR, затрачивая минимальное количество средств, что позволит решить большинство проблем, связанных с ускорением бизнес-процессов фирм, предприятий и компаний.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить предметную область HR-сферы;
* выполнить функциональное моделирование процессов поиска работы;
* разработать модели представления системы на основе UML;
* определить основные характеристики и принципы поиска работы соискателем и поиска подходящих кандидатов на должности сотрудниками HR;
* создать логическую и физическую модель представления данных;
* разработать базу данных;
* реализовать клиент-серверное взаимодействие, что позволит управлять базой данных и выполнять определенный ряд функций только на серверной части;
* разработать простой и удобный интерфейс приложения;
* протестировать программное приложение.

# **1 ОПИСАНИЕ ПРОЦЕССА ПОИСКА РАБОТЫ**

Основное значение для получения престижной и высокооплачиваемой работы имеют образование и профессиональный опыт соискателя. Однако, при прочих равных условиях, быстрее и с меньшими затратами подберет подходящую вакансию человек, грамотно построивший стратегию поиска работы. Очевидно, что она будет разной в зависимости от образования, специальности, специализации, квалификации соискателя.

К распространенным способам поиска работы относят:

1. Поиск через знакомых и родственников.
2. Поиск через объявления о вакансиях в газетах.
3. Размещение своего мини-резюме в периодических специализированных изданиях.
4. Рассылка резюме потенциальным работодателям.
5. Поиск через сеть Internet.
6. Прямое инициативное обращение к потенциальному работодателю.
7. Поиск через кадровые агентства.
8. Поиск через государственные центры занятости и проводимые ими ярмарки вакансий.
9. Поиск через структуры вузов, занимающиеся вопросами трудоустройства студентов и выпускников и проводимые ими ярмарки вакансий.

Поиск через знакомых особенно хорош для тех, кто имеет широкую сеть профессиональных и личных контактов. Здесь главное снабдить своим резюме тех, кто согласился вам помогать. Знакомые будут располагать конкретной информацией для продвижения вашей кандидатуры к потенциальному работодателю.

Чтобы найти подходящую работу, можно покупать газеты с объявлениями о работе, внимательно выбирать более или менее подходящие вакансии и высылать свое резюме. Сегодня информацию о свободных рабочих местах публикуют практически все газеты региона. Есть и специализированные издания.

Способ размещения резюме в периодических изданиях пользуется широким спросом. Конечно, размещать свое резюме нужно не в любой газете или журнале, а в специализированном источнике информации. Разместить сове резюме возможно на сайтах кадровых агентств. Причем, в отличие от первого варианта, за размещение резюме на сайте агентства плата не взимается и затрачивается минимальное время.

Однако, разместив свое резюме в газете или на сайте, останавливаться на этом не надо. Сегодня количество людей, ищущих работу столь велико, что кадровики, зачастую, просто не успевают просмотреть все резюме.

Если рассматривать поиск через сеть Internet, то здесь есть ряд распространенных сайтов, публикующих вакансии и мини-резюме. В последнее время все больше специалистов находят себе работу именно через Internet. Кроме того, сегодня многие организации имеют свои web-серверы, на которых выставляют информацию о свободных рабочих местах, а также адрес электронной почты для получения резюме специалиста. Некоторые организации предусматривают возможность соискателям заполнить анкету в электронном варианте. Информация, содержащаяся в анкете, автоматически распределяется в электронной базе данных организации. Корпоративные сайты могут быть интересны соискателям, заинтересованным в получении работы в конкретной фирме или ряде фирм. Существуют интернет ресурсы, посвященные непосредственно молодым специалистам. Данный ресурс охватывает спектр вопросов в области трудоустройства и профессионального роста, возникающих у студентов и выпускников. На сайтах можно найти:

- новости и события в области профессионального роста и поиска работы;

- программы для выпускников, вакансии, стажировки в Беларуси и за рубежом;

- информацию о компаниях-работодателях, ориентированных на привлечение молодых специалистов;

- юридические аспекты трудовой деятельности и поиска работы;

- статьи, исследования и иные аналитические материалы по вопросам Graduate Recruitment.

Как правило, резюме рассылаются по адресам предприятий, вакансии которых были предварительно отобраны. Однако можно пойти и другим путем. Если вы хотите найти работу по определенной специальности в конкретной фирме – можно выслать свое резюме адресату. Даже если Вы сразу не получите работу, велика вероятность того, что Вас возьмут на заметку и при открытии подходящей вакансии пригласят на собеседование. Ротация кадров неизбежна. Вопрос только в сроке ожидания.

Подобную тактику могут применять именно молодые специалисты, не имеющие опыта работы, но рассчитывающие устроиться в солидную компанию. Единственная проблема в том, что по статистике на сотню разосланных подобным образом резюме приходит 2 – 3 ответа. Таким образом, если Вы выбрали тактику адресной рассылки, то предприятий – адресатов должно быть много. Однако не претендуйте на несколько вакансий одновременно. Ваше резюме должно быть направленно на одну вакансию. В противном случае, работодатель может усомниться в Вашем профессионализме.

Поиск через государственные центры занятости и проводимые ими ярмарки вакансий - свободных рабочих мест здесь всегда много. Однако в большинстве случаев это рабочие специальности, либо вакансии с низкой заработной платой. Для молодого специалиста более продуктивный способ поиска работы через ярмарки вакансий, проводимые непосредственно в вузе.

Рассматривая поиск через кадровые агентства следует понимать, что поиск работы - процесс вероятностный. Необходимо, чтобы была вакансия, по требованиям которой вы подходили. В кадровые агентства, чаще всего, поступают заявки от престижных организаций. Этим объясняется высокий уровень требований к соискателю. Агентство после изучения требований работодателя осуществляет поиск и подбор кандидатов, где на первое предложение рекомендуется три наиболее подходящих претендента с опытом работы.

Как показывает практика, реальный шанс найти работу через кадровые агентства имеют лишь действительно квалифицированные и опытные работники. Остальные, в том числе молодые специалисты, как правило, включаются в базу данных агентства [2].

# **2 ОПИСАНИЕ ОСНОВНЫХ ПРОЦЕССОВ ПОИСКА РАБОТЫ**

Для получения более полного представления о том, как можно автоматизировать систему поиска работы с помощью программного продукта, была создана функциональная модель.

Для этого был выбран стандарт IDEF0. IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

На контекстной диаграмме верхнего уровня (рисунок 2.1) представлена функциональная модель «Поиск работы», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными.

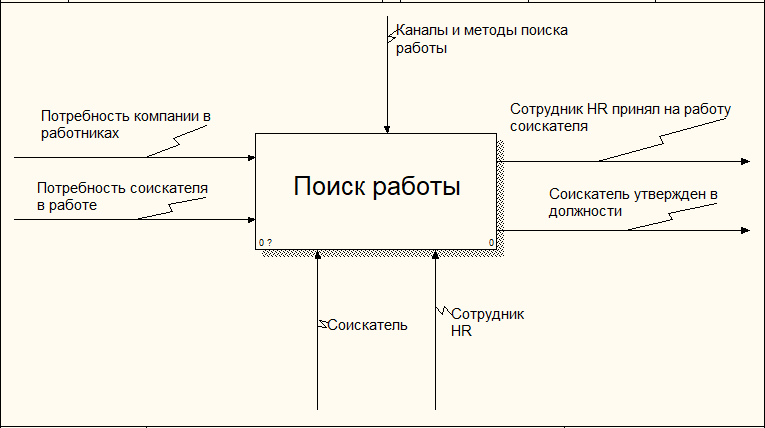


Рисунок 2.1 – Диаграмма IDEF0. Верхний уровень «Поиск работы

Входной поток включает в себя потребность компании в работниках и потребность соискателя в работе. После соответствующей обработки в выводном потоке имеем сотрудника HR, принявшего на работу соискателя и соискателя, утвержденного в должности. В роли управляющего механизма выступает каналы и методы поиска работы. Механизмами являются соискатель и сотрудник HR.

На рисунке 2.2 отображена декомпозиция контекстной диаграммы, состоящая из четырех блоков: «Публикация соискателем резюме», «Нахождение сотрудником HR потенциального кандидата», «Прохождение этапа собеседования» и «Оформление сотрудника на работу».

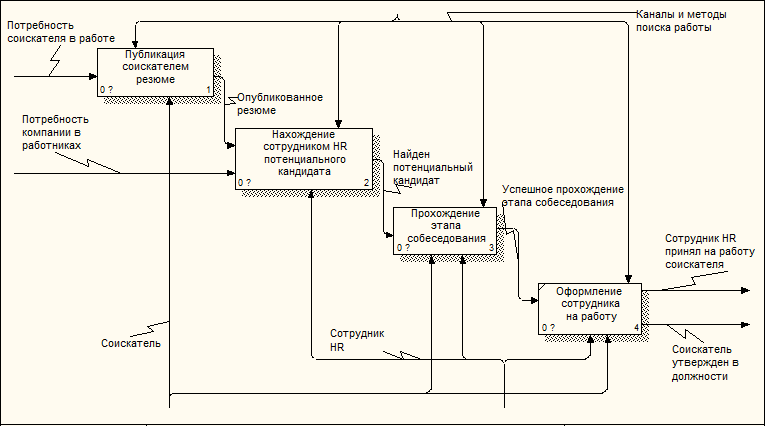


Рисунок 2.2 – Диаграмма IDEF0. Уровень 1 – Поиск работы

Первый компонент данной декомпозиции («Публикация соискателем резюме») подразумевает, что соискатель определился с желаемой должностью, составил и опубликовал корректно составленное резюме, отображающее его личностные и профессиональные качества, которые могут быть полезны на предприятии, а также облегчить процедуру отбора кандидатов для сотрудников HR.

Второй компонент декомпозиции контекстной диаграммы («Нахождение сотрудником HR потенциального кандидата») означает, что в свое время поступил приказ об открытии вакансии, на которую нужны работники и сотрудник HR составил и опубликовал вакансию, на которую уже в последствии могли откликаться соискатели, либо сотрудник компании сам нашел подходящие резюме на данную должность и связался с соискателями.

Третий компонент данной декомпозиции («Прохождение этапа собеседования») подразумевает, что сотрудник HR воспринял соискателя как потенциально подходящего на данную должность, связался и назначил собеседование с ним.

Четвертый компонент данной декомпозиции («Оформление сотрудника на работу») подразумевает, что соискатель успешно прошел все этапы собеседования, устроил сотрудников HR и компетентных лиц компании и готов приступить к работе на данном предприятии.

Декомпозиция блока «Публикация соискателем резюме» отображена на рисунке 2.3. Она представлена четырьмя компонентами: «Определение желаемой должности», «Выбор каналов поиска работы», «Составление резюме» и «Публикация резюме».

Декомпозиция блока «Нахождение сотрудником HR потенциального кандидата» отображена на рисунке 2.4. Она представлена четырьмя компонентами: «Публикация вакансии», «Замечен отклик соискателя на вакансию либо найден подходящий кандидат», «Ознакомление сотрудника HR с резюме соискателя» и «Связь с соискателем».

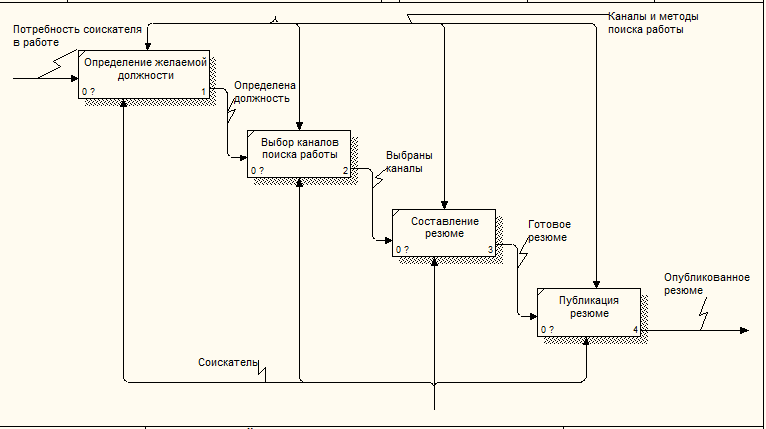


Рисунок 2.3 – Диаграмма IDEF0. Уровень 2 – Публикация соискателем резюме

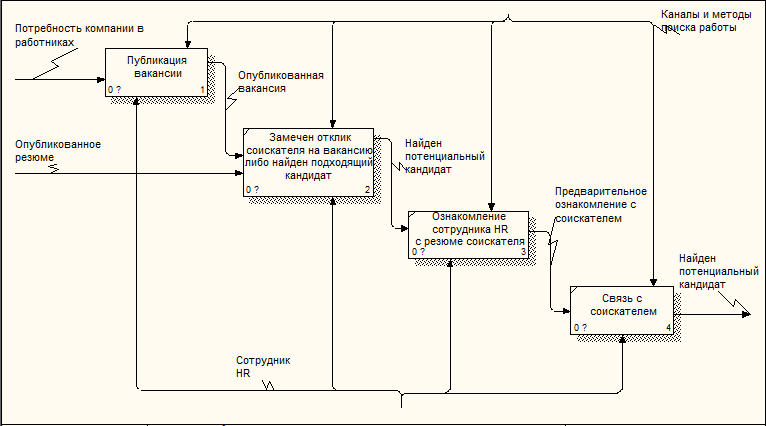


Рисунок 2.4 – Диаграмма IDEF0. Уровень 2 – Нахождение сотрудником HR потенциального кандидата

На рисунке 2.5 отображена декомпозиция блока «Публикация вакансии», состоящая из четырех блоков: «Поступление информации о необходимости открытия вакансии», «Сбор и анализ данных вакансии», «Составление и упорядочивание информации о вакансии» и «Публикация в различные каналы поиска работы».

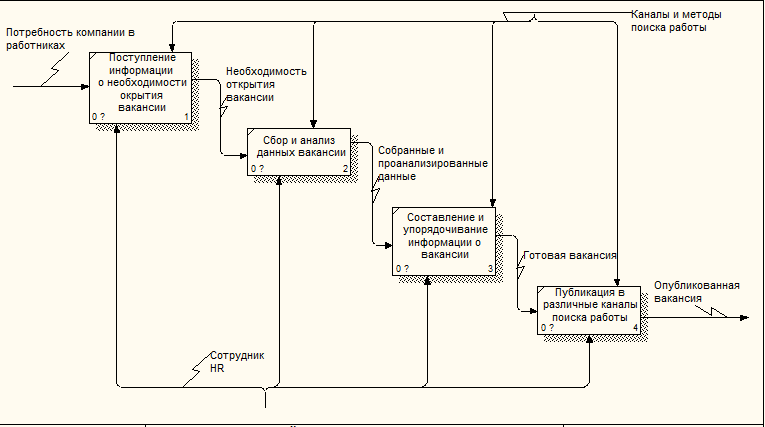


Рисунок 2.5 – Диаграмма IDEF0. Уровень 3 – Публикация вакансии

Соответственно каждый блок на выходе предоставляет необходимость открытия вакансии, собранные и проанализированные данные, готовая и опубликованная вакансия, имея при этом каналы и методы поиска работы в роли управляющего механизма, а также соискателя и сотрудника HR в роли механизмов.

Декомпозиция блока «Прохождение этапа собеседования» отображена на рисунке 2.6. Она представлена четырьмя компонентами: «Прохождение телефонного интервью», «Прохождение собеседования на знание английского языка», «Прохождение очного собеседования» и «Прохождение технического интервью».

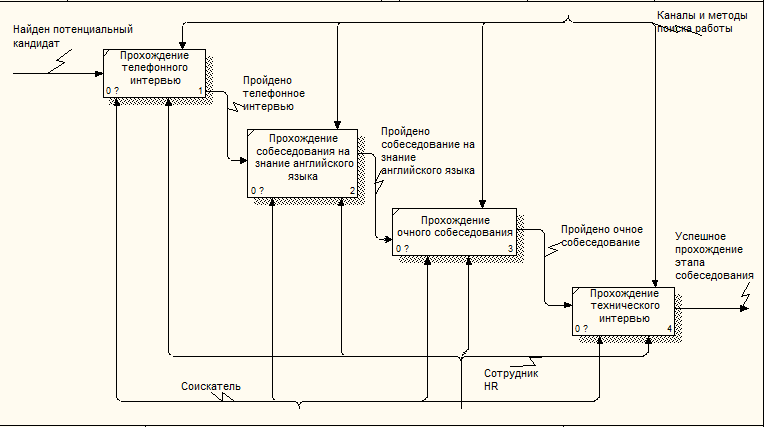


Рисунок 2.6 – Диаграмма IDEF0. Уровень 2 – Прохождение этапа собеседования

Смоделировав бизнес-процессы можно начинать проектировать архитектуру разрабатываемого проекта по автоматизации бизнес-процессов.

Разработка диаграмм стандарта IDEF0 является одним из этапов при

проектировании. Таким образом, можно сделать вывод, что моделирования

бизнес-процессов играет важную роль в разработке информационных систем.

# **3 СПЕЦИФИКАЦИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ СИСТЕМЫ ПОИСКА РАБОТЫ**

Диаграмма вариантов использования состоит из актеров, которым система предоставляет возможность выбора действия, и вариантов использования описывающих возможности системы.

В данной диаграмме вариантов использования в роли актёров выступают администратор (рисунок 3.1), соискатель (рисунок 3.2) и сотрудник HR (рисунок 3.3).

1. Администратор – главный управляющий системой. Данный актер наделен следующими функциями: авторизация (прецедент «Авторизация»); возможность выбора языка интерфейса между русским и английским (прецедент «Выбрать используемый языка интерфейса»); просмотр информации о компании или людях, которые разрабатывали данный программный продукт (прецедент «Просмотреть информацию о разработчике») и просмотр различной информации, хранящейся в системе (прецедент «Просмотреть информацию»).

От прецедента «Просмотреть информацию» установлены связи включения к прецедентам «Просмотреть информацию о соискателях», «Просмотреть информацию о сотрудниках HR», «Просмотреть информацию о вакансиях» и «Просмотреть информацию об интервью».

В свою очередь, прецеденты «Удалить информацию о соискателях» и «Редактировать информацию о соискателях» обобщаются до прецедента «Просмотреть информацию о соискателях», прецеденты «Удалить информацию о сотрудниках HR» и «Редактировать информацию о сотрудниках HR» обобщаются до прецедента «Просмотреть информацию о сотрудниках HR», прецеденты «Удалить информацию о вакансиях» и «Редактировать информацию о вакансиях» обобщаются до прецедента «Просмотреть информацию о вакансиях» и, соответственно, прецеденты «Удалить информацию об интервью» и «Редактировать информацию об интервью» обобщаются до прецедента «Просмотреть информацию об интервью».

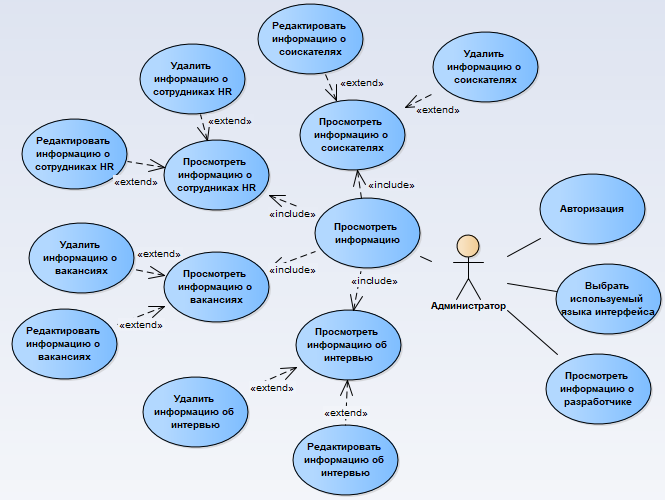


Рисунок 3.1 – Диаграмма вариантов использования онлайн-системы «Поиск работы». Актер «Администратор»

1. Соискатель – это физическое или юридическое лицо, рядовой пользователь интернет-пространства, который заинтересован в поиске и устройстве на работу. Диаграмма вариантов использования для данной роли представлена на рисунке 3.2.

Данный актер наделен следующими функциями: авторизация (прецедент «Авторизация»); регистрация (прецедент «Регистрация»); возможность выбора языка интерфейса между русским и английским (прецедент «Выбрать используемый языка интерфейса»); просмотр информации о компании или людях, которые разрабатывали данный программный продукт (прецедент «Просмотреть информацию о разработчике»); возможность изменения логина и пароля, которые соискатель указал при регистрации в системе (прецедент «Изменить данные аккаунта»); возможность окончательно и безвозвратно удалить свой имеющийся, текущий аккаунт системы (прецедент «Удалить аккаунт»); просмотр информации о вакансиях, хранящейся в системе (прецедент «Просмотреть вакансии») и возможность добавить свое резюме (прецедент «Добавить резюме»).

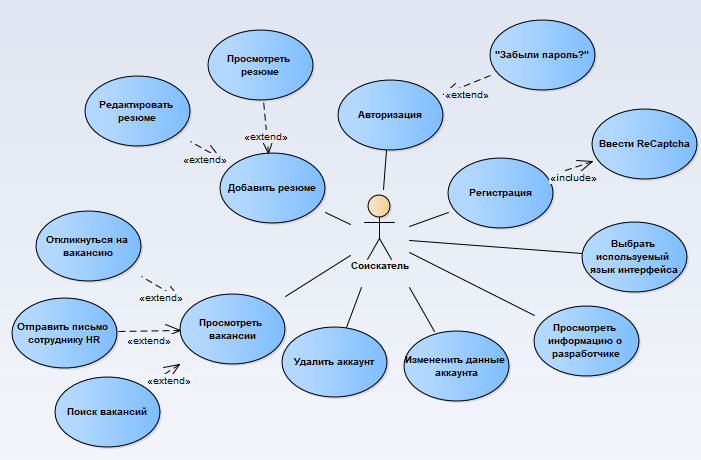


Рисунок 3.2 – Диаграмма вариантов использования онлайн-системы «Поиск работы». Актер «Соискатель»

От прецедента «Регистрация» установлена связь включения к прецеденту «Ввести ReCaptcha». Это так называемое поле подтверждения, что вы не робот. Не поставив галочку в данном поле, регистрация окончится неудачей.

В свою очередь, прецедент «Забыли пароль?» обобщается до прецедента «Авторизация», прецеденты «Просмотреть резюме» и «Редактировать резюме» обобщаются до прецедента «Добавить резюме», прецеденты «Откликнуться на вакансию», «Отправить письмо сотруднику HR» и «Поиск вакансий» обобщаются до прецедента «Просмотреть вакансии».

1. Сотрудник HR (Human Resource) – специалист, менеджер по работе с человеческими ресурсами, качественным управлением персонала, расстановке приоритетов, развитию сотрудников, прорисовке целей для них. Диаграмма вариантов использования для данной роли представлена на рисунке 3.3.

Данный актер наделен следующими функциями: авторизация (прецедент «Авторизация»); регистрация (прецедент «Регистрация»); возможность выбора языка интерфейса между русским и английским (прецедент «Выбрать используемый языка интерфейса»); просмотр информации о компании или людях, которые разрабатывали данный программный продукт (прецедент «Просмотреть информацию о разработчике»); возможность изменения логина и пароля, которые соискатель указал при регистрации в системе (прецедент «Изменить данные аккаунта»); возможность окончательно и безвозвратно удалить свой имеющийся, текущий аккаунт системы (прецедент «Удалить аккаунт»); просмотр информации обо всех интервью, хранящихся в системе (прецедент «Просмотреть интервью») и возможность добавить различного рода информацию (прецедент «Добавить информацию»).

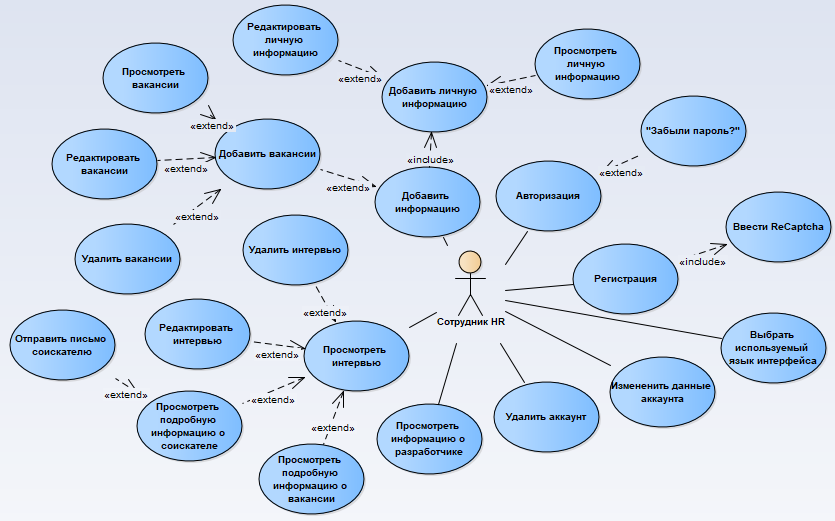


Рисунок 3.3 – Диаграмма вариантов использования онлайн-системы «Поиск работы». Актер «Сотрудник HR»

От прецедента «Регистрация» установлена связь включения к прецеденту «Ввести ReCaptcha». Это так называемое поле подтверждения, что вы не робот. Не поставив галочку в данном поле, регистрация окончится неудачей.

В свою очередь, прецедент «Забыли пароль?» обобщается до прецедента «Авторизация», прецеденты «Удалить интервью», «Редактировать интервью», «Просмотреть подробную информацию о соискателе» и «Просмотреть подробную информацию о вакансии» обобщаются до прецедента «Просмотреть интервью». Кроме того, прецедент «Отправить письмо соискателю» обобщается до прецедента «Просмотреть подробную информацию о соискателе».

Также от прецедента «Добавить информацию» установлена связь включения к прецеденту «Добавить личную информацию», который обобщает прецеденты «Просмотреть личную информацию» и «Редактировать личную информацию».

Независимо от этого, прецедент «Добавить вакансии» обобщается до прецедента «Добавить информацию». А прецеденты «Просмотреть вакансии», «Редактировать вакансии» и «Удалить вакансии», в свою очередь, обобщаются до прецедента «Добавить вакансии».

# **4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ ПО ПОИСКУ РАБОТЫ И ЕЕ ОПИСАНИЕ**

В результате информационного моделирования были выделены следующие сущности:

* информация об аккаунте соискателя и сотрудника HR;
* информация о конкретном соискателе;
* информация о конкретном сотруднике HR;
* информация о вакансиях;
* информация о собеседованиях.

В первой выше представленной сущности системы «информация об аккаунте соискателя и сотрудника HR» содержатся синтетический идентификационный номер аккаунта, данные о логине, пароле и принадлежности аккаунта к соискателю либо к сотруднику HR.

Сведения о личностных и профессиональных качествах соискателя должны вноситься в сущность «информация о конкретном соискателе». Здесь содержится синтетический идентификационный номер соискателя, информация о фамилии, имени, отчестве соискателя, его возраст, адрес электронной почты, гражданство, номер телефона, желаемая должность, образование, опыт работы, уровень английского языка, его профессиональные навыки, а также идентификационный номер аккаунта. Связь с таблицей «информация об аккаунте соискателя и сотрудника HR» - один-ко-многим.

Для хранения личной информации о сотрудниках HR нужно использовать соответствующую сущность «информация о конкретном сотруднике HR». Здесь содержится синтетический идентификационный номер сотрудника HR, информация о фамилии, имени, отчестве, также адрес проживания, номер телефона, адрес электронной почты, название компании, которую представляет сотрудник, и идентификационный номер аккаунта. Связь с таблицей «информация об аккаунте соискателя и сотрудника HR» - один-ко-многим.

В четвертой выше представленной сущности системы «информация о вакансиях» содержатся синтетический идентификационный номер вакансии, должность, на которую идет набор, компания, которой принадлежит данная открытая вакансия, приблизительный уровень заработной платы, которую будет получать работник на этой должности, местоположение, требуемый опыт работы, необходимый уровень английского языка, основная, более подробная текстовая информация о вакансии, состояние вакансии (открыта либо закрыта) и идентификационный номер сотрудника HR, который курирует данную вакансию. Связь с таблицей «информация о конкретном сотруднике HR» - один-ко-многим.

Также имеется таблица «информация о собеседованиях», куда заносится информация о состоянии продвижения соискателя на конкретной вакансии по этапу собеседований. Здесь содержится синтетический идентификационный номер собеседования, идентификационный номер соискателя, идентификационный номер вакансии, состояние предварительного собеседования, состояние финального технического собеседования. Связи с таблицей «информация о вакансиях» - один-ко-многим, с таблицей «информация о конкретном соискателе» - один-ко-многим.

Данная модель системы приведена к третьей нормальной форме:

1. Все атрибуты являются простыми, все используемые домены содержат только скалярные значения, вероятность повторения строк в таблице отсутствует (первая нормальная форма).
2. Каждый не ключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа (вторая нормальная форма).
3. Отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых (третья нормальная форма) [3].

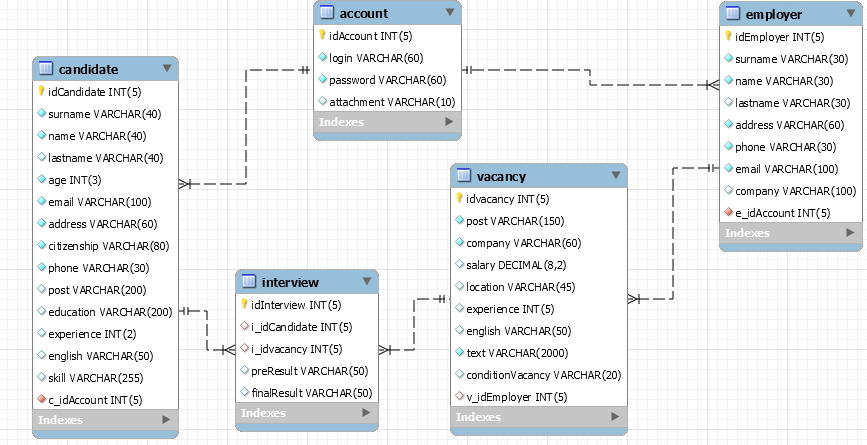


Рисунок 4.1 – Информационная модель базы данных онлайн-системы по поиску работы

# **5 ОБОСНОВАНИЕ ВЫБОРА КОМПОНЕНТОВ И ТЕХНОЛОГИЙ**

Задачей курсового проекта является создание клиент-серверного программного приложения (онлайн-системы), позволяющего оптимизировать процесс поиска работы.

Программный код должен быть написан на объектно-ориентированном языке Java.

Объектно-ориентированный язык Java, разработанный в Sun Microsystems, предназначен для создания переносимых на различные платформы и операционные системы программ. Язык Java нашел широкое применение в Интернет-приложениях, добавив на статические и клиентские Web-страницы динамическую графику, улучшив интерфейсы и реализовав вычислительные возможности. Но объектно-ориентированная парадигма и кроссплатформенность привели к тому, что уже буквально через несколько лет после своего создания язык практически покинул клиентские страницы и перебрался на сервера. На стороне клиента его место занял язык JavaScript.

При создании язык Java предполагался более простым, чем его синтаксический предок С++. На сегодняшний день с появлением версий J2SE 9 и J2EE 8 возможности языка Java существенно расширились и во многом перекрывают функциональность С/С++/С#. Отсутствие указателей (наиболее опасного средства языка С++) нельзя считать сужением возможностей, а тем более недостатком, это просто требование безопасности. Возможность работы с произвольными адресами памяти через бестиповые указатели позволяет игнорировать защиту памяти. Отсутствие в Java множественного наследования легко заменяется на более понятные конструкции с применением, например, интерфейсов.

Системная библиотека классов языка Java содержит классы и пакеты, реализующие и расширяющие базовые возможности языка, а также сетевые возможности, взаимодействие с базами данных, графические интерфейсы и многое другое. Методы классов, включенных в эти библиотеки, вызываются из JVM (Java Virtual Machine) во время интерпретации программы [4].

Язык Java делает сетевое программирование простым благодаря наличию специальных средств и классов. Большинство этих классов находится в пакете java.net. Сетевые классы имеют методы для установки сетевых соединений передачи запросов и сообщений. Многопоточность позволяет обрабатывать несколько соединений. Сетевые приложения используют Internet-приложения, к которым относятся Web-браузер, e-mail, сетевые новости, передача файлов. Для создания таких приложений используются сокеты, порты, протоколы TCP/IP, UDP.

Приложения клиент/сервер используют компьютер, выполняющий специальную программу-сервер, которая обычно устанавливается на удаленном компьютере и предоставляет услуги другим программам-клиентам. Клиент − это программа, получающая услуги от сервера. Клиент устанавливает соединение с сервером и пересылает серверу запрос. Сервер осуществляет прослушивание клиентов, получает и выполняет запрос после установки соединения. Результат выполнения запроса может быть возвращен сервером клиенту. Запросы и сообщения представляют собой записи, структура которых определяется используемыми протоколами.

В качестве основных фреймворков были выбраны Spring и Hibernate.

Hibernate – самая популярная реализация спецификации JPA, предназначенная для решения задач объектно-реляционного отображения (ORM). Она является хорошей альтернативой Entity Beans, что, вероятно, послужило одной из причин того, что EJB 3 имеет теперь много методов из Hibernate (JDO и Toplink). Его целью является освобождение разработчика от значительного объёма сравнительно низкоуровневого программирования при работе в объектно-ориентированных средствах в реляционной базе данных. Hibernate можно использовать как в процессе проектирования системы классов и таблиц «с нуля», так и для работы с уже существующей базой данных.

Библиотека не только решает задачу связи классов Java с таблицами базы данных (и типов данных Java с типами данных SQL), но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC-кода. Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки результирующего набора данных и преобразования объектов, максимально облегчая перенос (портирование) приложения на любые базы данных SQL.

Hibernate обеспечивает прозрачную поддержку сохранности данных (persistence) для «POJO» (то есть для стандартных Java-объектов); единственное строгое требование для сохраняемого класса — наличие конструктора по умолчанию (без параметров). Для корректного поведения в некоторых приложениях требуется также уделить внимание методам equals() и hashCode().

Среда Spring Framework, содержащая большое количество классов и пакетов, была разработана как модульная среда, которая может быть поэтапно или частично введена в проект, т.е. использованы будут только необходимые средства (например, среда выполнения Web). Среда Spring дополняет систему Java/JEE, предоставляя контейнер инверсии управления (Inversion of Control - IoC), среду выполнения Web, слой абстракции управления транзакциями, вспомогательные классы JDBC, API планирования задач, возможности электронной почты и многое другое.

Среда Spring является лидером в области контейнеров IоС; однако и ее среда выполнения Web также удивительно популярна. Spring была выбрана в качестве среды выполнения Web потому, что необходимы также и другие ее возможности, такие как IоС, управление транзакциями, электронная почта, планирование и т.д. Среда выполнения Web Spring MVC не имеет аналогов в области отказоустойчивости и гибкости.

В качестве основной среды разработки в проекте используется IntelliJ IDEA 2017.1.2. В качестве СУБД в курсовом проекте используется MySQL. Для создания UML-диаграмм в работе было использовано средство Enterprise Architect, которое позволяет осуществлять создание диаграмм вариантов использования, состояний, последовательность с помощью просто моделирования. Для выполнения UML-моделей в стандарте IDEF0 использовалось CASE-средство CA AllFusion Process Modeler r7 (BPwin). Для информационного моделирования применялось средство MySQL Workbench 6.3 CE.

Для более подробного изучения и рассмотрения системы программы необходимо рассмотреть следующие модели:

* диаграмма последовательностей;
* диаграмма компонентов;
* диаграмма развертывания.

## **5.1 Диаграмма последовательностей (Sequence diagram)**

С помощью диаграммы последовательности рассмотрим процесс редактирования различного рода информации (информации о соискателе, сотруднике HR, вакансии, интервью) администратором системы (рисунок 5.1).

Объектами диаграммы являются администратор, клиент, сервер и база данных (MySQL). От администратора требуется выбрать поле, информацию которого требуется редактировать. После выбора оно будет подсвечено и можно будет вводить новые данные. Как только предыдущие действия будут выполнены, клиентское приложение проведет валидацию данных и, если данные верны, они отправляются на сервер. Тот в свою очередь проверяет корректность принятой информации и в случае успеха сервер формирует запрос и отправляет данные в базу данных, где они заносятся в соответствующие поля и таблицы. В случае успеха, соответствующий ответ отсылается на сервер. Клиентскому приложению передаётся сигнал об успешном редактировании. Администратор может продолжать работу в системе.

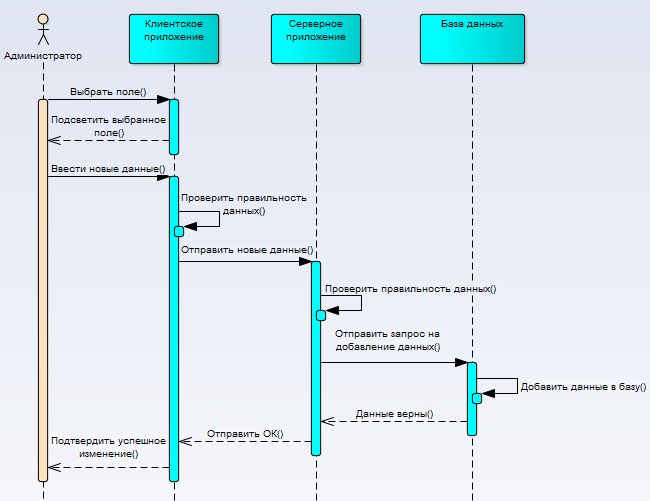


Рисунок 5.1 – Диаграмма последовательностей для вариантов использования «Редактировать информацию»

## **5.2 Диаграмма компонентов**

Полный проект разрабатываемого приложения состоит из совокупности моделей разных уровней, которые должны быть согласованы между собой. Диаграмма компонентов отражает общие зависимости между компонентами. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код. Диаграмма компонентов для данного проекта представлена на рисунке 5.2. Приведенная диаграмма отражает отношение подсистем веб-приложения:

* СУБД предоставляет интерфейс веб-сервису;
* используя предоставленный интерфейс, веб-сервис обращается к СУБД,
* веб-клиенту предоставляется интерфейс веб-сервисом;
* веб-клиент обращается к веб-сервису, применяя данный интерфейс;
* пользователь взаимодействует с веб-клиентом при помощи веб-браузера.



Рисунок 5.2 – Диаграмма компонентов онлайн-системы по поиску работы

## **5.3 Диаграмма развертывания**

С целью визуализации существующих на этапе исполнения элементов программы, обратимся к диаграмме развёртывания. Разработанная диаграмма развёртывания продемонстрирована на рисунке 5.3.

На диаграмме показаны следующие узлы:

* устройство пользователя (персональный компьютер, ноутбук, планшет, смартфон и т.п.), на котором размещается веб-браузер;
* сервер приложений 2, на котором размещается веб-клиент;
* сервер приложений 1, на котором размещается веб-сервис;
* сервер баз данных, на котором размещается СУБД MySQL.



Рисунок 5.3 – Диаграмма развертывания онлайн-системы по поиску работы

# **6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ**

## **6.1 Диаграмма состояний**

Диаграммы состояний являются отличным известным средством описания поведения систем. Они обусловливают все возможные состояния, в которых может находиться конкретный объект, а также процесс изменения состояний объекта в результате воздействия некоторых событий. Состояние (statement) − это ситуация, когда моделируемый объект осуществляет какое-либо условие, проводит операцию или ожидает события. Диаграмма состояния данного проекта описывает состояния аккаунта кандидата в системе и этапы, через которые этот объект проходит. Диаграмма состояний для данного объекта представлена на рисунке 6.1.

****

Рисунок 6.1 – Диаграмма состояний аккаунта кандидата в системе

## **6.2 Описание алгоритма авторизации пользователя**

Данный алгоритм описывает авторизацию всех пользователей, которые пытаются зайти в систему.

На главной странице авторизации пользователя просят ввести логин и пароль для авторизации на сайте. После выполнения данных шагов, прохождения валидации на стороне клиента и получения сервером данных, выполняется соответствующая проверку на корректность этой информации на стороне сервера. В неудачном случае будет выведено сообщение о неверно введенных данных и пользователь так и останется на главной странице авторизации. В случае успеха, полученные логин и пароль будут сверяться с данными единого администратора всей системы, и если они совпадут, то в сессию всего приложения сохранится статус администратора и клиента (в данном случае администратора) перебросит на соответствующую страницу.

Если же логин и пароль не совпали с данными администратора, то идет проверка на попытку зайти на сайт рядовому пользователю. Это означает, что в базу данных посылается запрос на получение объекта с соответствующей информацией, и если вернулся целостный, не null объект, то данный клиент уже был зарегистрирован, успешно сохранен в базу и его можно пропустить в систему для пользования ее услугами.

Но сперва нужно проверить статус пользователя: это клиент либо сотрудник HR. Следующий шаг именно это и делает, и после его выполнения в сессию также устанавливается статус и информация клиента и его перебрасывает на соответствующую его статусу страницу.

Если же объект из базы пришел как null, или же вообще было словлено sql исключение, то клиенту посылается сообщение о неверно введенных данных. Это будет означать, что он либо не зарегистрирован в системе и в базе не было найдено совпадений, либо он просто ввел действительно ошибочные данные. Блок-схема для данного алгоритма представлена на рисунке 6.2.



Рисунок 6.2 – Блок-схема авторизации пользователя

## **6.3 Описание алгоритма отклика кандидатом на вакансию**

Данный алгоритм описывает процесс отклика на любую вакансию любым кандидатом.

Когда кандидат, просматривая вакансии, решил откликнуться на какую-либо из них нажав на соответствующую кнопку, происходит ряд действий, описанных ниже.

Сперва из http-сессии берутся данные о кандидате. На основе этой информации посылается запрос к базе данных на получение идентификатора аккаунта кандидата.

Если же идентификатор пришел не корректный, то пользователя перебрасывает на свою домашнюю страницу с уведомлением об ошибке на веб-сайте. Это объясняется тем, что данные сохраняются в сессию только в случае успешной авторизации и не могут быть не валидными, а это означает, что при попытке взять id произошла ошибка в базе.

В случае корректного возвращения идентификатора аккаунта, на его основе происходит получение из базы информации о данном кандидате.

Если объект пришел пустым либо некорректным, то пользователя также перебрасывает на свою домашнюю страницу с сообщением, что он не добавил резюме и без него не может откликаться на вакансии.

В обратном случае, с пришедшего на сервер запроса достается идентификатор вакансии, на которую откликнулись и на его основе и информации кандидата формируется объект интервью.

Далее проверяется, существует ли уже данное интервью на случай, если кандидат уже откликался на нее. Если это так, то пользователя просто возвращает на страницу просмотра всех вакансий. Но если же данного интервью еще нет в базе, то идет попытка добавить его. В случае провала кандидата возвращает на свою домашнюю страницу с сообщением об ошибке на сервере. При успешном завершении операции пользователя выкидывает на страницу просмотра всех вакансий, уведомляя его об успешном отклике на данную вакансию.

Блок-схема для данного алгоритма представлена на рисунке 6.3.



Рисунок 6.3 – Блок-схема отклика кандидатом на вакансию

# **7 ОПИСАНИЕ ПРИМЕНЕНИЯ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ**

## **7.1 Паттерн «Одиночка» (Singleton)**

Паттерн «Одиночка» принадлежит классу порождающих паттернов. Его используют для создания единственного экземпляра класса. Есть гарантия, что во время работы программы не появиться второй, и предоставляет этому объекту глобальную точку доступа.

В данном приложении паттерн используется в случае работы с пулом соединений для базы данных на серверной стороне.

Например, в работе с базой данных данный паттерн необходим, так как объект класса, работающего с пулом соединений, требуется в разных классах, и чтобы каждый раз не происходило подключение к базе данных, необходим объект с глобальной точкой доступа.

UML-схема паттерна проектирования «Одиночка» изображена на рисунке 7.1.

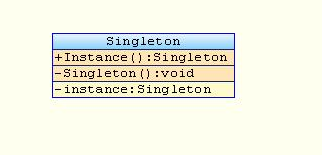


Рисунок 7.1 – UML схема паттерна проектирования «Одиночка»

## **7.2 Паттерн «Команда» (Command)**

Команда (Command) — поведенческий шаблон проектирования, используемый при объектно-ориентированном программировании, представляющий действие. Объект команды заключает в себе само действие и его параметры.

При работе с приложением пользователь выполняет различные операции, в ответ система всегда должна знать, где находятся данные для ее выполнения и какие действия следует выполнить. Все данные, необходимые для выполнения операции можно объединить в один объект, который и будет определять действие, или, по-иному, команду.

При передаче в бизнес-логику системы запроса на действие в произвольном виде, запрос таким образом преобразуется в объект-команду, метод которого может быть вызван, а сам объект может быть сохранен и/или передан в качестве параметра метода или возвращен как любой другой объект. Инкапсулирует запрос как объект.

Объект-источник запроса отделяется от команды, но от его типа зависит, какая из команд будет выполнена.

В данном случае паттерн используется для разделения источника запроса и его исполнителя, а также для построения последовательности команд, порядок которых определяет сам пользователь в зависимости от своего желания или результата выполнения предыдущей команды.

UML-схема паттерна проектирования «Команда» изображена на рисунке 7.2.

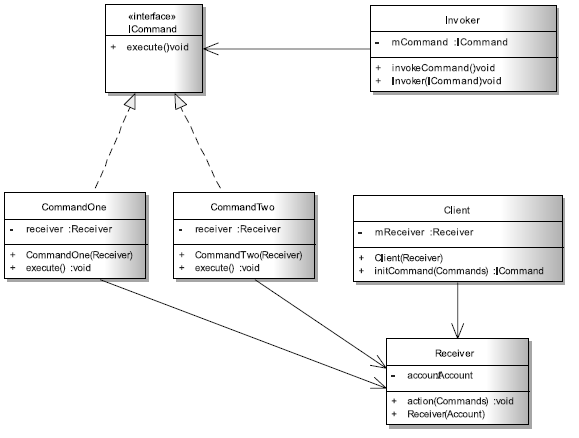


Рисунок 7.2 – UML схема паттерна проектирования «Команда»

## **7.3 Паттерн «Заместитель» (Proxy)**

Заместитель (Proxy) — структурный шаблон проектирования, предоставляющий объект, который контролирует доступ к другому объекту, перехватывая все вызовы (выполняет функцию контейнера).

Прокси-объект представляет другой объект. С точки зрения клиента интерфейс и функциональность класса остаются неизменными. Чтобы это представление было естественным, прокси-объект обязан реализовывать тот же интерфейс, что и реальный класс. Кроме того, прокси-объект должен содержать реальный класс в качестве поля, чтобы при необходимости обращаться к функционалу реального класса. В общем случае реализует технологию обертывания (wrapping) класса с целью повышения безопасности или оптимизации взаимодействия.

UML-схема паттерна проектирования «Команда» изображена на рисунке 7.3.

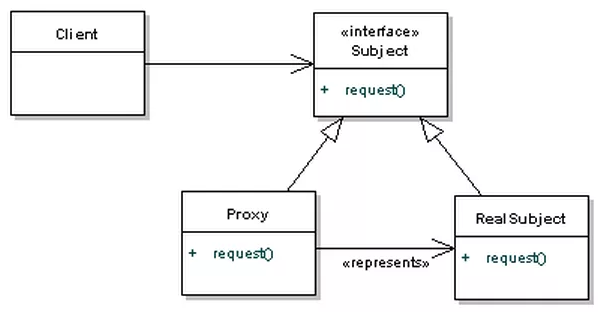


Рисунок 7.3 – UML схема паттерна проектирования «Заместитель»

# **8 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ СИСТЕМЫ**

Процесс по развертыванию данного проекта по поиску работы подразделяется на несколько шагов.

Сперва для функционирования веб-приложения требуется СУБД MySQL. Рекомендуется версия 6.3 CE или выше.

С помощью MySQL необходимо создать новую базу данных. Для этого необходимо открыть файл SQL-скрипта script.sql из поставки и запустить его посредством нажатия на кнопку «Execute the selection portion of the script or everything, if there is no selection» (рисунок 8.1).

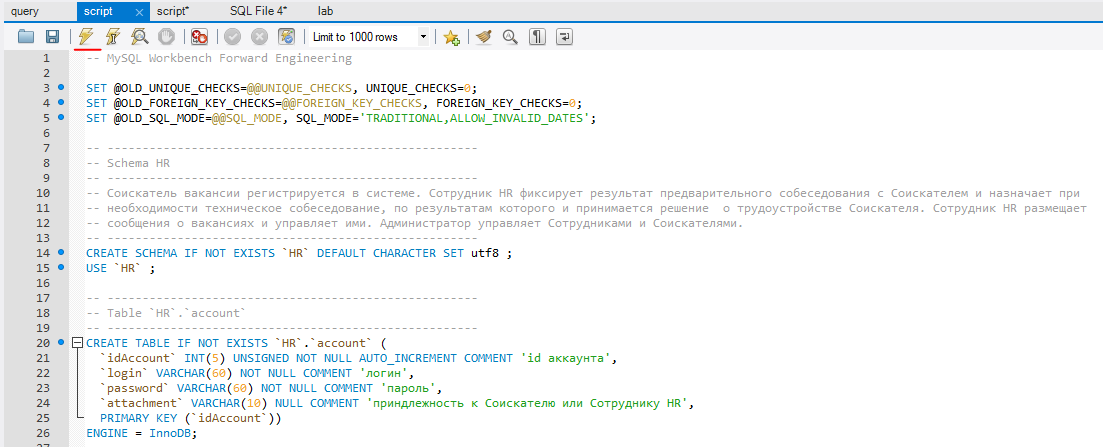


Рисунок 8.1 – Выполнение sql-скрипта

В результате успешного выполнения данного скрипта в разделе «SCHEMAS» должна появиться полностью готовая к эксплуатации база данных (рисунок 8.2). Такой вариант развертывания позволяет не привязываться к конкретному имени базы данных.

Далее необходимо настроить права доступа к базе данных согласно применяемой схеме авторизации согласно общим инструкциям.

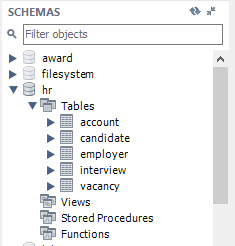


Рисунок 8.2 – Результат успешного выполнения sql-скрипта

Следующим шагом по развертыванию системы будет загрузка сервера на хостинг. Для его правильного выполнения нужно скачать и установить контейнер сервлетов либо сервер приложений. Рекомендуются известные реализации такие как Apache Tomcat, Jetty, WildFly (бывший JBoss), GlassFish, IBM WebSphere. В плане установки все просто и для каждого из перечисленных контейнеров и серверов установка – это просто распаковка архива.

Настройка сервера описывается для Apache Tomcat и начинается с настройки порта HTTP, на котором он будет работать. Однако следует предупредить, что для каждого из серверов есть свои особенности настройки.  
Для Tomcat нужно зайти в папку с распакованным tomcat, далее папка conf, файл server.xml. Нужно найти в этом файле число 8080 (http порт по умолчанию) и поменять его на что захотите.

Теперь нужно запустить tomcat. Для этого требуется перейти в папку с распакованным tomcat, далее папка bin и запустить файл startup.bat. Затем перейти в браузере по адресу http://localhost:8080/ (порт по умолчанию). Tomcat успешно начал свою работу, если в браузере откроется страничка как на рисунке 8.3.

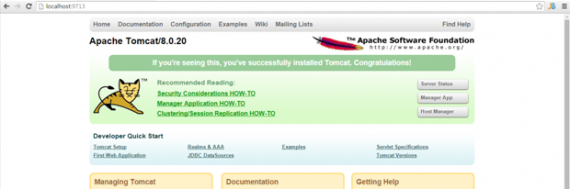


Рисунок 8.3 – Успешный старт Apache Tomcat

Кроме того, можно прописать себя как админа сервера для увеличения функционала, например, для деплоя приложения и через графический интерфейс, и через удаленное подключение.

Для этого нужно перейти в папку с распакованным Tomcat, затем в папку conf, открыть файл tomcat-users.xml. В нем перед закрывающим тегом </tomcat-users> прописать следующий тег:

<user username="egarmin" password="1" roles="manager-gui,manager-script,manager-status,manager-jmx"/>

После чего необходимо скопировать war-файл из поставки в папку webapps и запустить Tomcat c помощью файла startup.bat

Последним шагом развертывания будет загрузка клиента на хостинг, которая выполняется в точно такой же последовательности, как и второй шаг.

# **9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СИСТЕМЫ**

После запуска рассматриваемого программного продукта нас встречает страница авторизации (рисунок 9.1).

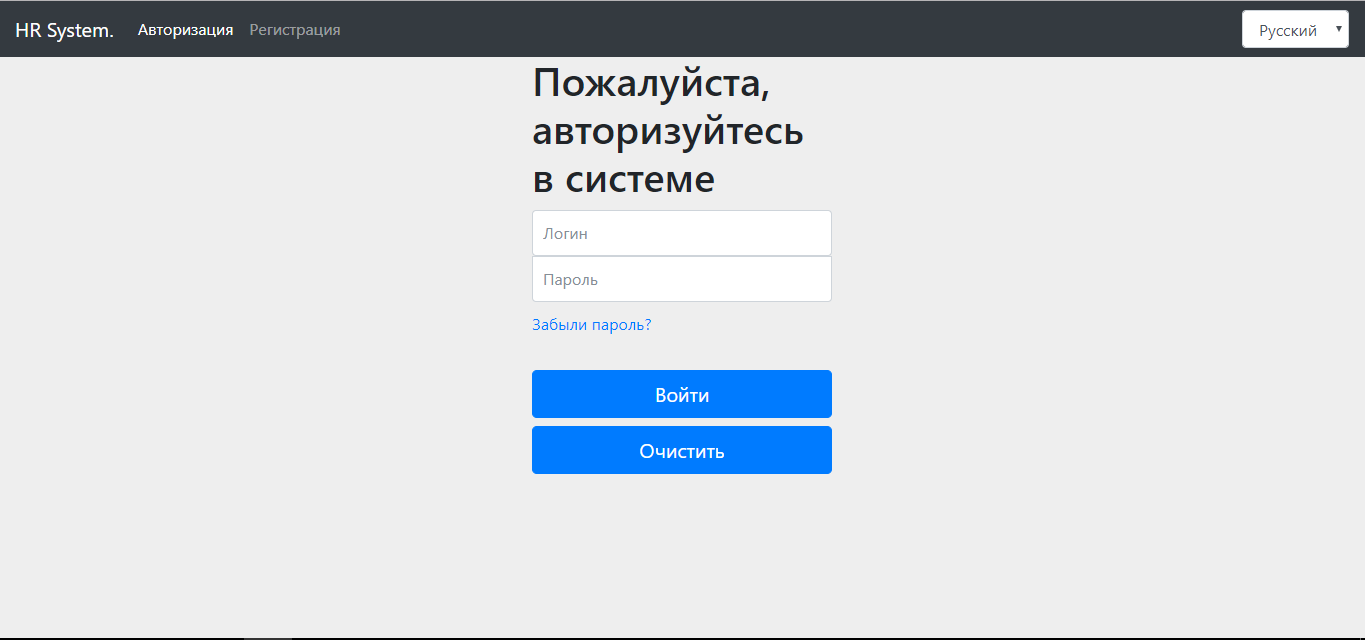


Рисунок 9.1 – Страница авторизации

После ввода данных, идет проверка их корректности и в зависимости от этих данных, в базе проверяется совпадение. Если они были найдены, пользователя перебросит на соответствующую страницу – кандидата, сотрудника HR или же администратора. В обратном случае, будет выведено сообщение о некорректно введенных данных.

Также имеется кнопка «Забыли пароль?» (рисунок 9.2).

Также на данном этапе можно перейти на страницу регистрации (рисунок 9.3).

Здесь требуется ввести свой логин, пароль, подтвердить его, а также нажать на галочку, что вы не робот. Все поля также наделены проверкой корректности как на клиентской, так и на серверной стороне.

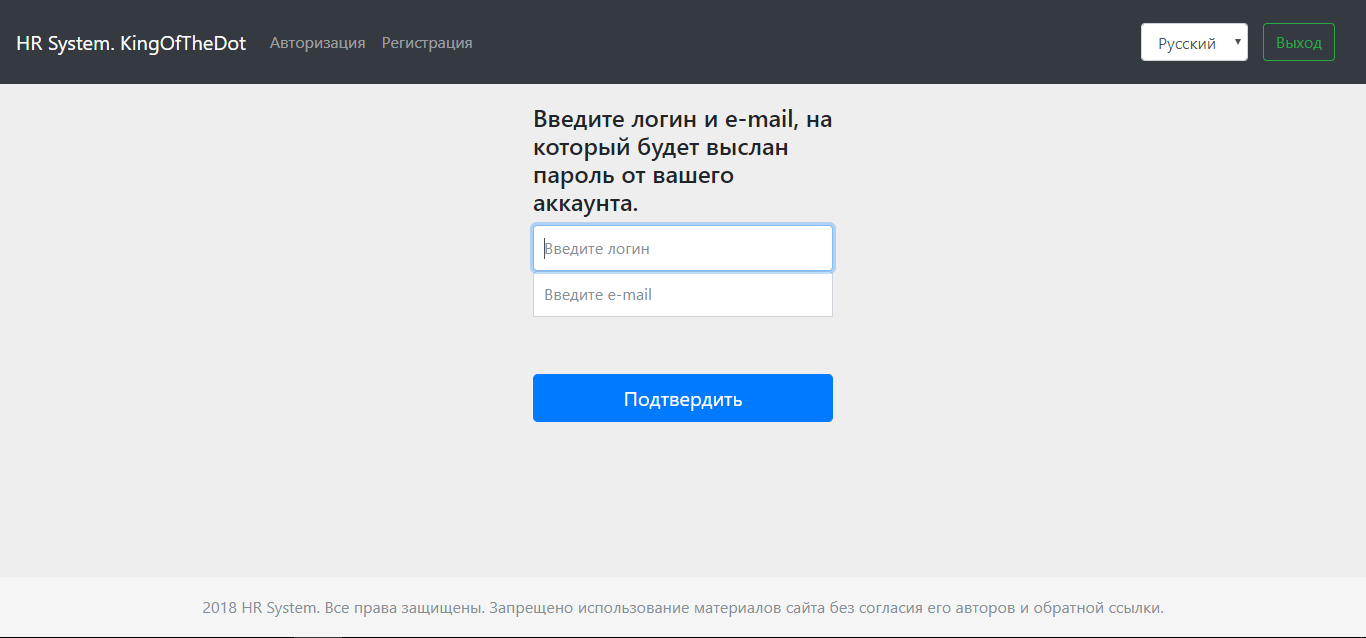


Рисунок 9.2 – Страница «Забыли пароль?»

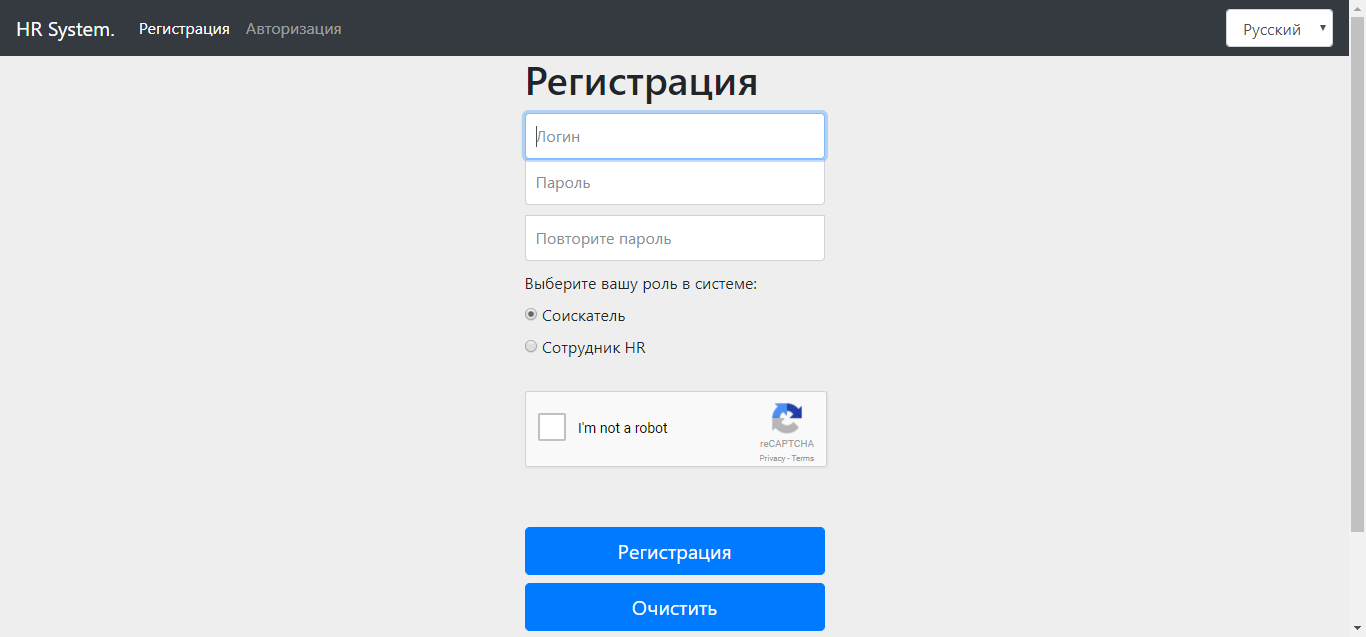


Рисунок 9.3 – Страница регистрации

Кроме того, на каждой странице есть выбор используемого языка интерфейса (рисунок 9.4).

Для демонстрации работы программного продукта создадим аккаунт кандидата на странице регистрации. После успешного прохождения данного этапа нас перебросит на его домашнюю страницу (рисунок 9.5).

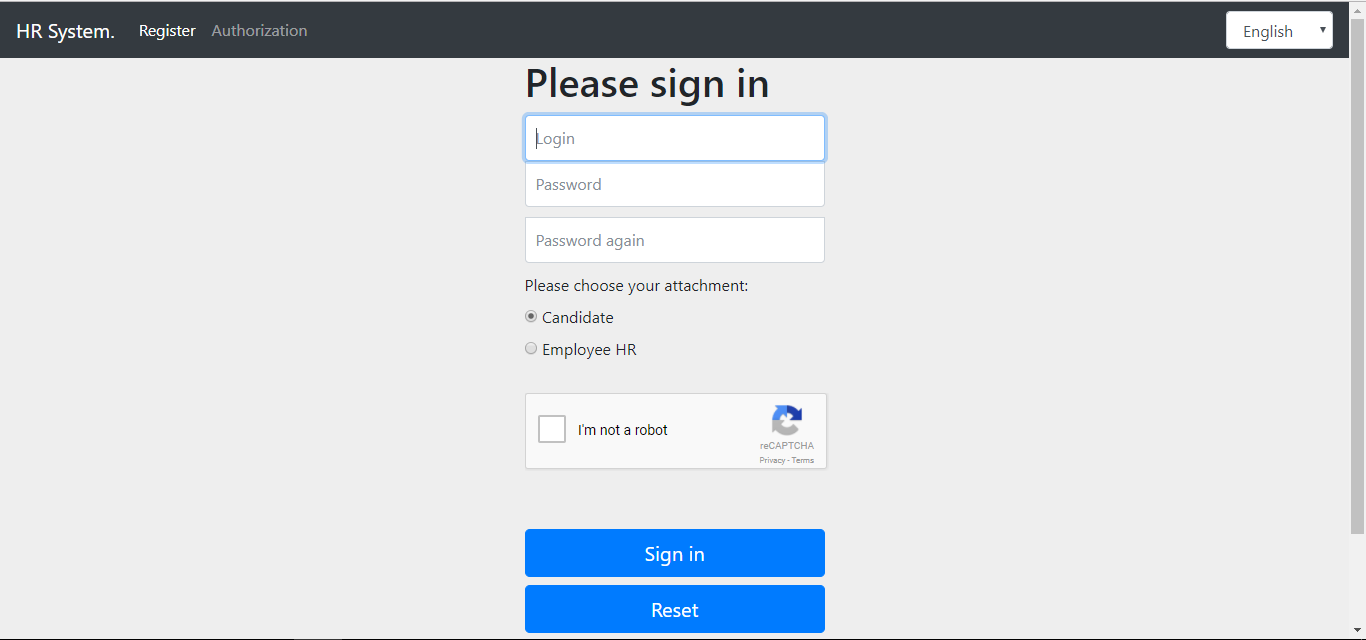


Рисунок 9.4 – Демонстрация английского языка интерфейса

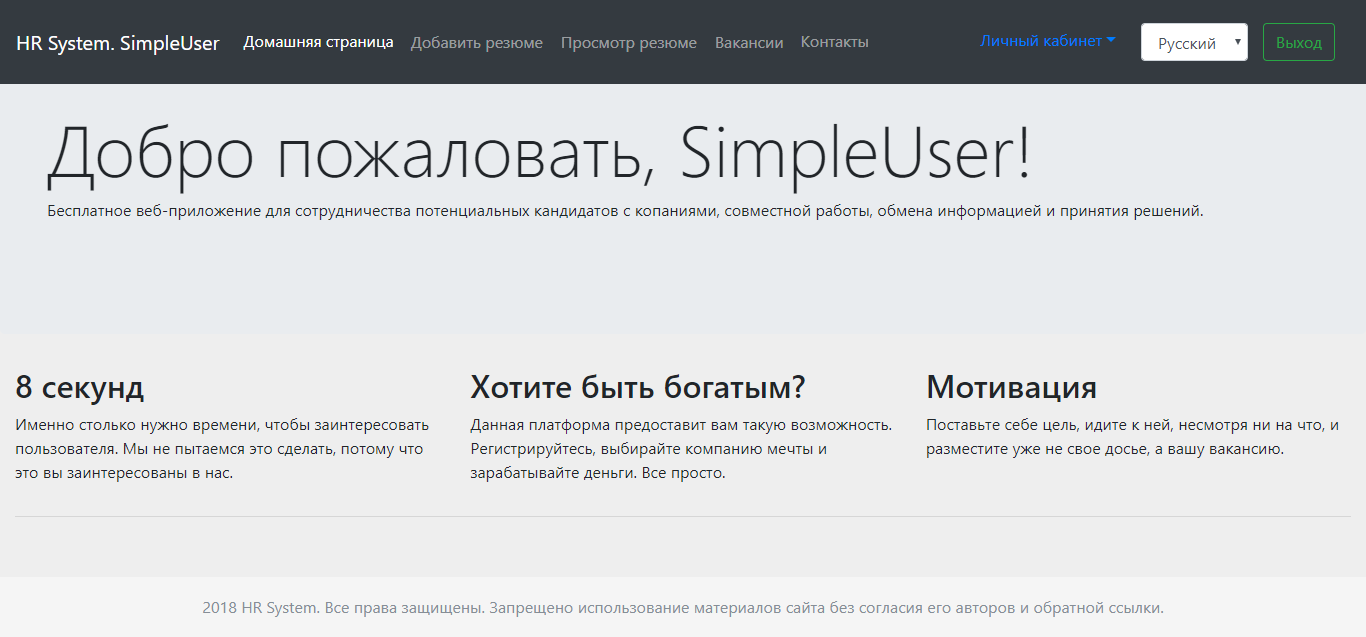


Рисунок 9.5 – Домашняя страница кандидата

Как кандидат, так и сотрудник HR имеют возможность изменить данные своего аккаунта (рисунок 9.6), а также удалить его вовсе (рисунок 9.7).

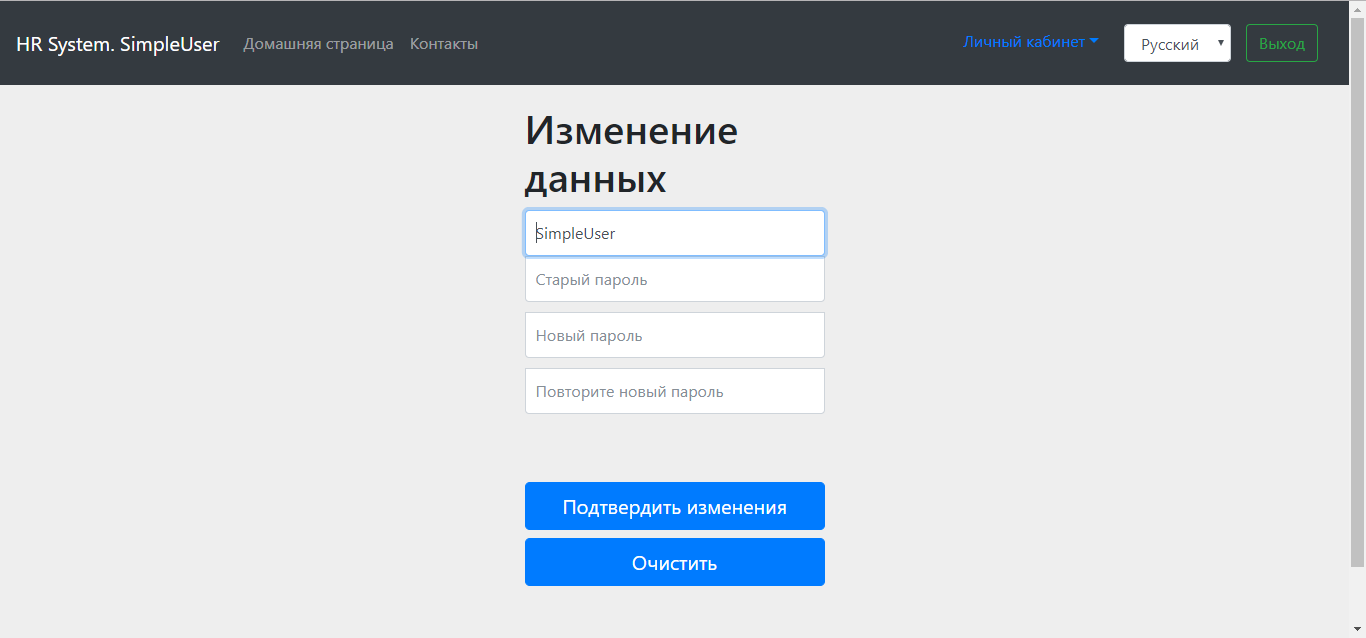


Рисунок 9.6 – Изменение данных аккаунта

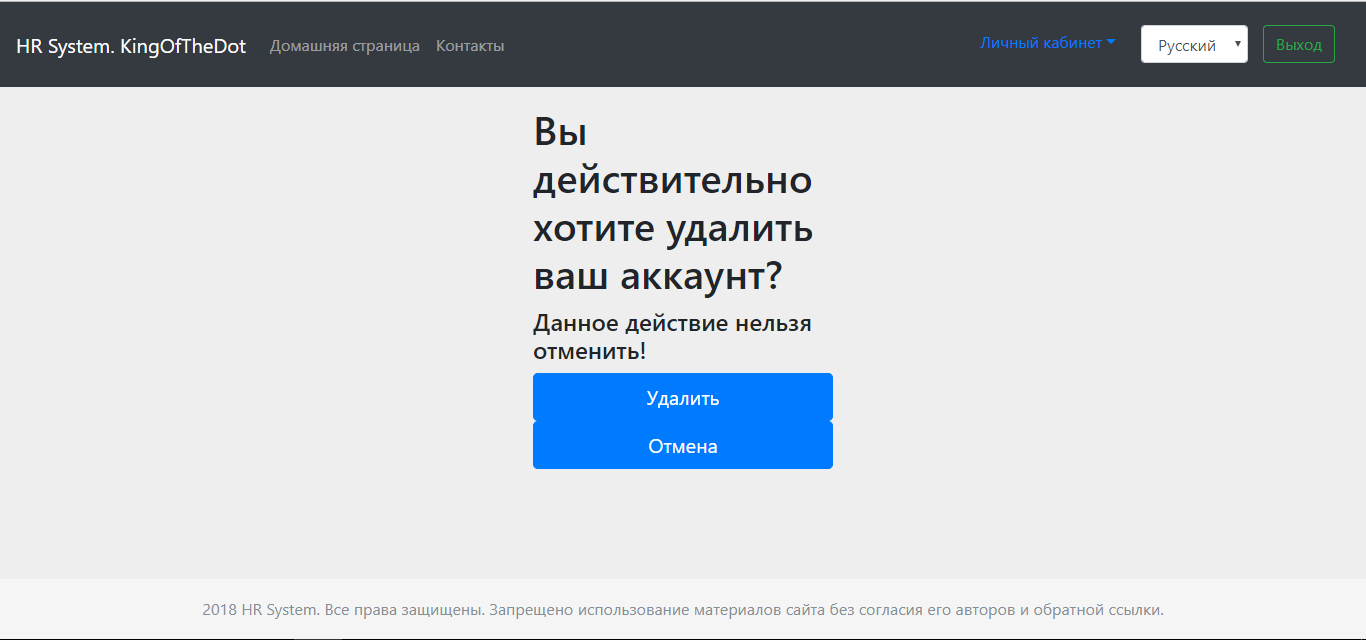


Рисунок 9.7 – Удаление аккаунта

Перейдем к рассмотрению основных функций кандидата.

Первое – это добавление резюме о себе (рисунок 9.8).

Каждое поле имеет защиту от неправильного ввода как на клиентской, так и на серверной стороне.

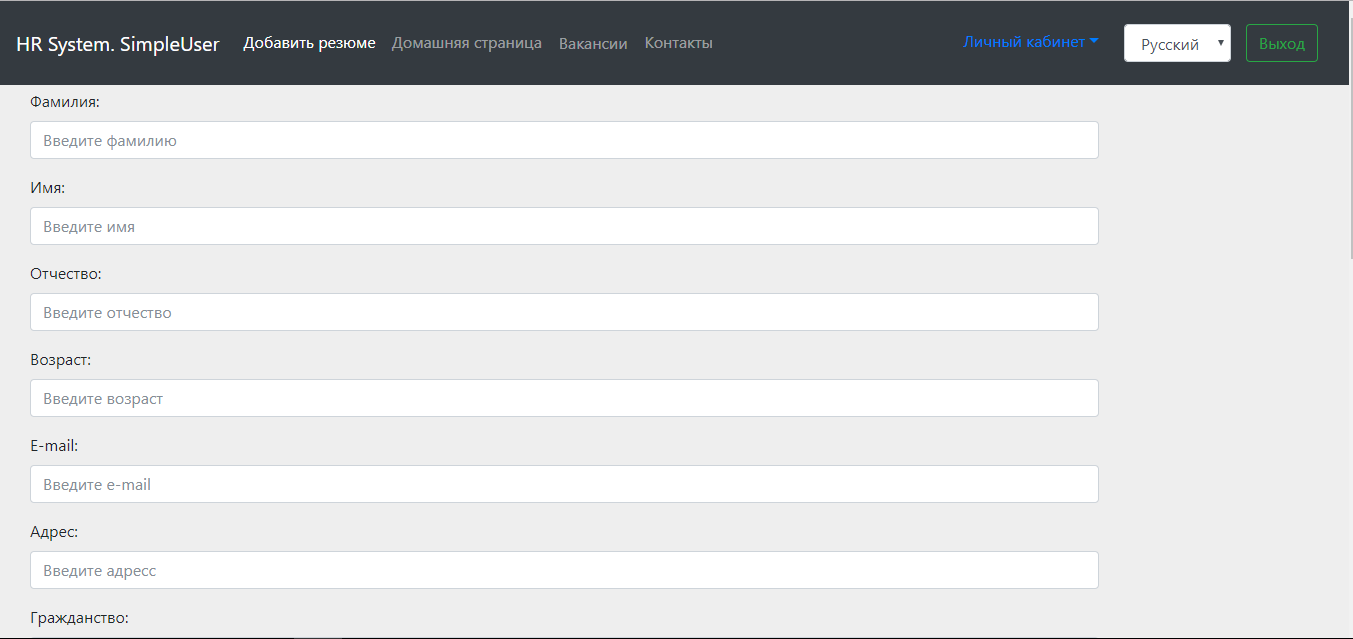


Рисунок 9.8 – Добавление резюме кандидатом

Кроме того, если кандидат попытается создать резюме, если оно уже имеется, то ему будет высвечено соответствующее предупреждающее сообщение (рисунок 9.9).

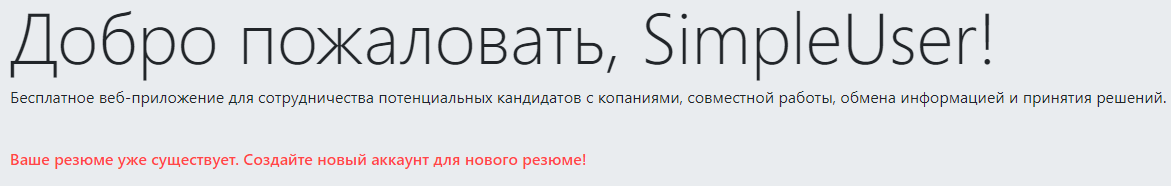


Рисунок 9.9 – Предупреждение о уже имеющимся резюме

Далее в списке основных функций – это просмотр и редактирование своего резюме (рисунок 9.10).

По умолчанию все поля заблокированы и предназначены для просмотра, однако если нажать соответствующею каждому полю кнопку «Редактировать», то поле станет открытым для ввода и изменения данных.

Все поля на данной странице также обладают защитой от неправильного ввода.

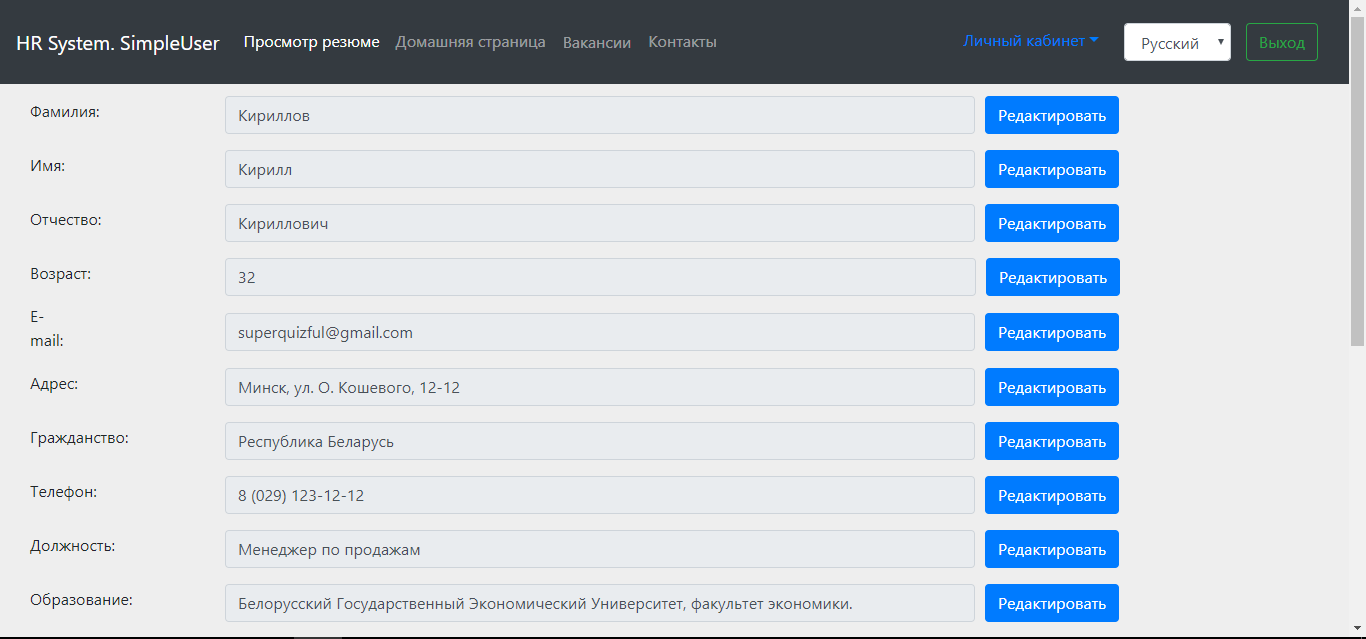


Рисунок 9.10 – Просмотр и редактирование резюме

Далее – это возможность кандидату просматривать список всех вакансий (рисунок 9.11).

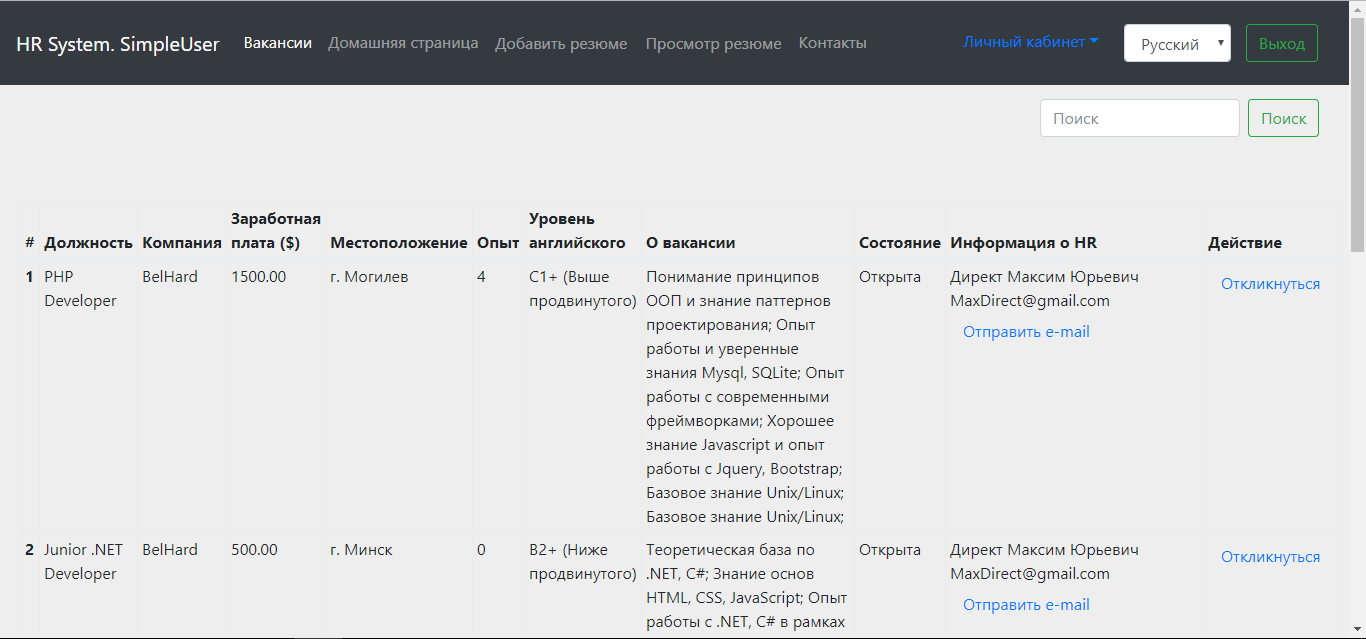


Рисунок 9.11 – Просмотр всех вакансий

Информация, представленная на данной странице понятна сама собой, поэтому перейдем к рассмотрению кнопки «Отправить e-mail» в колонке «Информация о HR».

По названию колонки следует, что в ней содержатся краткие данные о HR’e, который представлен за данной вакансией и имеется возможность отправить ему письмо на указанный электронный адрес (рисунок 9.12).

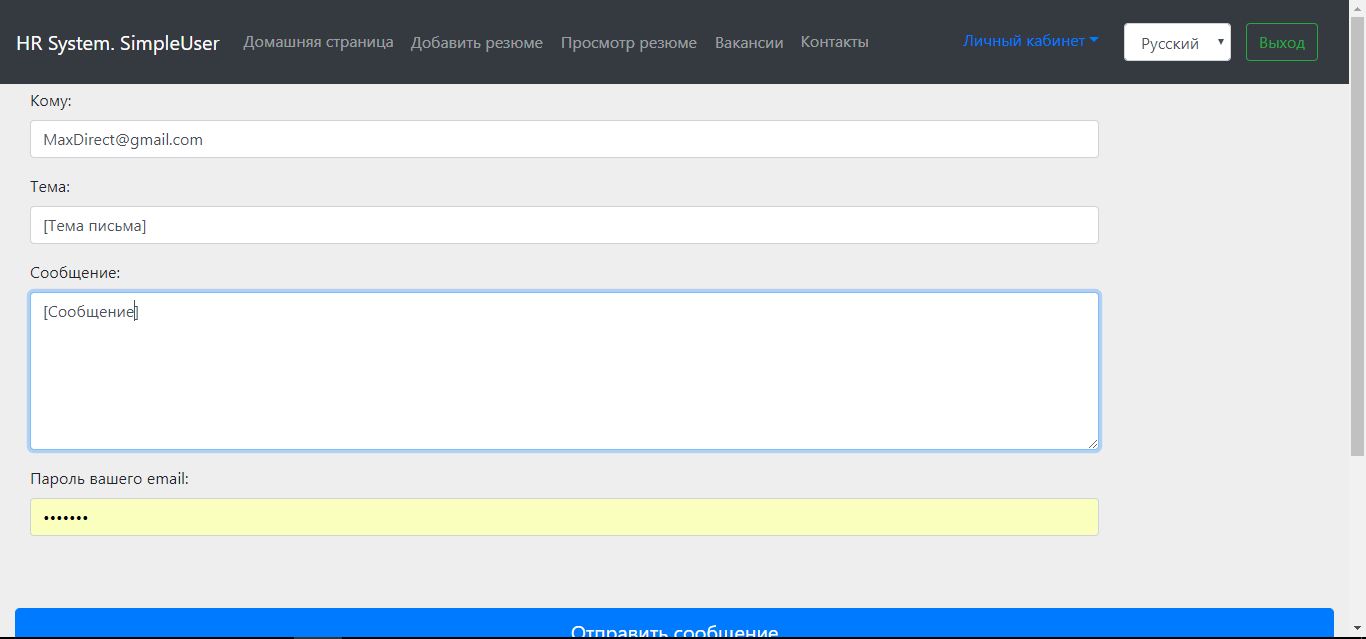


Рисунок 9.12 – Отправка электронного письма

Также в колонке «Действие» имеется кнопка «Откликнуться».

При нажатии на нее в базу идет запрос на формирование нового интервью. Результат нажатия на нее представлен на рисунке 9.13.

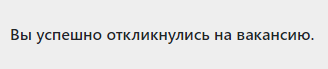


Рисунок 9.13 – Результат отклика на вакансию

Однако, если кандидат попытается откликнуться на вакансию, на которую он уже откликался, то ему выскачет соответствующее сообщение (рисунок 9.14).

D:\УНИВЕР\Курсачи\Курсач СИТАИРИС\screenshots\14_Кандидат_уже_откликнулся.PNG

Рисунок 9.14 – Попытка отклика на вакансию, на которую уже откликались ранее

Также почти на каждой странице в блоке навигации можно заметить ссылку на вкладку «Контакты». Эта страничка содержит краткие сведения о компании и людях, которые разрабатывали данный программный продукт (рисунок 9.15).

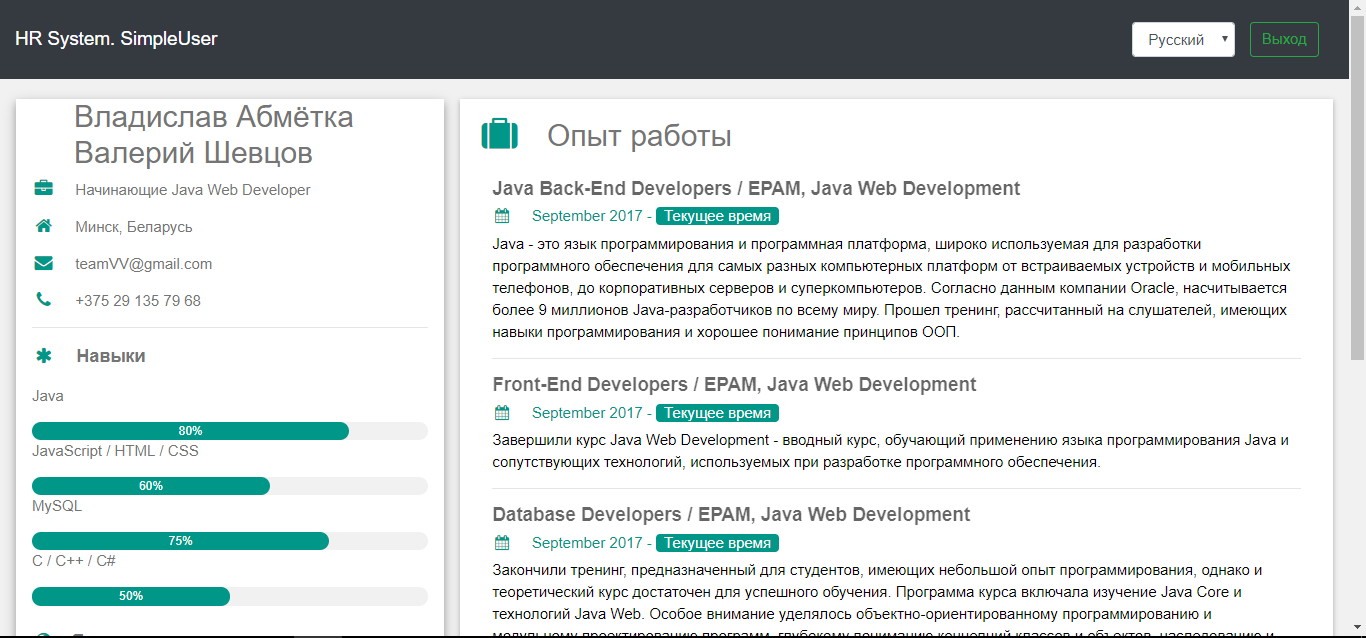


Рисунок 9.15 – Страница «Контакты»

Теперь перейдем к рассмотрению функций сотрудника HR.

Для него также создадим демонстрационный аккаунт и после этого мы попадаем на его главную страницу (рисунок 9.16).

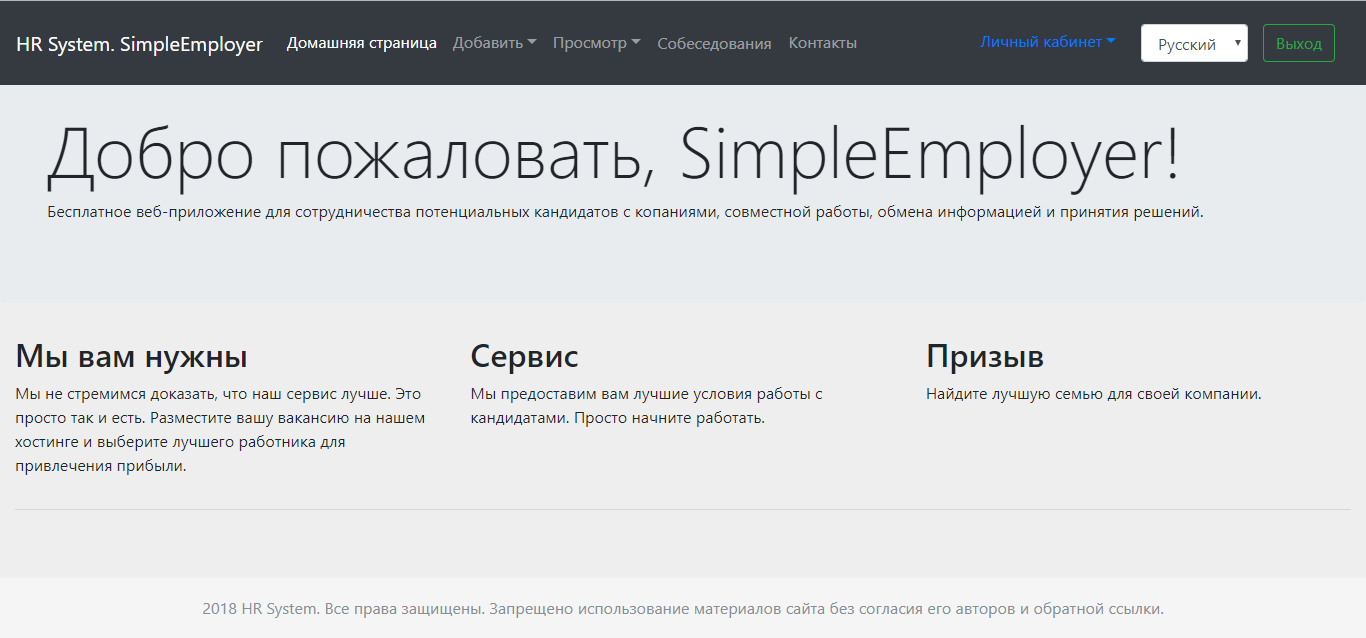


Рисунок 9.16 – Главная страница сотрудника HR

Первым делом, что должен сделать сотрудник – это добавить личную информацию о себе. Без этого он не сможет осуществлять все остальные функции, такие как создание резюме (рисунок 9.17), просмотр личной информации (рисунок 9.18), просмотр вакансий (рисунок 9.19) и просмотр собеседований (рисунок 9.20).

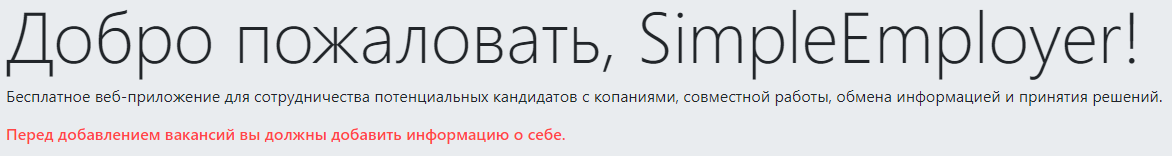


Рисунок 9.17 – Запрет на создание вакансий

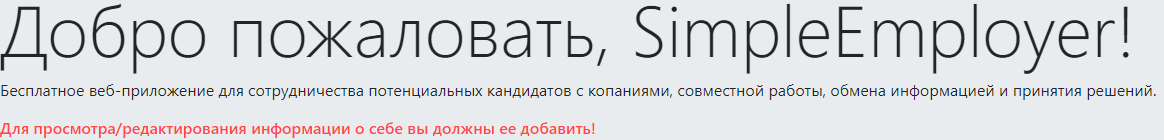


Рисунок 9.18 – Запрет на просмотр и редактирование личной информации

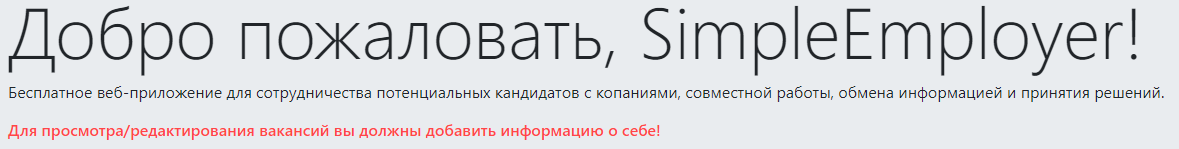


Рисунок 9.19 – Запрет на просмотр и редактирование вакансий

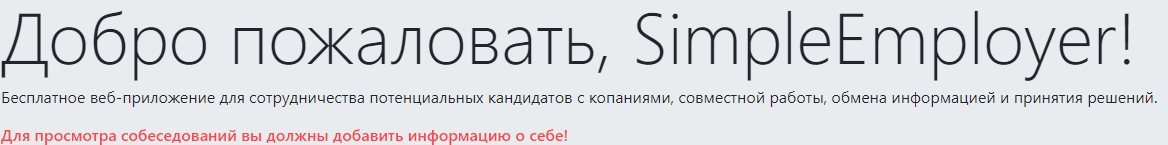


Рисунок 9.20 – Запрет на просмотр собеседований

Итак, добавим личную информацию о сотруднике (рисунок 9.21).

Все также, каждое поле имеет защиту от неправильного ввода как на клиента, так и на сервере, и при попытке ввести некорректные данные будет высвечено соответствующая информация.

После успешного добавления личной информации ее можно только просматривать и редактировать, опять создать ее уже будет нельзя. Попытка это сделать представлено на рисунке 9.22, где показано сообщение об ошибке.

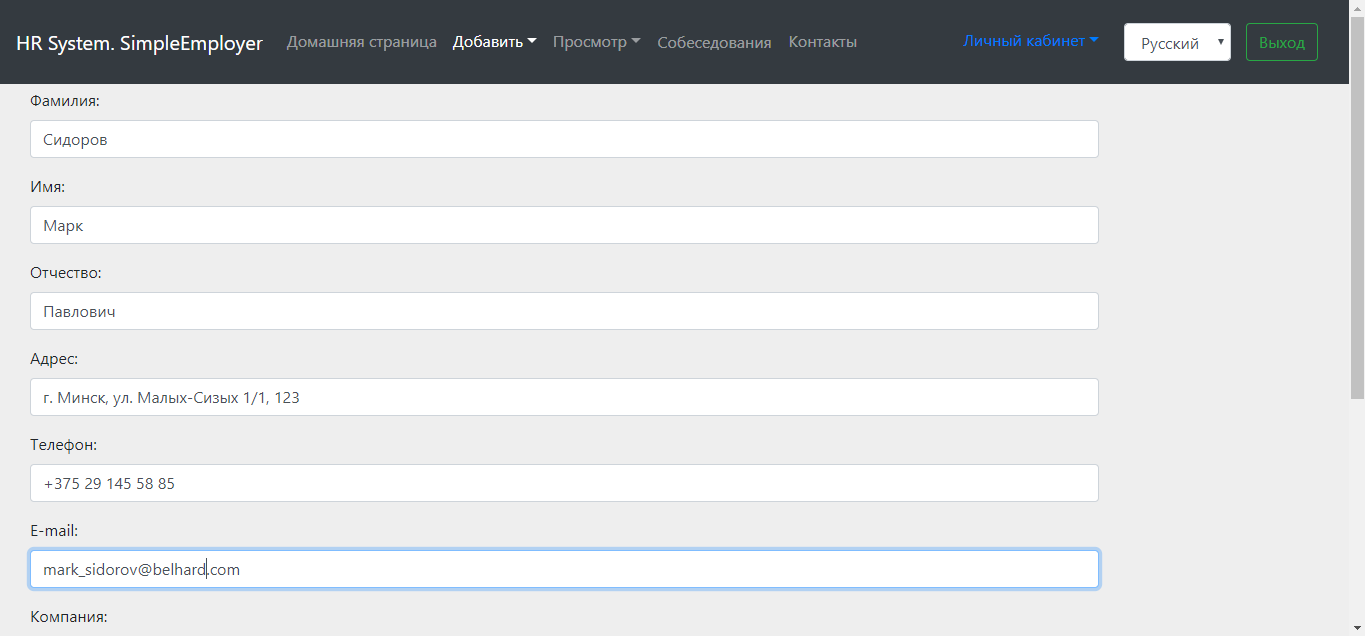


Рисунок 9.21 – Добавление личной информации о сотруднике HR

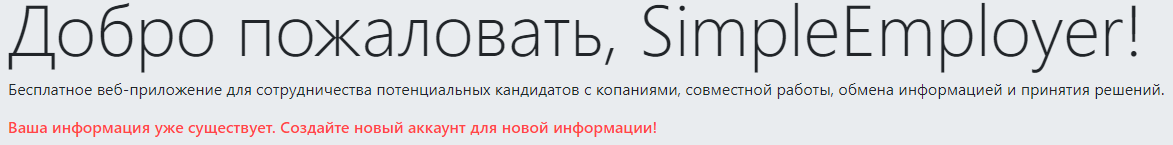


Рисунок 9.22 – Запрет на повторное добавление личной информации

Далее после добавления личной информации уже можно выполнять и все оставшиеся функции. Создадим новую вакансию (рисунок 9.23).

Опять же, все представленные на рисунке поля имеют защиту от некорректного ввода на клиенте и сервере.

Функция просмотра и редактирования личной информации о сотруднике HR представлена на рисунке 9.24.

Логика действий точно такая же, как и в случае с кандидатом.

По умолчанию все поля заблокированы и предназначены для просмотра, однако если нажать соответствующею каждому полю кнопку «Редактировать», то поле станет открытым для ввода и изменения данных.

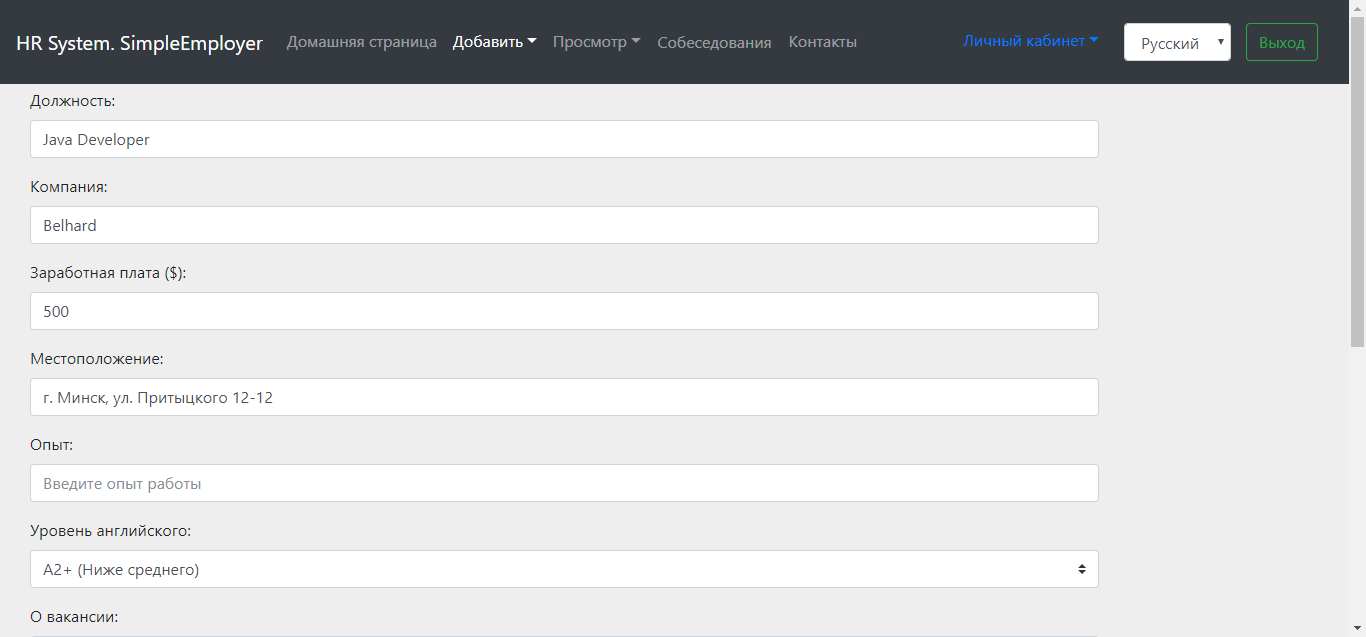


Рисунок 9.23 – Добавление вакансии

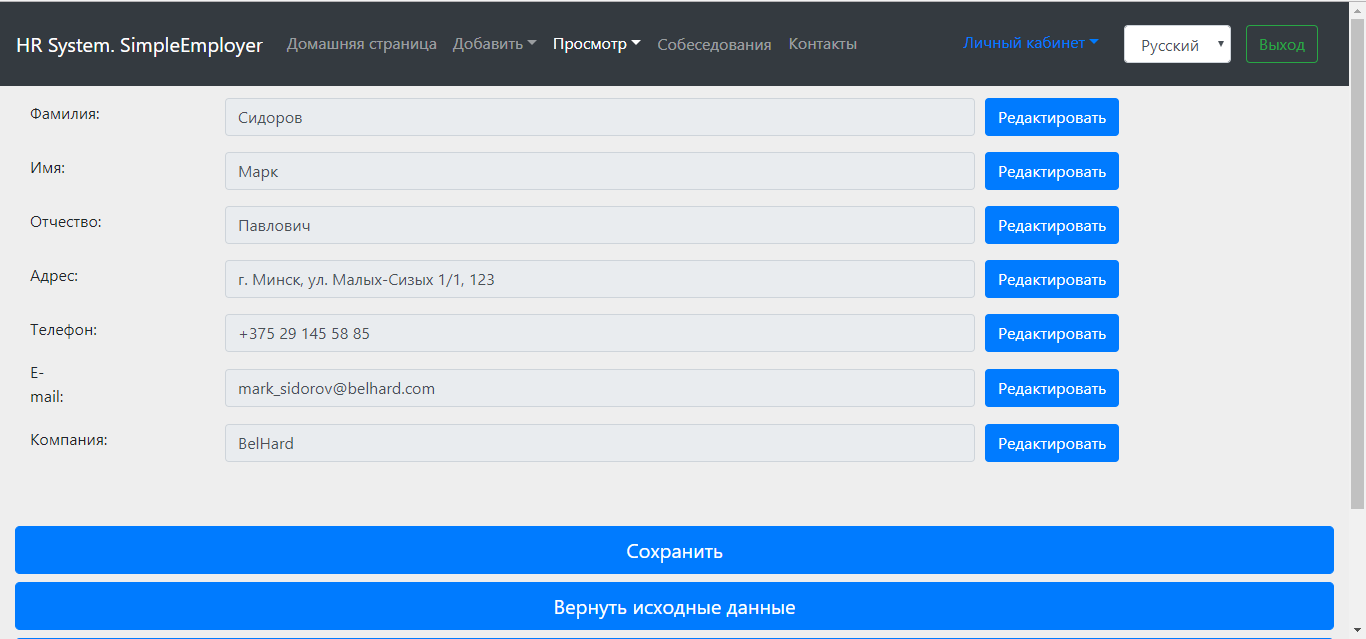


Рисунок 9.24 – Просмотр и редактирование личной информации сотрудника HR

На рисунке 9.25 представлена возможность просмотра всех вакансий.

На этой странице данному сотруднику доступны только те вакансии, которые он сам же и создал. Соответственно функции удаления и редактирования доступны только для этих вакансий.

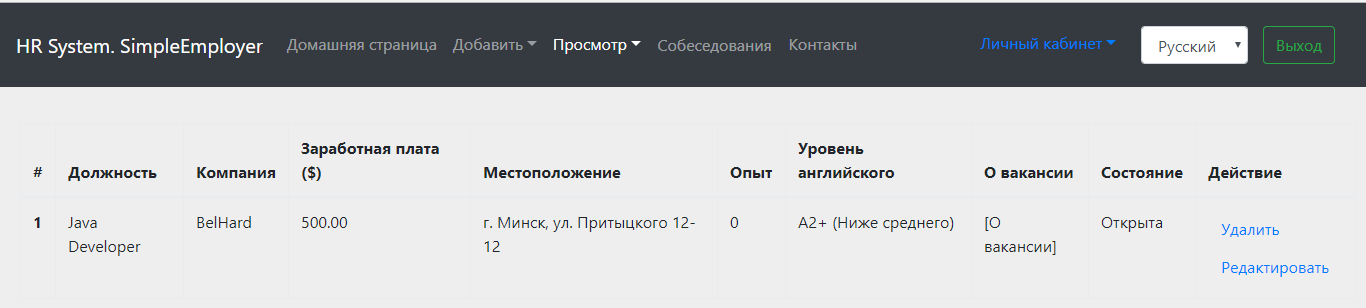


Рисунок 9.25 – Просмотр всех вакансий

Далее перейдем к рассмотрению просмотра и управления собеседованиями. Однако, если на выложенные в систему вакансии данного сотрудника HR никто не откликался, при попытке открыть страницу просмотра интервью, HR получит сообщение, представленное на рисунке 9.26.

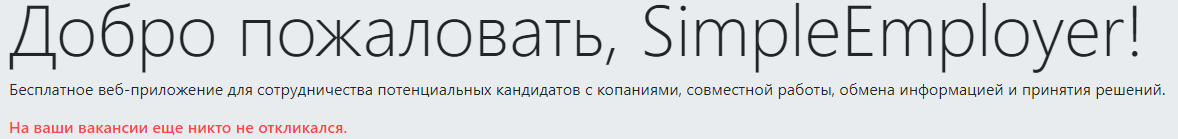


Рисунок 9.26 – На вакансии сотрудника HR никто не откликался

Но если хотя бы единственная душа все-таки откликнулась на вакансию, то просмотр собеседований будет доступен (рисунок 9.27).

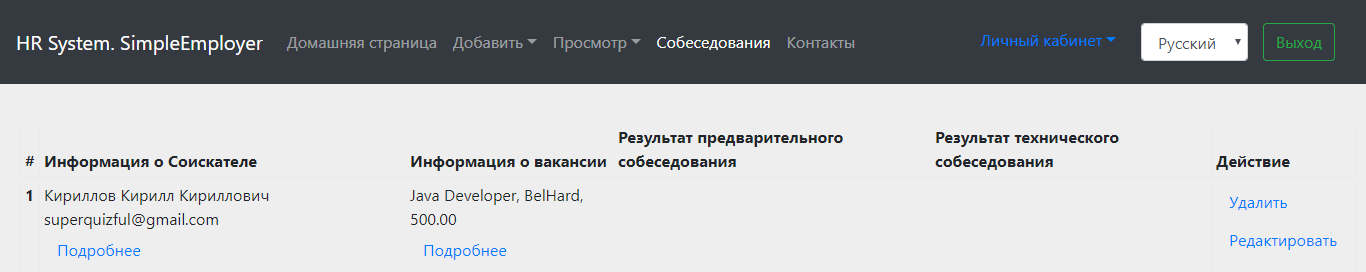


Рисунок 9.27 – Просмотр собеседований

В колонках «Информация о Соискателе» и «Информация о вакансии» находится лишь краткая информация о них, но посредством нажатия на кнопку «Подробнее» имеется возможность просмотреть полные данные.

Также имеется возможность удалить собеседование из системы (кнопка «Удалить» в колонке «Действий») и редактировать результаты собеседований (кнопка «Редактировать» в колонке «Действий»), что и представлено на рисунке 9.28.

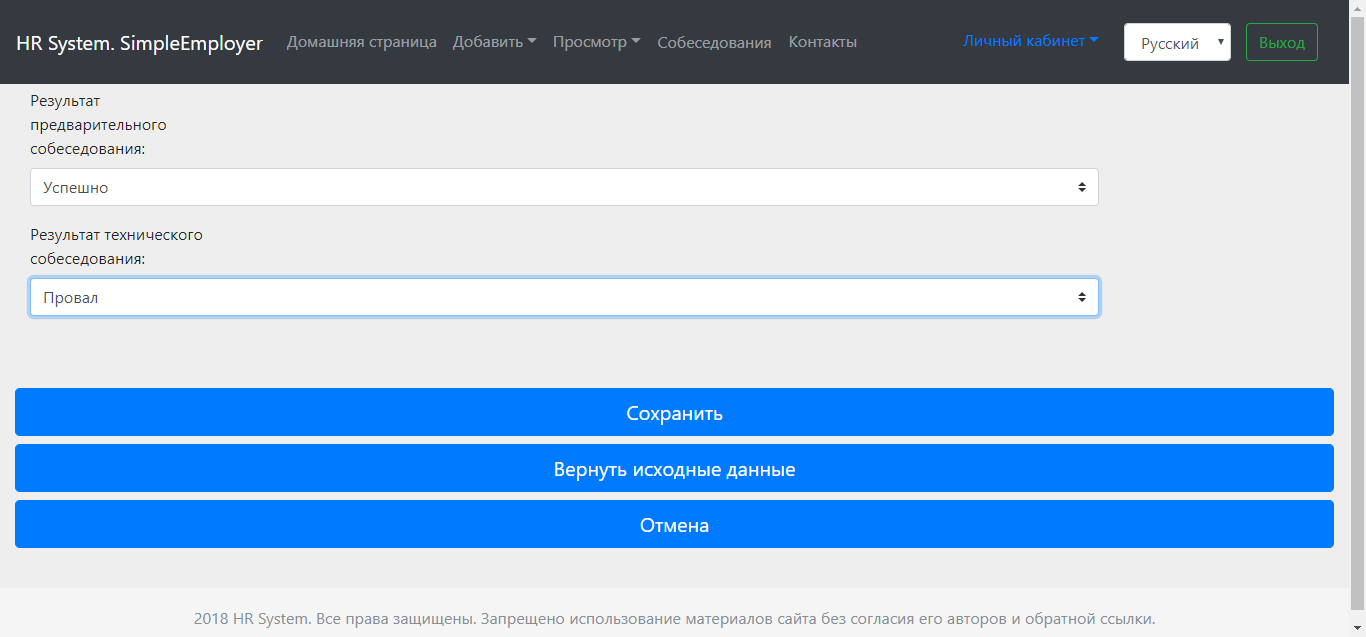


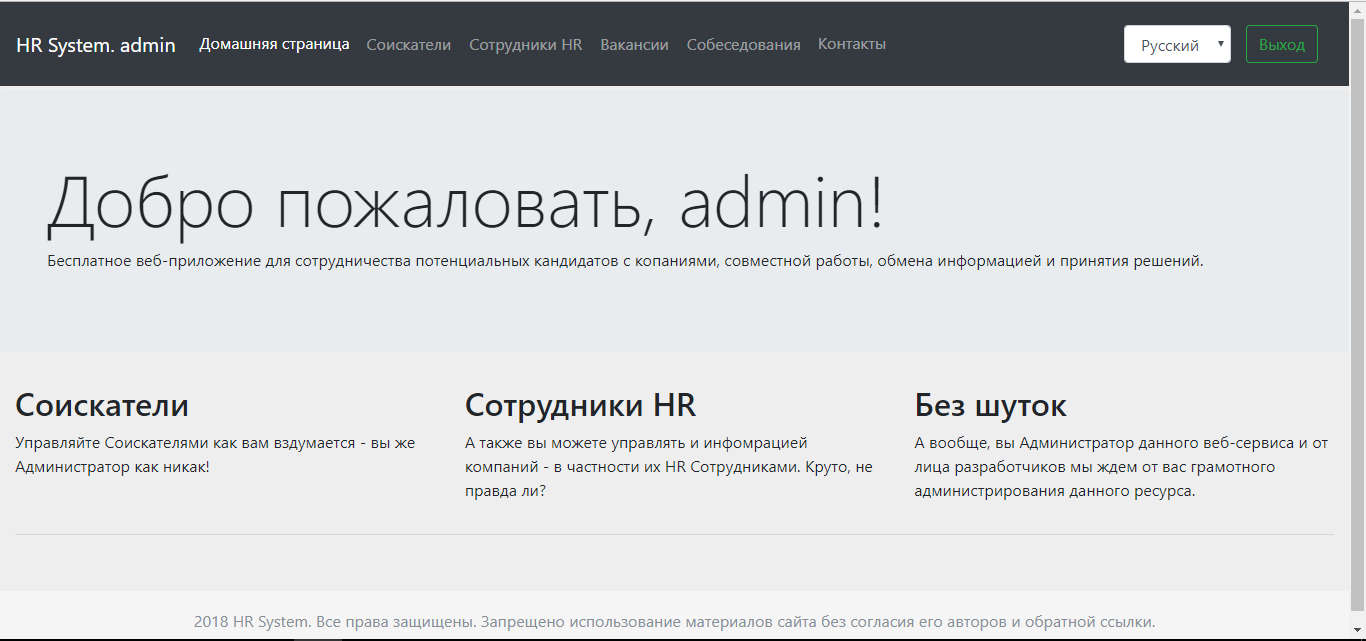
Рисунок 9.28 – Редактирование результатов собеседования

Далее перейдем к рассмотрению возможностей администратора.

Вкратце, он имеет все полномочия как кандидата, так и сотрудника HR, описанные выше.

Введем логин и пароль администратора на странице авторизации и попадем на его главную страницу (рисунок 9.29).

Одними из всех полномочий администратора являются возможность просмотра списка всех кандидатов (рисунок 9.30), всех сотрудников HR (рисунок 9.31), всех вакансий (рисунок 9.32), а также всех собеседований (рисунок 9.33).



Рисунок

Рисунок 9.29 – Главная страница администратора

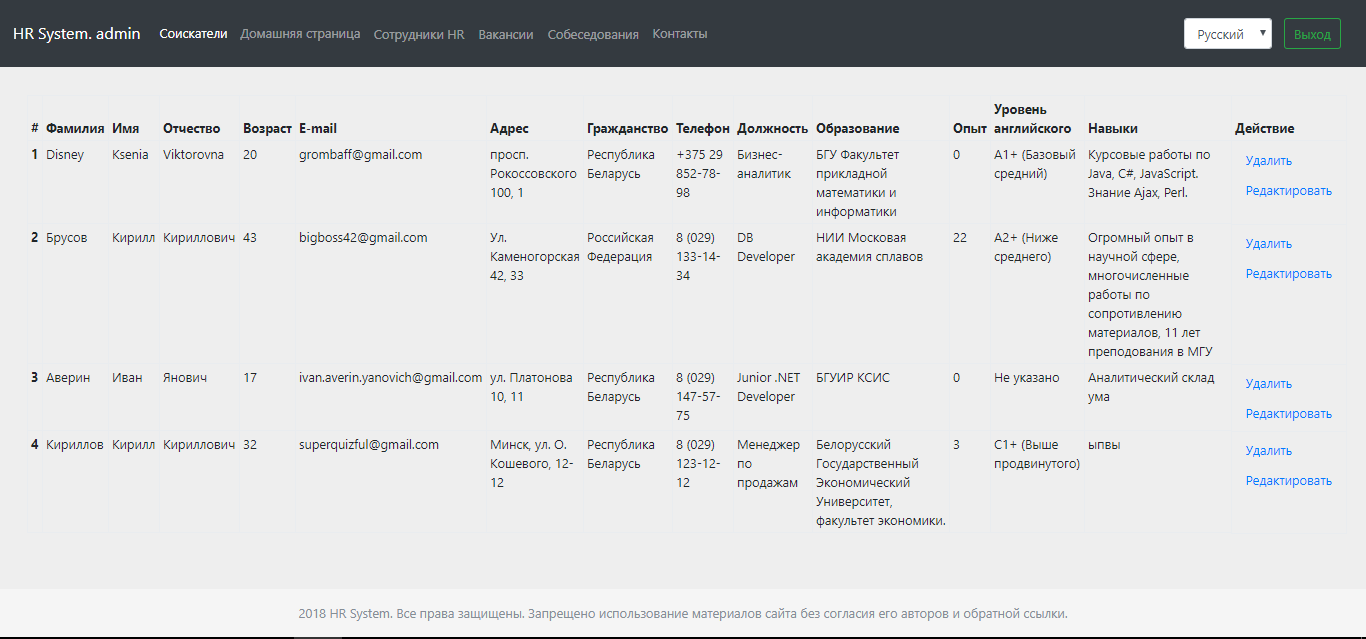


Рисунок 9.30 – Просмотр всех кандидатов администратором

В колонке «Действие» напротив каждого кандидата находятся кнопки «Удалить» и «Редактировать», которые выполняют соответствующие функции для соответствующего кандидата.

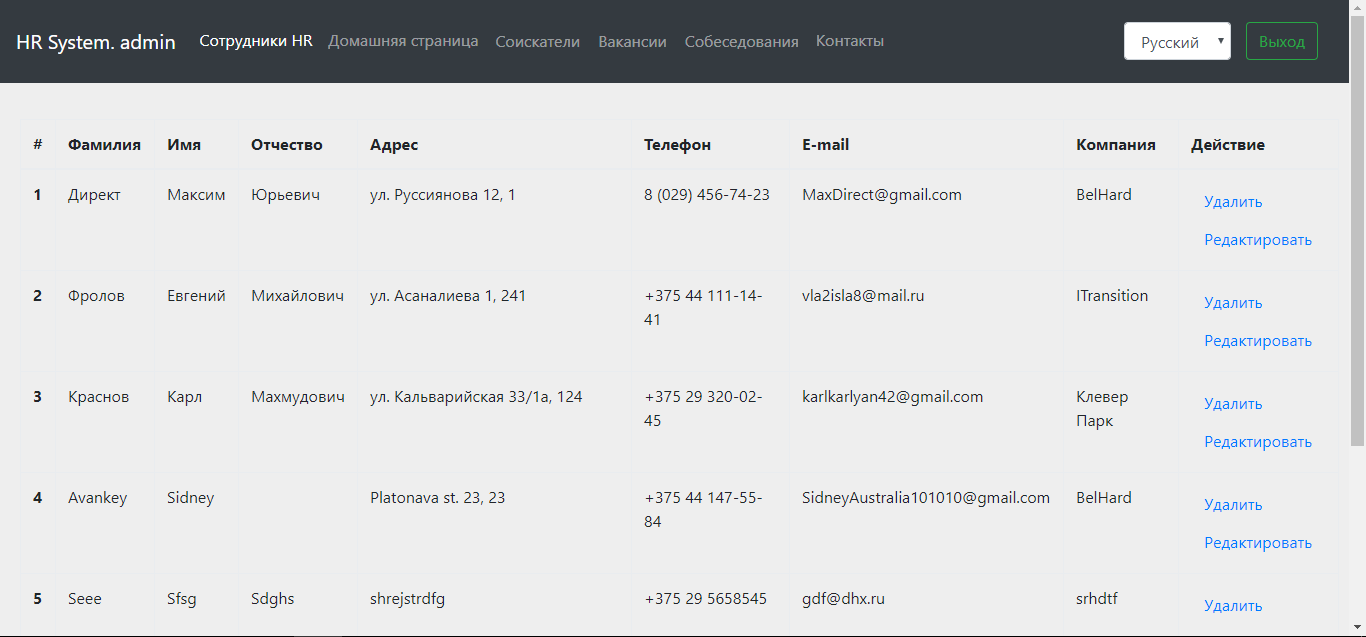


Рисунок 9.31 – Просмотр всех сотрудников HR администратором

В колонке «Действие» напротив каждого сотрудника находятся кнопки «Удалить» и «Редактировать», которые выполняют соответствующие функции для соответствующего сотрудника HR. Это же относится и к вакансиям (рисунок 9.32) и к собеседованиям (рисунок 9.33).

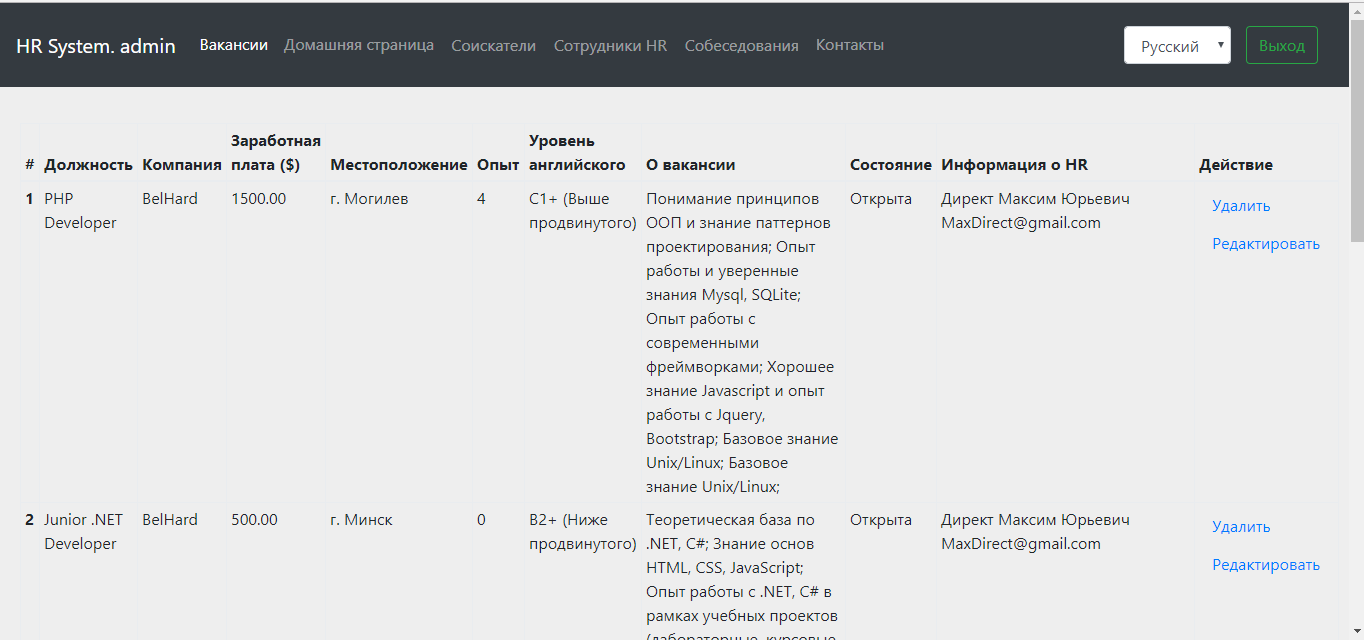


Рисунок 9.32 – Просмотр всех вакансий администратором



Рисунок 9.33 – Просмотр всех собеседований администратором

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта было разработано приложение, позволяющее в онлайн режиме автоматизировать процесс поиска работы, а также поиска подходящих сотрудников для компаний.

В качестве интерфейсной части системы была реализована клиентская сторона приложения.

Применение данной архитектуры предоставляет пользователям возможность обращаться к серверу за необходимой информацией. Сервер предоставляет возможность для клиента для работы с базой данных. Он выступает в качестве посредника между клиентом и базой данных. Он принимает запросы от клиента, их обрабатывает и направляет в базу данных. Также он перенаправляет запросы назад клиенту.

Клиент имеет возможность работы с информацией, хранимой в базе данных. Такая информация представлена в виде таблиц базы данных. Клиент может просматривать, редактировать, удалять, осуществлять поиск, фильтрацию и т.д. необходимой ему информации.

Применение базы данных в качестве хранилища информации позволяет оптимально и эффективно хранить информацию, ее структурировать.

Дополнительно в результате выполнения данного курсового проекта были получены дополнительные навыки в работе с языком JAVA, UML, получены знания о различных методах построения функциональных и информационных моделей.

Для достижения цели написания данного курсового проекта было изучено множество вопросов, связанных с программированием на языке Java. Также была проанализирована и спроектирована предметная область – онлайн-система по поиску работы. Итогом проделанной работы является приложение, которое по своим основным параметрам полностью удовлетворяет установленным требованиям.

Программный продукт характеризуется своей модульностью и лаконичностью, что обеспечивает его читаемость и удобство в эксплуатации для пользователей. Данное приложение достаточно информативно, позволяет пользователю работать с множеством различных функций. Также оно отвечает такому требованию, как экономичность.

Также программа устойчива к ошибкам, которые могут быть допущены пользователем в процессе эксплуатации данного программного продукта. Также приложение имеет доступный и понятный интерфейс, который помогает пользователям быстро освоиться и легко оперировать информацией.

# **Список использованных источников**

[1] Технологии поиска работы [Электронный ресурс]. – Электронные данные. – Режим доступа: https://superbotanik.net/referati/referaty-po-menedzhmentu/referat-tehnologii-poiska-raboty

[2] Способы поиска работы [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://studfiles.net/preview/3741646/page:2/>

[3] MySQL [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.mysql.com/>

[4] Б. Эккель. Философия Java. // СПб: Питер, 2001. – 880 с.

[5] К. Ларман. Применение UML и шаблонов проектирования // М., СПб., К.: Вильямс, 2001. – 495 с.

[6] Дубейковский, В.И. Эффективное моделирование с AllFusion Process Modeler 4.1.4 и AllFusion PM / В.И. Дубейковский. – М.: Нолидж, 2013. – 284 с.

[7] Новиков, Б.А. Настройка приложений баз данных / Б.А. Новиков, Г.Р. Домбровская. – СПб.: БХВ-Петербург, 2014. – 240 с.

# **ПРИЛОЖЕНИЕ А**

# **(Обязательное)**

# **Диаграммы классов**

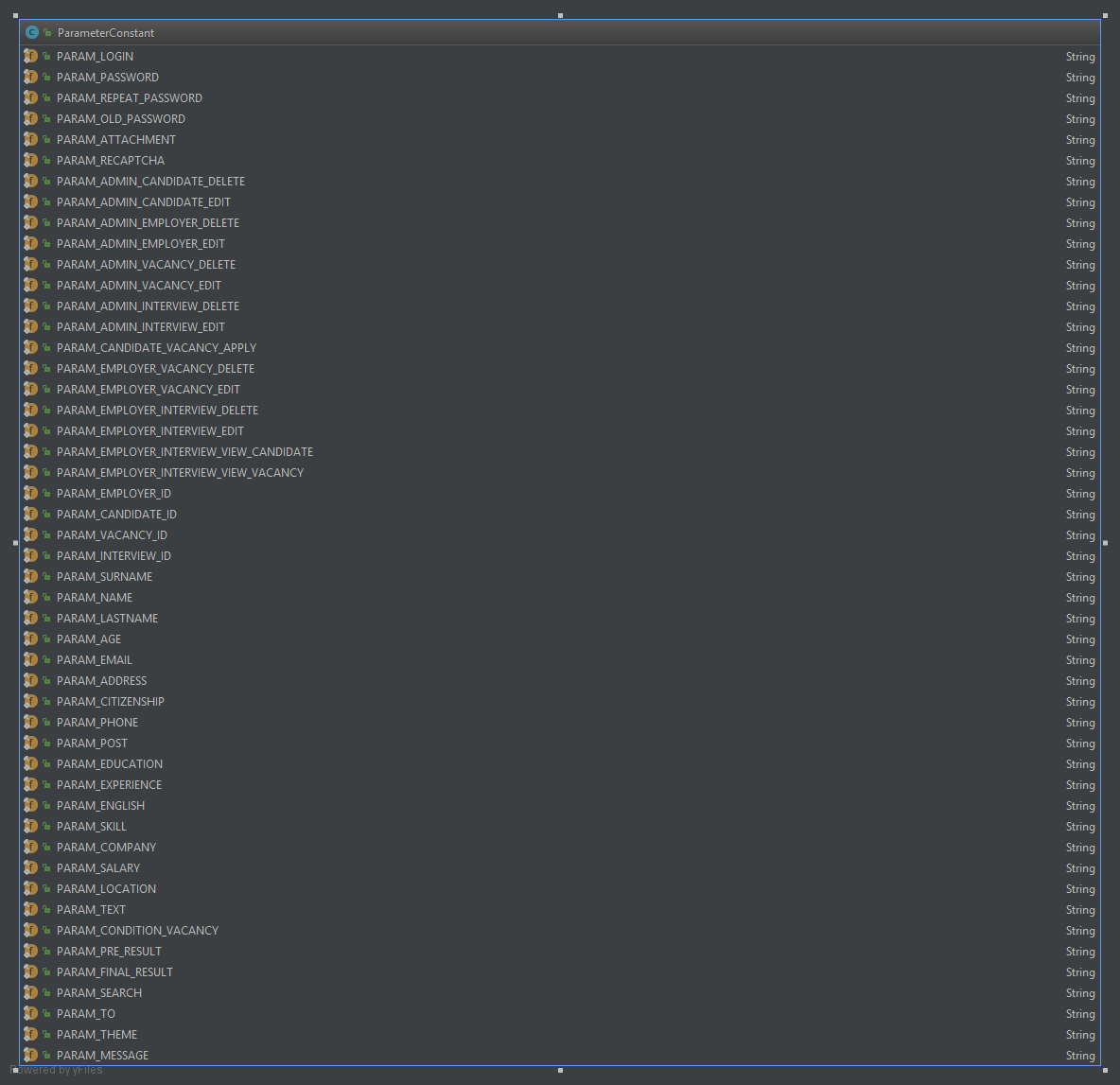


Рисунок А.1 – Диаграмма классов пакета constant

Продолжение приложения А

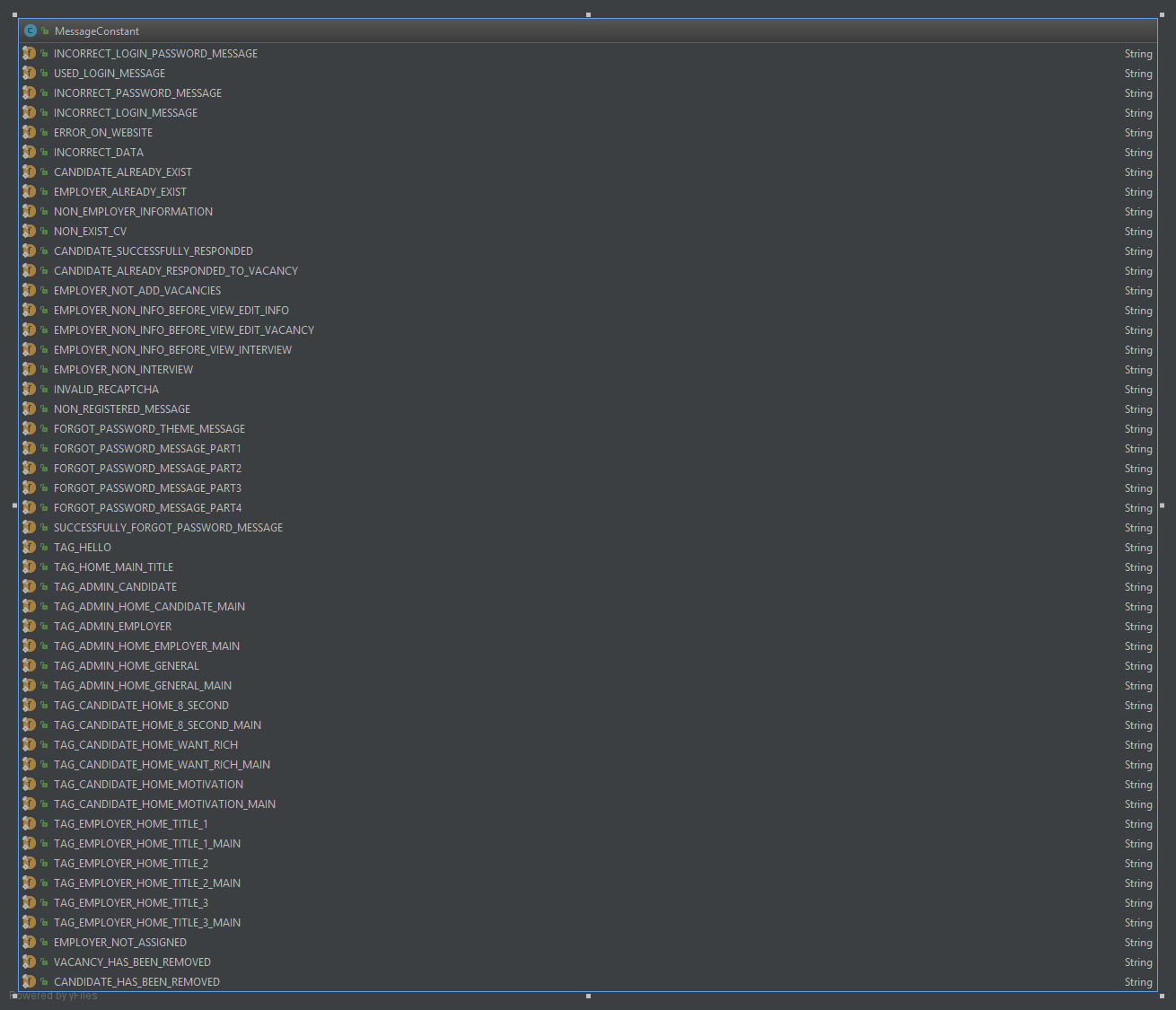


Рисунок А.2 – Диаграмма классов пакета constant

Продолжение приложения А

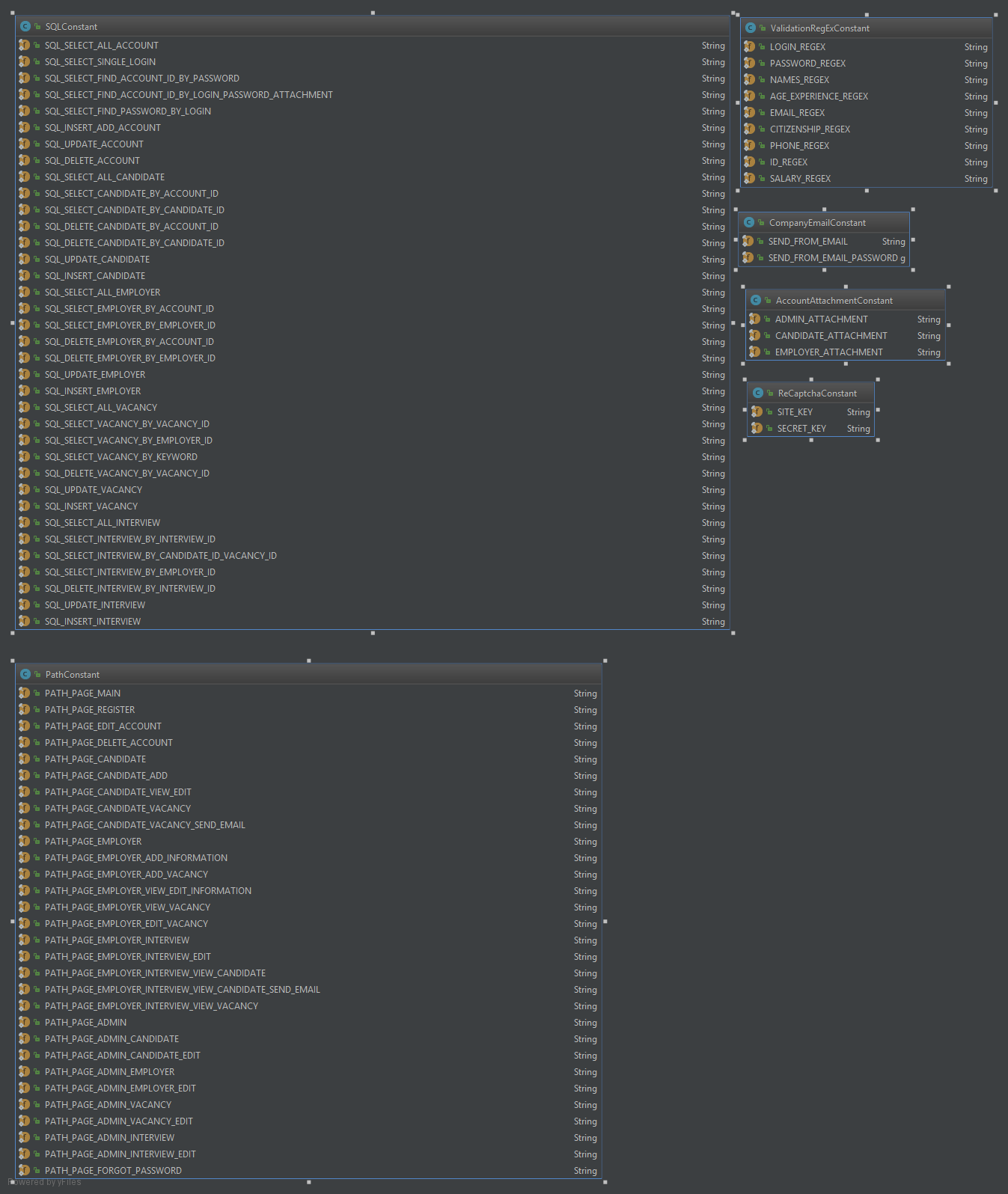


Рисунок А.3 – Диаграмма классов пакета constant

Продолжение приложения А

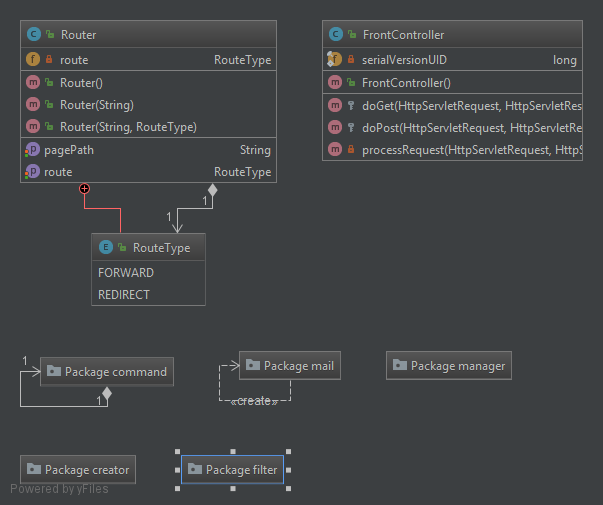


Рисунок А.4 – Диаграмма классов пакета controller

Продолжение приложения А

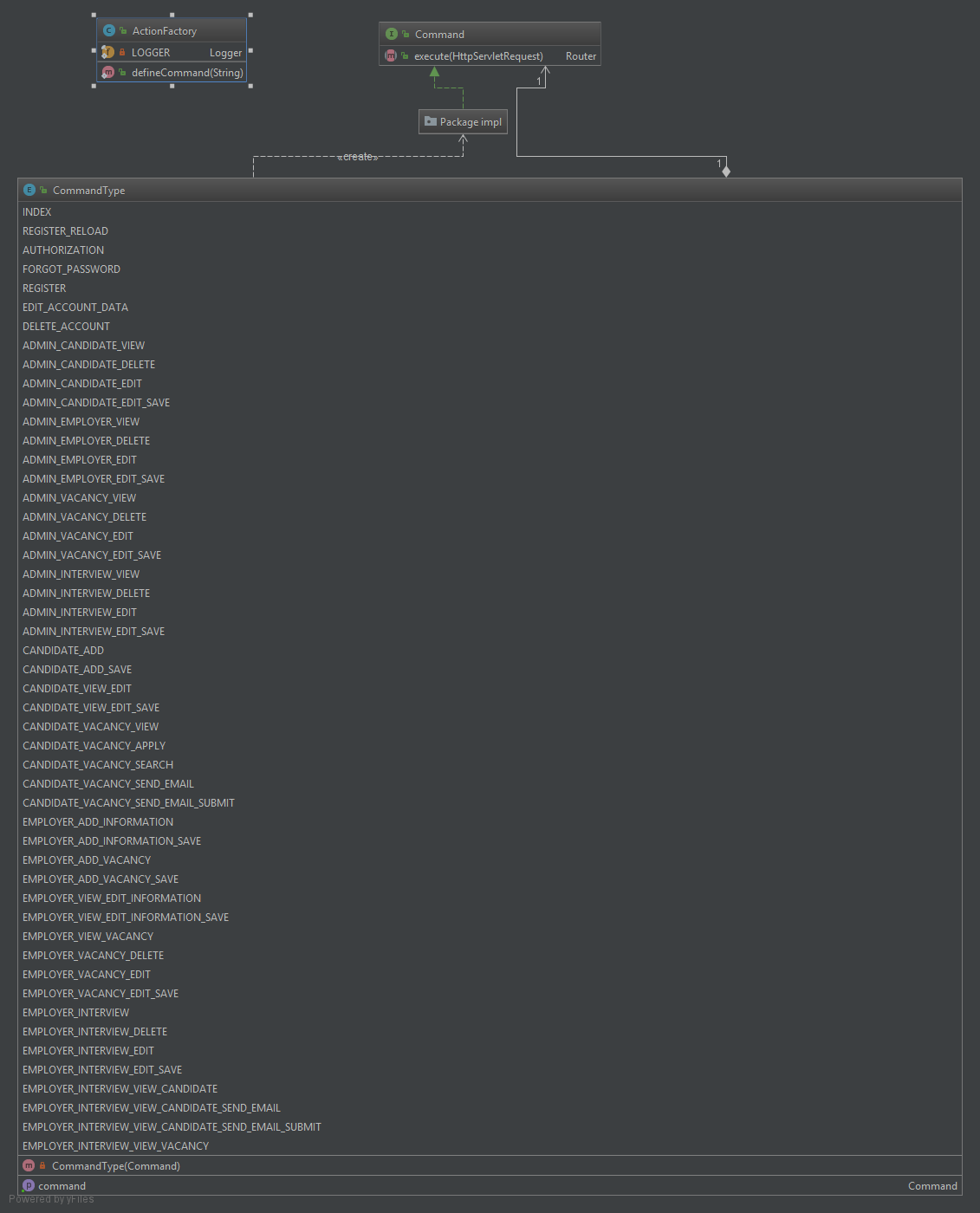


Рисунок А.5 – Диаграмма классов пакета command

Продолжение приложения А

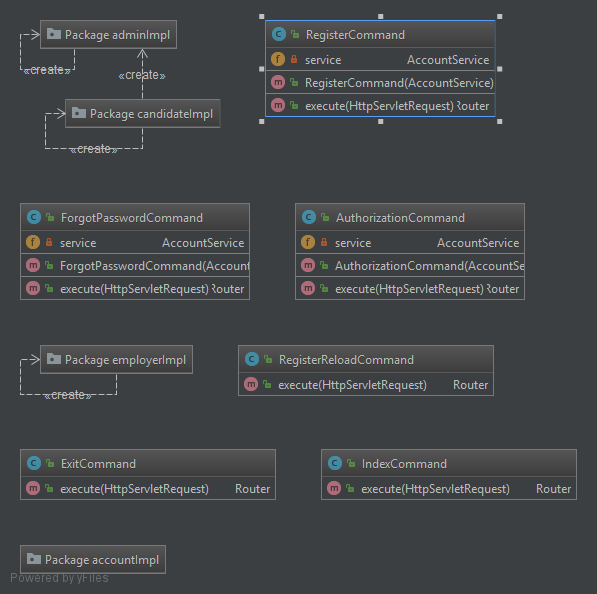


Рисунок А.6 – Диаграмма классов пакета impl

Продолжение приложения А

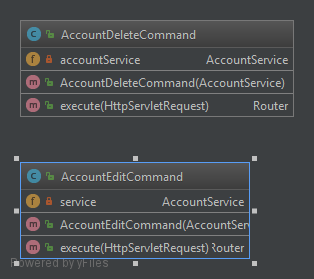


Рисунок А.7 – Диаграмма классов пакета accountimpl

Продолжение приложения А

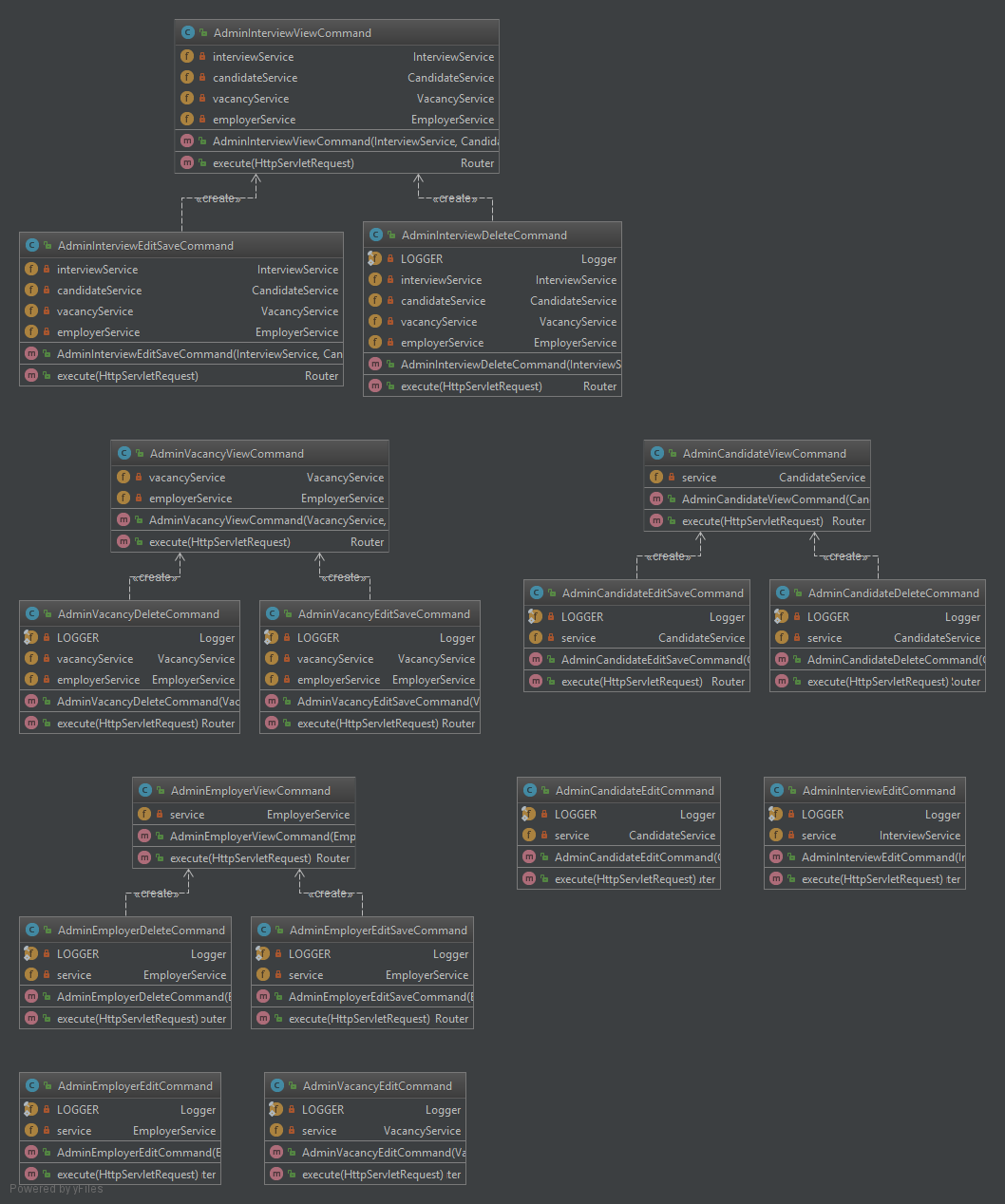


Рисунок А.8 – Диаграмма классов пакета adminimpl

Продолжение приложения А

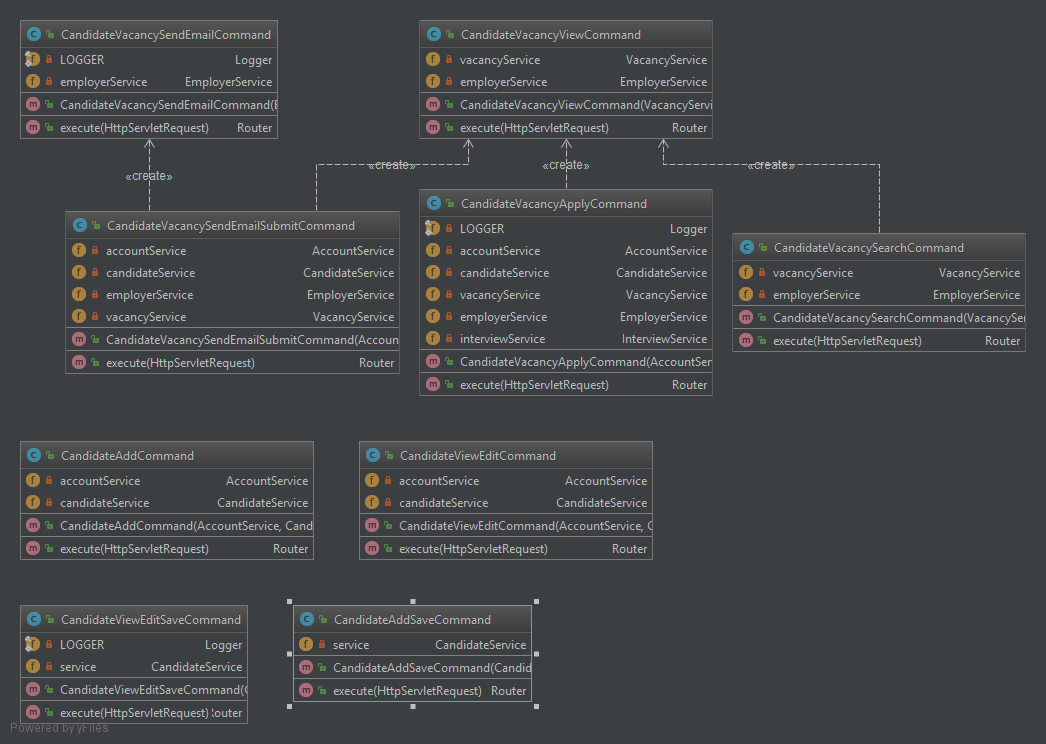


Рисунок А.9 – Диаграмма классов пакета candidateimpl

Продолжение приложения А

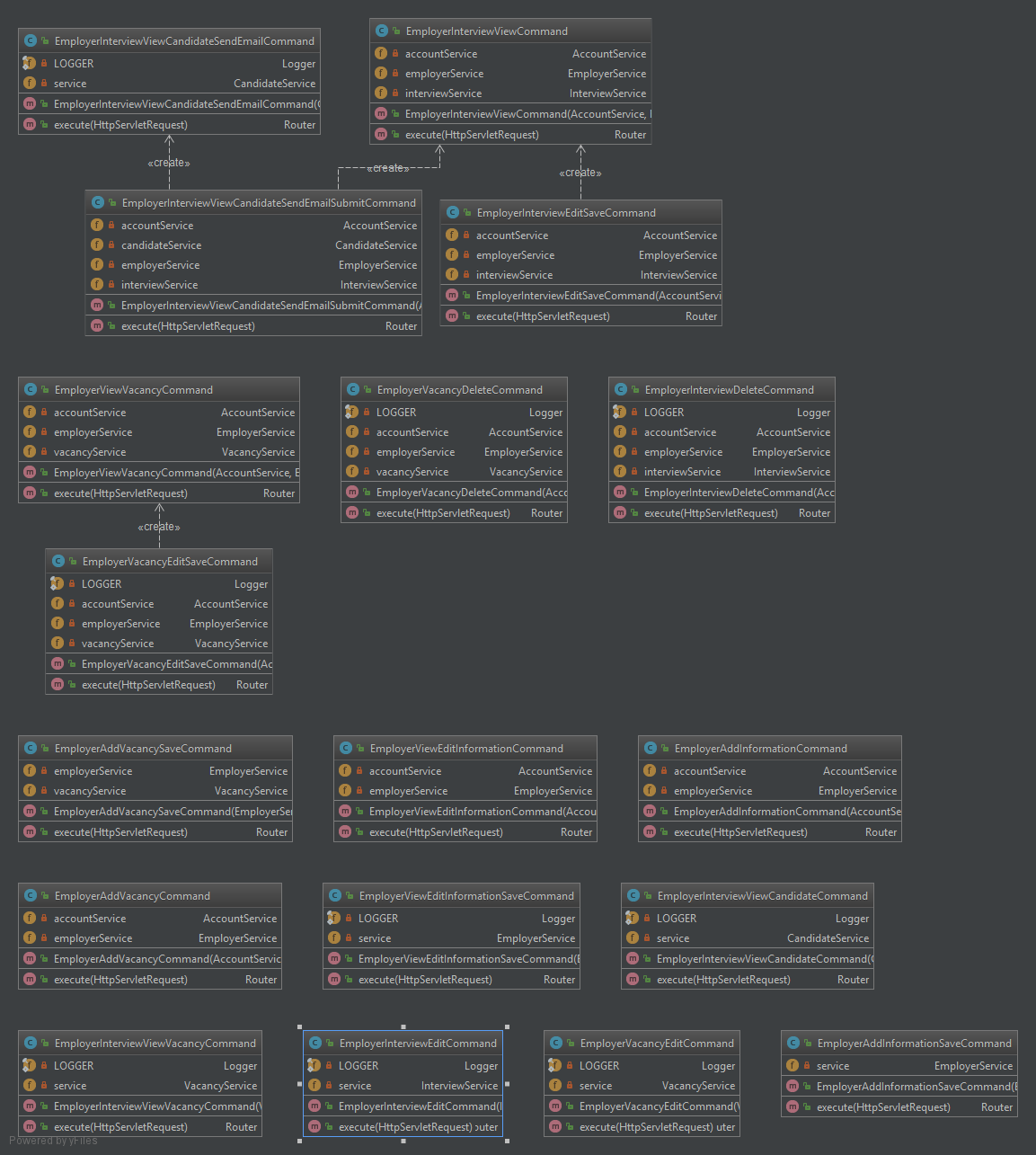


Рисунок А.10 – Диаграмма классов пакета employerimpl

Продолжение приложения А

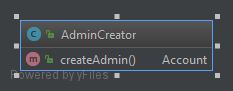


Рисунок А.11 – Диаграмма классов пакета creator

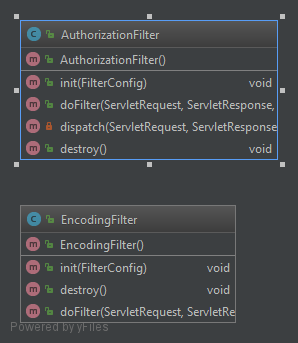


Рисунок А.12 – Диаграмма классов пакета filter

Продолжение приложения А

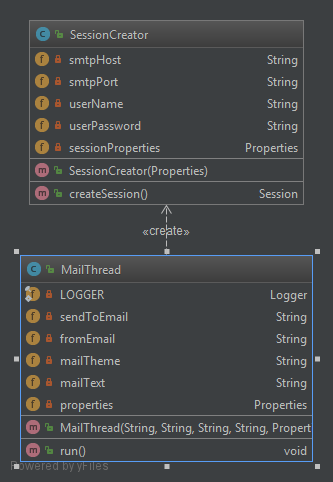


Рисунок А.13 – Диаграмма классов пакета mail

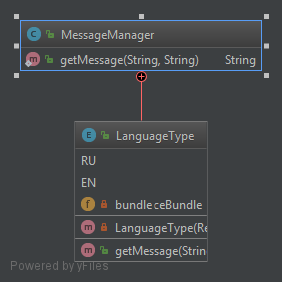


Рисунок А.14 – Диаграмма классов пакета manager

Продолжение приложения А

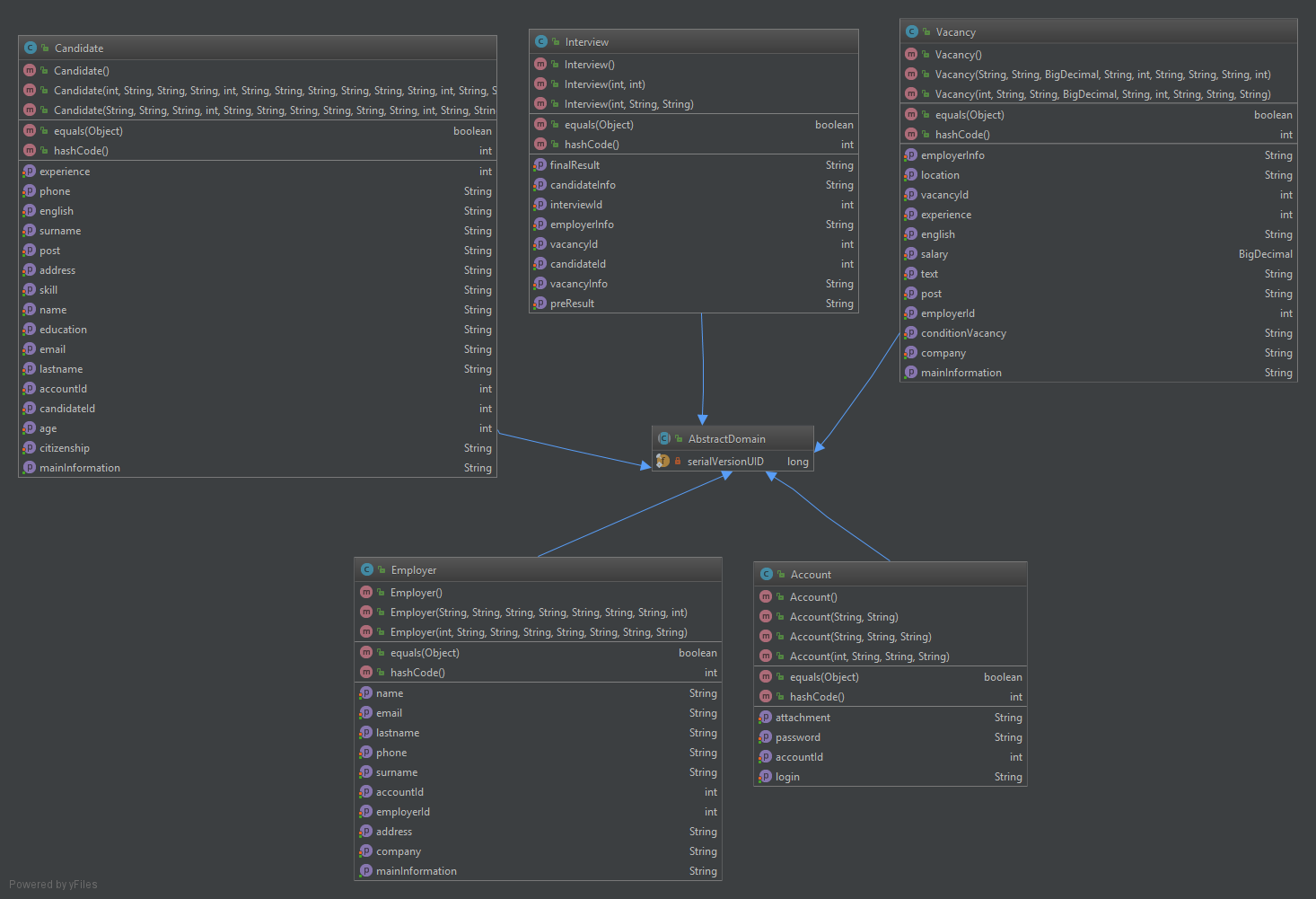


Рисунок А.15 – Диаграмма классов пакета domain

Продолжение приложения А

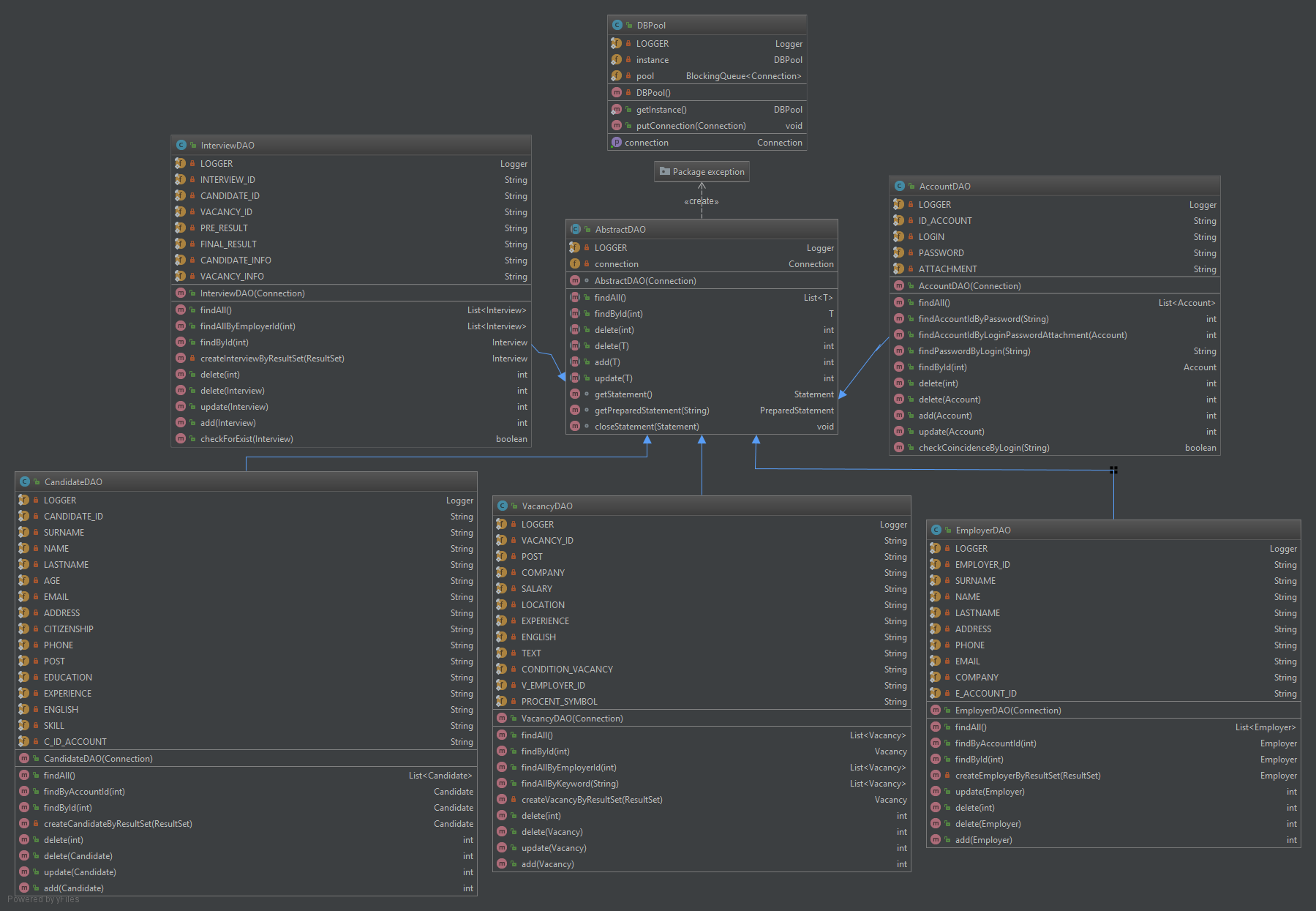


Рисунок А.16 – Диаграмма классов пакета repository

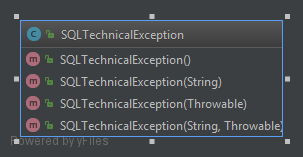


Рисунок А.17 – Диаграмма классов пакета exception

Продолжение приложения А

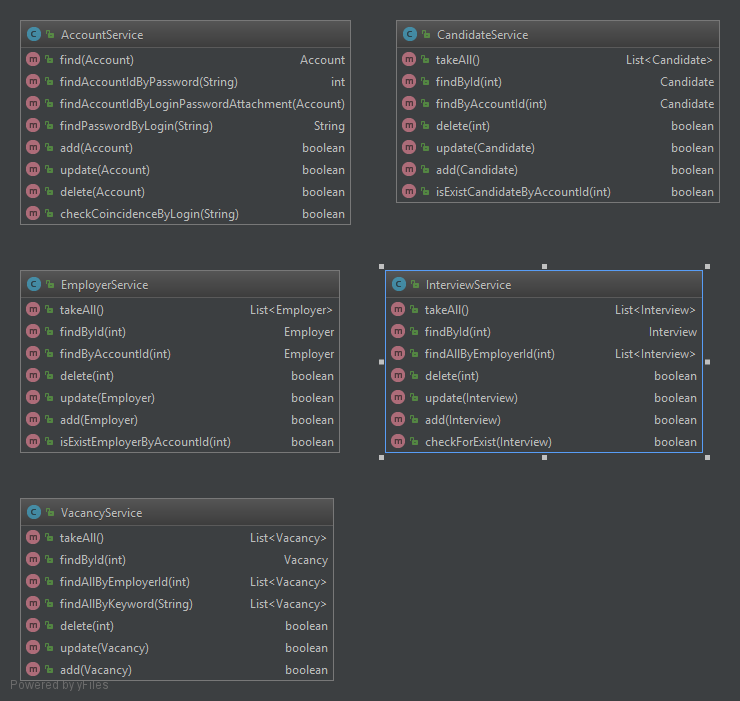


Рисунок А.18 – Диаграмма классов пакета service

Продолжение приложения А

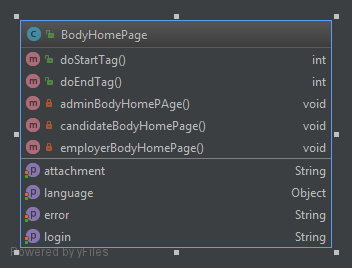


Рисунок А.19 – Диаграмма классов пакета tag

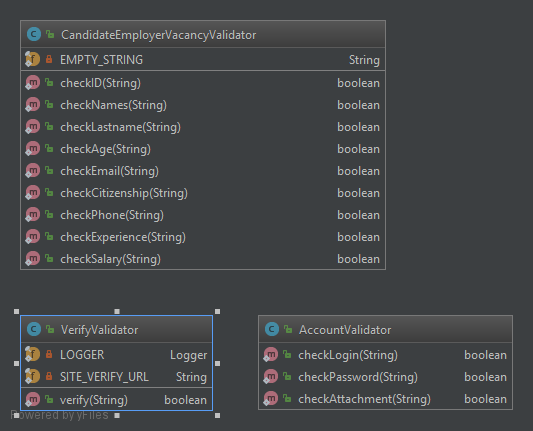


Рисунок А.20 – Диаграмма классов пакета validator

# **ПРИЛОЖЕНИЕ Б**

# **(Обязательное)**

# **Листинг основных элементов программы**

Файл FrontController.java

package by.bsuir.abmyotkashevtsov.controller;

import by.bsuir.abmyotkashevtsov.constant.PathConstant;

import by.bsuir.abmyotkashevtsov.controller.command.ActionFactory;

import by.bsuir.abmyotkashevtsov.controller.command.Command;

import by.bsuir.abmyotkashevtsov.controller.command.impl.ExitCommand;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.Optional;

@WebServlet(urlPatterns = { "/"})

public class FrontController extends HttpServlet{

private static final long serialVersionUID = 1L;

public FrontController() {

super();

}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

processRequest(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

private void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setCharacterEncoding("UTF-8");

response.setContentType("text/html; charset=UTF-8");

Optional<Command> commandOptional = ActionFactory.defineCommand(request.getParameter("command"));

Command command = commandOptional.orElse(new ExitCommand());

Продолжение приложения Б

Router router = command.execute(request);

if (router != null) {

String page = router.getPagePath();

if (router.getRoute() == Router.RouteType.FORWARD) {

RequestDispatcher dispatcher = request.getRequestDispatcher(page);

dispatcher.forward(request, response);

} else {

response.sendRedirect(request.getContextPath() + page);

}

} else {

request.getSession().invalidate();

response.sendRedirect(request.getContextPath() + PathConstant.PATH\_PAGE\_MAIN);

}

}

}

Файл ActionFactory.java

package by.bsuir.abmyotkashevtsov.controller.command;

import org.apache.logging.log4j.Level;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import java.util.Optional;

/\*\*

\* Class, to select a command from the enum 'CommandType' combing by the string value of the command that came with

\* jsp.

\*/

public class ActionFactory {

private final static Logger LOGGER = LogManager.getLogger(ActionFactory.class);

public static Optional<Command> defineCommand(String commandName) {

Optional<Command> current = Optional.empty();

if (commandName == null) {

return current;

}

try {

CommandType type = CommandType.valueOf(commandName.toUpperCase());

current = Optional.of(type.getCommand());

} catch (IllegalArgumentException e) {

LOGGER.log(Level.ERROR, "Illegal argument type of \"Command\"! Detail: " + e.getMessage());

}

return current;

}

}

Продолжение приложения Б

Файл AuthorizationCommand.java

package by.bsuir.abmyotkashevtsov.controller.command.impl;

import by.bsuir.abmyotkashevtsov.constant.MessageConstant;

import by.bsuir.abmyotkashevtsov.constant.ParameterConstant;

import by.bsuir.abmyotkashevtsov.constant.PathConstant;

import by.bsuir.abmyotkashevtsov.controller.Router;

import by.bsuir.abmyotkashevtsov.controller.command.Command;

import by.bsuir.abmyotkashevtsov.controller.creator.AdminCreator;

import by.bsuir.abmyotkashevtsov.controller.manager.MessageManager;

import by.bsuir.abmyotkashevtsov.domain.Account;

import by.bsuir.abmyotkashevtsov.service.AccountService;

import by.bsuir.abmyotkashevtsov.validator.AccountValidator;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import static by.bsuir.abmyotkashevtsov.constant.AccountAttachmentConstant.ADMIN\_ATTACHMENT;

import static by.bsuir.abmyotkashevtsov.constant.AccountAttachmentConstant.CANDIDATE\_ATTACHMENT;

import static by.bsuir.abmyotkashevtsov.constant.AccountAttachmentConstant.EMPLOYER\_ATTACHMENT;

/\*\*

\* Implementation of the "authorization" command

\*

\* <p>

\* Takes a username and password. If the data are validated, and they are available in the database

\* (i.e. the account was indeed created earlier) based on them it creates an object of type "Account" and saved

\* in the session. Otherwise, you will receive a message about invalid data.

\* </p>

\*

\* @see Command#execute(HttpServletRequest)

\*/

public class AuthorizationCommand implements Command {

private AccountService service;

/\*\*

\* Constructs and initialize commands type of 'authorization'

\*

\* @param service - instance of the service type of "Account" to access the database table "account"

\*/

public AuthorizationCommand(AccountService service) {

this.service = service;

}

@Override

public Router execute(HttpServletRequest request) {

Router router = new Router();

String login = request.getParameter(ParameterConstant.PARAM\_LOGIN);

Продолжение приложения Б

String password = request.getParameter(ParameterConstant.PARAM\_PASSWORD);

HttpSession session = request.getSession(true);

String message;

if(AccountValidator.checkLogin(login) && AccountValidator.checkPassword(password)) {

Account user = new Account(login, password);

AdminCreator creator = new AdminCreator();

Account admin = creator.createAdmin();

Account potentialAdmin = new Account(user.getLogin(), user.getPassword(), ADMIN\_ATTACHMENT);

if (admin.equals(potentialAdmin)) {

session.setAttribute("role", potentialAdmin);

router.setPagePath(PathConstant.PATH\_PAGE\_ADMIN);

router.setRoute(Router.RouteType.FORWARD);

} else {

Account role = service.find(user);

router.setRoute(Router.RouteType.FORWARD);

if (role != null) {

session.setAttribute("role", role);

if (role.getAttachment().equals(CANDIDATE\_ATTACHMENT)) {

router.setPagePath(PathConstant.PATH\_PAGE\_CANDIDATE);

} else if (role.getAttachment().equals(EMPLOYER\_ATTACHMENT)) {

router.setPagePath(PathConstant.PATH\_PAGE\_EMPLOYER);

}

} else {

Object language = request.getSession(true).getAttribute("language");

message = MessageManager.getMessage(language.toString(),

MessageConstant.INCORRECT\_LOGIN\_PASSWORD\_MESSAGE);

request.setAttribute("errorMessage", message);

router.setPagePath(PathConstant.PATH\_PAGE\_MAIN);

}

}

} else {

Object language = request.getSession(true).getAttribute("language");

message = MessageManager.getMessage(language.toString(), MessageConstant.INCORRECT\_LOGIN\_PASSWORD\_MESSAGE);

request.setAttribute("errorMessage", message);

router.setPagePath(PathConstant.PATH\_PAGE\_MAIN);

}

return router;

}

}

Продолжение приложения Б

Файл CandidateVacancyApplyCommand.java

package by.bsuir.abmyotkashevtsov.controller.command.impl.candidateImpl;

import by.bsuir.abmyotkashevtsov.constant.MessageConstant;

import by.bsuir.abmyotkashevtsov.constant.ParameterConstant;

import by.bsuir.abmyotkashevtsov.constant.PathConstant;

import by.bsuir.abmyotkashevtsov.controller.Router;

import by.bsuir.abmyotkashevtsov.controller.command.Command;

import by.bsuir.abmyotkashevtsov.controller.manager.MessageManager;

import by.bsuir.abmyotkashevtsov.domain.Account;

import by.bsuir.abmyotkashevtsov.domain.Candidate;

import by.bsuir.abmyotkashevtsov.domain.Interview;

import by.bsuir.abmyotkashevtsov.service.\*;

import org.apache.logging.log4j.Level;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

/\*\*

\* Implementation of the "candidate\_vacancy\_apply" command.

\*

\* <p>

\* Represents the implementation of clicking on the "Apply" button to the vacancy for the candidate.

\* After clicking on the button, in the database searching information about this candidate, retrieved his id

\* and created a new interview based on the candidate's id and the id of the vacancy that was apply.

\* </p>

\*

\* @see Command#execute(HttpServletRequest)

\*/

public class CandidateVacancyApplyCommand implements Command {

private final static Logger LOGGER = LogManager.getLogger(CandidateVacancyApplyCommand.class);

private AccountService accountService;

private CandidateService candidateService;

private VacancyService vacancyService;

private EmployerService employerService;

private InterviewService interviewService;

/\*\*

\* Constructs and initialize commands type of 'candidate\_vacancy\_apply'

\*

\* @param accountService - instance of the service type of "Account" to access the database table "account"

\* @param candidateService - instance of the service type of "Candidate" to access the database table "candidate"

\* @param vacancyService - instance of the service type of "Vacancy" to access the database table "vacancy"

Продолжение приложения Б

\* @param employerService - instance of the service type of "Employer" to access the database table "employer"

\* @param interviewService - instance of the service type of "Interview" to access the database table "interview"

\*/

public CandidateVacancyApplyCommand(AccountService accountService, CandidateService candidateService,

VacancyService vacancyService, EmployerService employerService,

InterviewService interviewService) {

this.accountService = accountService;

this.candidateService = candidateService;

this.vacancyService = vacancyService;

this.employerService = employerService;

this.interviewService = interviewService;

}

@Override

public Router execute(HttpServletRequest request) {

Router router = new Router(PathConstant.PATH\_PAGE\_CANDIDATE\_VACANCY, Router.RouteType.FORWARD);

HttpSession session = request.getSession(true);

Account candidateAccount = (Account)session.getAttribute("role");

int accountId = accountService.findAccountIdByLoginPasswordAttachment(candidateAccount);

if(accountId != 0) {

Candidate candidate = candidateService.findByAccountId(accountId);

if (candidate != null) {

String stringVacancyId = request.getParameter(ParameterConstant.PARAM\_CANDIDATE\_VACANCY\_APPLY);

int vacancyId = 0;

try {

vacancyId = Integer.parseInt(stringVacancyId);

} catch (NumberFormatException e) {

LOGGER.log(Level.ERROR, "Error while parsing string value vacancyId to integer! Detail: " +

e.getMessage());

}

Interview interview = new Interview(candidate.getCandidateId(), vacancyId);

if (!interviewService.checkForExist(interview)) {

if (interviewService.add(interview)) {

Command command = new CandidateVacancyViewCommand(vacancyService, employerService);

router = command.execute(request);

Object object = request.getSession(true).getAttribute("language");

String message = MessageManager.getMessage(object.toString(),

MessageConstant.CANDIDATE\_SUCCESSFULLY\_RESPONDED);

request.setAttribute("notificationMessage", message);

} else {

router.setPagePath(PathConstant.PATH\_PAGE\_CANDIDATE);

Object object = request.getSession(true).getAttribute("language");

Продолжение приложения Б

String message = MessageManager.getMessage(object.toString(),

MessageConstant.ERROR\_ON\_WEBSITE);

request.setAttribute("errorMessage", message);

}

} else {

Object object = request.getSession(true).getAttribute("language");

String message = MessageManager.getMessage(object.toString(),

MessageConstant.CANDIDATE\_ALREADY\_RESPONDED\_TO\_VACANCY);

request.setAttribute("errorMessage", message);

Command command = new CandidateVacancyViewCommand(vacancyService, employerService);

router = command.execute(request);

}

} else {

router.setPagePath(PathConstant.PATH\_PAGE\_CANDIDATE);

Object object = request.getSession(true).getAttribute("language");

String message = MessageManager.getMessage(object.toString(),

MessageConstant.NON\_EXIST\_CV);

request.setAttribute("errorMessage", message);

}

} else {

router.setPagePath(PathConstant.PATH\_PAGE\_CANDIDATE);

Object object = request.getSession(true).getAttribute("language");

String message = MessageManager.getMessage(object.toString(),

MessageConstant.ERROR\_ON\_WEBSITE);

request.setAttribute("errorMessage", message);

}

return router;

}

}

Файл EmployerInterviewViewCandidateSendEmailSubmitCommand.java

package by.bsuir.abmyotkashevtsov.controller.command.impl.employerImpl;

import by.bsuir.abmyotkashevtsov.constant.MessageConstant;

import by.bsuir.abmyotkashevtsov.constant.ParameterConstant;

import by.bsuir.abmyotkashevtsov.constant.PathConstant;

import by.bsuir.abmyotkashevtsov.controller.Router;

import by.bsuir.abmyotkashevtsov.controller.command.Command;

import by.bsuir.abmyotkashevtsov.controller.mail.MailThread;

import by.bsuir.abmyotkashevtsov.controller.manager.MessageManager;

import by.bsuir.abmyotkashevtsov.domain.Account;

import by.bsuir.abmyotkashevtsov.domain.Employer;

import by.bsuir.abmyotkashevtsov.service.AccountService;

import by.bsuir.abmyotkashevtsov.service.CandidateService;

import by.bsuir.abmyotkashevtsov.service.EmployerService;

import by.bsuir.abmyotkashevtsov.service.InterviewService;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import java.util.Objects;

import java.util.Properties;

Продолжение приложения Б

/\*\*

\* Implementation of the "employer\_interview\_view\_candidate\_send\_email\_submit" command.

\*

\* <p>

\* Accepts the data: to whom to send the letter, the subject of the letter, the message of the letter and

\* the password from the e-mail of the HR. Based on these data, it forms an object of the MailTread

\* class to send a message to Candidate.

\* </p>

\*

\* @see Command#execute(HttpServletRequest)

\*/

public class EmployerInterviewViewCandidateSendEmailSubmitCommand implements Command {

private AccountService accountService;

private CandidateService candidateService;

private EmployerService employerService;

private InterviewService interviewService;

/\*\*

\* Constructs and initialize commands type of 'employer\_interview\_view\_candidate\_send\_email\_submit'

\*

\* @param accountService - instance of the service type of "Account" to access the database table "account"

\* @param candidateService - instance of the service type of "Candidate" to access the database table "candidate"

\* @param employerService - instance of the service type of "Employer" to access the database table "employer"

\* @param interviewService - instance of the service type of "Interview" to access the database table "interview"

\*/

public EmployerInterviewViewCandidateSendEmailSubmitCommand(AccountService accountService,

CandidateService candidateService, EmployerService employerService, InterviewService interviewService) {

this.accountService = accountService;

this.candidateService = candidateService;

this.employerService = employerService;

this.interviewService = interviewService;

}

@Override

public Router execute(HttpServletRequest request) {

Router router = new Router(PathConstant.PATH\_PAGE\_EMPLOYER\_INTERVIEW\_VIEW\_CANDIDATE\_SEND\_EMAIL,

Router.RouteType.FORWARD);

String to = request.getParameter(ParameterConstant.PARAM\_TO);

String theme = request.getParameter(ParameterConstant.PARAM\_THEME);

String message = request.getParameter(ParameterConstant.PARAM\_MESSAGE);

String password = request.getParameter(ParameterConstant.PARAM\_PASSWORD);

Продолжение приложения Б

if (!Objects.equals(to, "") & !Objects.equals(theme, "") & !Objects.equals(message, "")) {

HttpSession session = request.getSession(true);

Account employerAccount = (Account) session.getAttribute("role");

int accountId = accountService.findAccountIdByLoginPasswordAttachment(employerAccount);

if (accountId != 0) {

Employer employer = employerService.findByAccountId(accountId);

Properties properties = new Properties();

properties.setProperty("mail.smtp.host", "smtp.gmail.com");

properties.setProperty("mail.smtp.port", "465");

properties.setProperty("mail.user.name", employer.getEmail());

properties.setProperty("mail.user.password", password);

MailThread mailOperator = new MailThread(to, employer.getEmail(), theme, message, properties);

mailOperator.start();

Command command = new EmployerInterviewViewCommand(accountService, employerService,

interviewService);

router = command.execute(request);

} else {

router.setPagePath(PathConstant.PATH\_PAGE\_EMPLOYER);

Object language = request.getSession(true).getAttribute("language");

String errorMessage = MessageManager.getMessage(language.toString(), MessageConstant.ERROR\_ON\_WEBSITE);

request.setAttribute("errorMessage", errorMessage);

}

} else {

Command command = new EmployerInterviewViewCandidateSendEmailCommand(candidateService);

router = command.execute(request);

Object language = request.getSession(true).getAttribute("language");

String errorMessage = MessageManager.getMessage(language.toString(), MessageConstant.INCORRECT\_DATA);

request.setAttribute("errorMessage", errorMessage);

}

return router;

}

}

# **ПРИЛОЖЕНИЕ В**

# **(Обязательное)**

# **Листинг скрипта генерации базы данных**

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema HR

-- -----------------------------------------------------

-- Соискатель вакансии регистрируется в системе. Сотрудник HR фиксирует результат предварительного собеседования с Соискателем и назначает при

-- необходимости техническое собеседование, по результатам которого и принимается решение о трудоустройстве Соискателя. Сотрудник HR размещает

-- сообщения о вакансиях и управляет ими. Администратор управляет Сотрудниками и Соискателями.

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `HR` DEFAULT CHARACTER SET utf8 ;

USE `HR` ;

-- -----------------------------------------------------

-- Table `HR`.`account`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HR`.`account` (

`idAccount` INT(5) UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'id аккаунта',

`login` VARCHAR(60) NOT NULL COMMENT 'логин',

`password` VARCHAR(60) NOT NULL COMMENT 'пароль',

`attachment` VARCHAR(10) NULL COMMENT 'приндлежность к Соискателю или Сотруднику HR',

PRIMARY KEY (`idAccount`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HR`.`candidate`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HR`.`candidate` (

`idCandidate` INT(5) UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'id соискателя',

`surname` VARCHAR(40) NOT NULL COMMENT 'фамилия',

`name` VARCHAR(40) NOT NULL COMMENT 'имя',

`lastname` VARCHAR(40) NULL COMMENT 'отчество',

`age` INT(3) NOT NULL COMMENT 'возраст',

`email` VARCHAR(100) NOT NULL COMMENT 'e-mail',

`address` VARCHAR(60) NOT NULL COMMENT 'адрес проживания',

`citizenship` VARCHAR(80) NOT NULL COMMENT 'гражданство',

`phone` VARCHAR(30) NOT NULL COMMENT 'мобильный телефон',

`post` VARCHAR(200) NULL COMMENT 'должность',

`education` VARCHAR(200) NULL COMMENT 'образование',

`experience` INT(2) DEFAULT 0 NULL COMMENT 'опыт работы',

`english` VARCHAR(50) NULL COMMENT 'уровень английского языка',

`skill` VARCHAR(255) NULL COMMENT 'основные (ключевые) навыки Соискателя',

`c\_idAccount` INT(5) UNSIGNED NOT NULL COMMENT 'id аккаунта',

INDEX `idAccount\_idx` (`c\_idAccount` ASC),

PRIMARY KEY (`idCandidate`),

CONSTRAINT `c\_idAccount`

FOREIGN KEY (`c\_idAccount`)

REFERENCES `HR`.`account` (`idAccount`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HR`.`employer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HR`.`employer` (

`idEmployer` INT(5) UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'id Сотрудника HR',

`surname` VARCHAR(30) NOT NULL COMMENT 'фамилия Сотрудника HR',

`name` VARCHAR(30) NOT NULL COMMENT 'имя Сотрудника HR',

`lastname` VARCHAR(30) NULL COMMENT 'отчество',

`address` VARCHAR(60) NOT NULL COMMENT 'адрес проживания',

`phone` VARCHAR(30) NOT NULL COMMENT 'мобильный телефон',

`email` VARCHAR(100) NOT NULL COMMENT 'E-mail',

`company` VARCHAR(100) NULL,

`e\_idAccount` INT(5) UNSIGNED NOT NULL COMMENT 'id аккаунта',

PRIMARY KEY (`idEmployer`),

INDEX `idAccount\_idx` (`e\_idAccount` ASC),

CONSTRAINT `e\_idAccount`

FOREIGN KEY (`e\_idAccount`)

REFERENCES `HR`.`account` (`idAccount`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HR`.`vacancy`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HR`.`vacancy` (

`idvacancy` INT(5) UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'id вакансии',

`post` VARCHAR(150) NOT NULL COMMENT 'должность на вакансии',

`company` VARCHAR(60) NOT NULL COMMENT 'названия компании',

`salary` DECIMAL(8, 2) NULL COMMENT 'заработная плата',

`location` VARCHAR(45) NULL COMMENT 'местоположение вакансии (например Минск)',

`experience` INT(5) DEFAULT 0 NULL COMMENT 'необходимый опыт работы',

`english` VARCHAR(50) NULL COMMENT 'необходимый уровень английского языка',

`text` VARCHAR(2000) NOT NULL COMMENT 'сам текст вакансии, необходимые навыки, условия работы и т.д.',

`conditionVacancy` VARCHAR(20) NULL COMMENT 'состояние вакансии',

`v\_idEmployer` INT(5) UNSIGNED NULL COMMENT 'id Сотрудника HR',

PRIMARY KEY (`idvacancy`),

INDEX `idEmployer\_idx` (`v\_idEmployer` ASC),

CONSTRAINT `v\_idEmployer`

FOREIGN KEY (`v\_idEmployer`)

REFERENCES `HR`.`employer` (`idEmployer`)

ON DELETE SET NULL

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `HR`.`interview`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `HR`.`interview` (

`idInterview` INT(5) UNSIGNED NOT NULL AUTO\_INCREMENT COMMENT 'id собеседования',

`i\_idCandidate` INT(5) UNSIGNED NULL COMMENT 'id Соискателя',

`i\_idvacancy` INT(5) UNSIGNED NULL COMMENT 'id Вакансии',

`preResult` VARCHAR(50) NULL COMMENT 'результат предварительного собеседования',

`finalResult` VARCHAR(50) NULL COMMENT 'результат технического (итогового) собеседования',

PRIMARY KEY (`idInterview`),

INDEX `idvacancy\_idx` (`i\_idvacancy` ASC),

CONSTRAINT `i\_idCandidate`

FOREIGN KEY (`i\_idCandidate`)

REFERENCES `HR`.`candidate` (`idCandidate`)

ON DELETE SET NULL

ON UPDATE CASCADE,

CONSTRAINT `i\_idvacancy`

FOREIGN KEY (`i\_idvacancy`)

REFERENCES `HR`.`vacancy` (`idvacancy`)

ON DELETE SET NULL

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- заполнение данными таблицы account

-- добавление аккаунтов для пяти кандидатов и четырех Сотрудников HR

INSERT INTO account (login, password, attachment) VALUES ('wolfKing1978', 'wolfCandidate1978', 'c');

INSERT INTO account (login, password, attachment) VALUES ('BigBoomTheory', 'BigBoomTheory1', 'c');

INSERT INTO account (login, password, attachment) VALUES ('KingOfTheDot', 'qwerty1234', 'c');

INSERT INTO account (login, password, attachment) VALUES ('IvanSlayer', 'IvanAverin2000', 'c');

INSERT INTO account (login, password, attachment) VALUES ('Western Cinema', 'NormKino1', 'c');

INSERT INTO account (login, password, attachment) VALUES ('Max Direct', 'TheBestHRInThisCompany1', 'e');

INSERT INTO account (login, password, attachment) VALUES ('Eugen Frolov', 'FrolovEugen1', 'e');

INSERT INTO account (login, password, attachment) VALUES ('Krasnov Karl', 'A1234567890', 'e');

INSERT INTO account (login, password, attachment) VALUES ('Avankey Sidney', 'MetroIsLove101010', 'e');

-- -----------------------------------------------------------------

-- заполнение данными таблицы candidate

-- добавление нескольких кандидатов

INSERT INTO candidate (surname, name, lastname, age, email, address, citizenship, phone, post, education, experience, english, skill, c\_idAccount)

VALUES ('Ivanov', 'Ivan', 'Ivanovich', 30, 'ivan.ivanov@gmail.com', 'ул. Притыцкого 12/1, 33', 'Республика Беларусь', '8 (029) 123-12-12',

'PHP Developer', 'БГУ Механико-математический факультет', 4, 'A0 (Абсолютный новичек)', 'Коммуникабелен, легко обучаем', 1);

INSERT INTO candidate (surname, name, lastname, age, email, address, citizenship, phone, post, education, experience, english, skill, c\_idAccount)

VALUES ('Disney', 'Ksenia', 'Viktorovna', 20, 'SmileWether@mail.ru', 'просп. Рокоссовского 100, 1', 'Республика Беларусь',

'+375 29 852-78-98', 'Бизнес-аналитик', 'БГУ Факультет прикладной математики и информатики', 0, 'A1+ (Базовый средний)', 'Курсовые работы по Java, C#, JavaScript. Знание Ajax, Perl.',

2);

INSERT INTO candidate (surname, name, lastname, age, email, address, citizenship, phone, post, education, experience, english, skill, c\_idAccount)

VALUES ('Брусов', 'Кирилл', 'Кириллович', 43, 'bigboss42@gmail.com', 'Ул. Каменогорская 42, 33', 'Российская Федерация', '8 (029) 133-14-34',

'DB Developer', 'НИИ Московая академия сплавов', 22, 'A2+ (Ниже среднего)', 'Огромный опыт в научной сфере, многочисленные работы по сопротивлению материалов, 11 лет преподования в МГУ',

3);

INSERT INTO candidate (surname, name, lastname, age, email, address, citizenship, phone, post, education, experience, english, skill, c\_idAccount)

VALUES ('Аверин', 'Иван', 'Янович', 17, 'ivan.averin.yanovich@gmail.com', 'ул. Платонова 10, 11', 'Республика Беларусь', '8 (029) 147-57-75',

'Junior .NET Developer', 'БГУИР КСИС', 0, 'Не указано', 'Аналитический склад ума', 4);

INSERT INTO candidate (surname, name, age, email, address, citizenship, phone, post, experience, english, skill, c\_idAccount) VALUES ('Сидоров',

'Марк', 23, 'MarkCreator@gmail.com', 'г. Витебск, ул. Маяковского 12/1б, 1', 'РБ', '+375 44 121-02-77', 'PHP developer', 1,

'C1 (Продвинутый)', 'Хакер по призванию', 5);

-- -----------------------------------------------------------------

-- заполнение данными таблицы employer

-- добавление нескольких Сотрудников HR

INSERT INTO employer (surname, name, lastname, address, phone, email, company, e\_idAccount) VALUES ('Директ', 'Максим', 'Юрьевич',

'ул. Руссиянова 12, 1', '8 (029) 456-74-23', 'MaxDirect@gmail.com', 'BelHard', 6);

INSERT INTO employer (surname, name, lastname, address, phone, email, company, e\_idAccount) VALUES ('Фролов', 'Евгений', 'Михайлович',

'ул. Асаналиева 1, 241', '+375 44 111-14-41', 'EugenFrolov@mail.ru', 'ITransition', 7);

INSERT INTO employer (surname, name, lastname, address, phone, email, company, e\_idAccount) VALUES ('Краснов', 'Карл', 'Махмудович',

'ул. Кальварийская 33/1а, 124', '+375 29 320-02-45', 'karlkarlyan42@gmail.com', 'Клевер Парк', 8);

INSERT INTO employer (surname, name, address, phone, email, company, e\_idAccount) VALUES ('Avankey', 'Sidney', 'Platonava st. 23, 23',

'+375 44 147-55-84', 'SidneyAustralia101010@gmail.com', 'BelHard', 9);

-- -----------------------------------------------------------------

-- заполнение данными таблицы vacancy

-- добавление нескольких вакансий

INSERT INTO vacancy (post, company, salary, location, experience, english, text, conditionVacancy, v\_idEmployer) VALUES ('PHP Developer',

'BelHard', 1500, 'г. Могилев', 4, 'C1+ (Выше продвинутого)', 'Понимание принципов ООП и знание паттернов проектирования; Опыт работы и уверенные знания Mysql, SQLite; Опыт работы с современными фреймворками; Хорошее знание Javascript и опыт работы с Jquery, Bootstrap; Базовое знание Unix/Linux; Базовое знание Unix/Linux;',

'Открыта', 1);

INSERT INTO vacancy (post, company, salary, location, english, text, conditionVacancy, v\_idEmployer) VALUES ('Junior .NET Developer', 'BelHard',

500, 'г. Минск', 'B2+ (Ниже продвинутого)', 'Теоретическая база по .NET, C#; Знание основ HTML, CSS, JavaScript; Опыт работы c .NET, C# в рамках учебных проектов (лабораторные, курсовые, диплом) приветствуется; Опыт коммерческой разработки приветствуется;',

'Открыта', 1);

INSERT INTO vacancy (post, company, salary, location, experience, english, text, conditionVacancy, v\_idEmployer) VALUES ('Бизнес-аналитик',

'ITransition', 450, 'г. Витебск', 1, 'B1+ (Выше среднего)', 'Работа с веб-проектами, составление технического задания, постановка задач инженерам программистам, представление результатов работ заказчику, сдача-приемка работ',

'Открыта', 2);

INSERT INTO vacancy (post, company, salary, location, english, text, conditionVacancy, v\_idEmployer) VALUES ('Бизнес-аналитик', 'Клевер Парк',

380, 'г. Минск', 'A2+ (Ниже среднего)', 'Умение работать в команде, Знание достаточно большого количества ПО, Знание методологии разработки программных продуктов, Активность и желание развиваться',

'Закрыта', 3);

INSERT INTO vacancy (post, company, salary, location, experience, english, text, conditionVacancy, v\_idEmployer) VALUES ('Database developer',

'BelHard', 720, 'г. Гомель', 2, 'A1+ (Базовый средний)', 'Знание языка запросов SQL (например, Transact-SQL); Знание прикладного языка программирования (например, С#); Знание технологий, обеспечивающих взаимодействие, связь приложения с базой данных (ADO.NET, Entity Framework, LINQ)',

'Открыта', 4);

-- -----------------------------------------------------------------

-- заполнение данными таблицы interview

-- добавление интервью между кандидатами на вакансию и сотрудниками HR

INSERT INTO interview (i\_idCandidate, i\_idvacancy, preResult, finalResult) VALUES (1, 1, 'Не указано', 'Не указано');

INSERT INTO interview (i\_idCandidate, i\_idvacancy, preResult, finalResult) VALUES (5, 1, 'Провал', 'Провал');

INSERT INTO interview (i\_idCandidate, i\_idvacancy, preResult, finalResult) VALUES (3, 5, 'Успешно', 'Успешно');

INSERT INTO interview (i\_idCandidate, i\_idvacancy, preResult, finalResult) VALUES (2, 3, 'Есть вопросы', 'Не указано');

INSERT INTO interview (i\_idCandidate, i\_idvacancy, preResult, finalResult) VALUES (2, 4, 'Успешно', 'Сомнительно');

INSERT INTO interview (i\_idCandidate, i\_idvacancy, preResult, finalResult) VALUES (4, 2, 'Успешно', 'Есть вопросы');

-- -----------------------------------------------------------------

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;