

Пенза, 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

П. П. Макарычев

«__» _____ 2022 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Скворцова Екатерина Владимировна гр. 18ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Информационная система автоматизации процесса набора в учебный центр ИТ-компании

Тема утверждена приказом ПГУ № 327/О от "18" _____ апреля _____ 2022 г.

3. Руководитель работы _____ к.т.н., доцент кафедры «МО и ПЭВМ» Балашова
И.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Назначение: информационная система для автоматизации процесса набора в учебный
Центр ИТ-компании.*

2) *Среда выполнения: операционная система Windows 10.*

Языковые средства разработки: Java 11.

Инструментальные средства: IntelliJ IDEA, PostgreSQL, DBeaver, Camunda Platform.

Технологии разработки: ООП.

3) *Основные функции приложения:*

а) отправка заявок в учебный центр;

б) отмена заявки в учебный центр;

в) назначение кандидату собеседования;

г) проведение собеседования;

д) получение xls-отчета о наборе в учебный центр.

5. Перечень подлежащих разработке вопросов:

1. Анализ предметной области и постановка задачи;
2. Проектирование;
3. Реализация;
4. Контроль качества программного обеспечения;
5. План разработки бюджета;
6. Оформление пояснительной записки.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	13.04.2022 – 26.04.2022	
2	Проектирование	25%	27.04.2022 – 17.05.2022	
3	Реализация	35%	18.05.2022 – 31.05.2022	
4	Контроль качества программного обеспечения	10%	01.06.2022 – 07.06.2022	
5	План разработки бюджета	5%	08.06.2022 – 09.06.2022	
6	Оформление пояснительной записки	15%	10.06.2022 – 16.06.2022	

Дата выдачи задания «22» _____ декабря _____ 2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 89 листов, 26 рисунков, 4 таблицы, 18 использованных источников и 3 приложения.

CRM-СИСТЕМА, БИЗНЕС-ПРОЦЕСС, BPMN, CAMUNDA PLATFORM, СУБД, POSTGRESQL, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, DOCKER.

Объект исследования – CRM-системы для управления деятельностью учебных центров.

Цель выпускной квалификационной работы – разработать информационную систему с возможностью автоматизации процесса набора в учебный центр ИТ-компании.

Технологии разработки – разработка информационной системы проводилась на языке программирования Java версии 11 с использованием фреймворка Spring, библиотеки liquibase как средства для миграции и версионирования баз данных и библиотеки Apache POI для генерации xlsx-документов. В качестве платформы для разработки бизнес-процессов была выбрана Camunda Platform. В качестве СУБД была выбрана PostgreSQL. В качестве инструмента для работы с базами данных был выбран DBeaver. В качестве средства сборки проекта был выбран Maven. В качестве среды разработки была выбрана IDE IntelliJ IDEA.

Среда исполнения – Windows 10.

Результаты работы – в ходе выполнения выпускной квалификационной работы была разработана информационная система с возможностью автоматизации процесса набора в учебный центр ИТ-компании. Все требования к разработке информационной системы соблюдены.

					ПГУ 09.03.04 – 8ВР182.18 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Информационная система автоматизации процесса набора в учебный центр ИТ-компании. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Скворцова Е. В						
Пров.		Балашова И. Ю					4	89
						Группа 18ВП2		
Н. Контр.		Такташкин Д. Е						
Утв.								

Содержание

Введение.....	7
1 Анализ предметной области и требований к информационной системе автоматизации набора в учебный центр ИТ-компании	9
1.1 Понятие CRM-систем	9
1.2 Применение CRM-систем в организации образовательного процесса	11
1.3 Понятие бизнес-процесса	11
1.4 Тестирование бизнес-процессов.....	13
1.5 Платформы для разработки бизнес-процессов	14
1.6 Исследование CRM-систем для управления деятельностью учебного центра	18
1.6.1 Обзор программных продуктов аналогичного назначения	18
1.6.2 Сравнительный анализ программных продуктов-аналогов	20
1.7 Постановка задачи на разработку информационной системы автоматизации процесса набора в учебный центр ИТ-компании	24
1.8 Анализ функциональных требований к разрабатываемой информационной системе	24
1.9 Технологии и средства разработки.....	25
2 Проектирование информационной системы автоматизации процесса набора в учебный центр ИТ-компании	27
2.1 Базовая архитектура информационной системы	27
2.2 Проектирование структуры данных информационной системы	27
2.2.1 Разработка концептуальной модели базы данных.....	27
2.2.2 Разработка логической модели базы данных	29
2.2.3 Разработка физической модели базы данных	30
2.3 Проектирование информационной системы	30

2.3.1 Диаграмма бизнес-процесса.....	30
2.3.2 Диаграмма классов информационной системы	33
2.3.3 Проектирование REST API	34
3 Реализация информационной системы автоматизации набора в учебный центр ИТ-компании.....	37
3.1 Структура базы данных	37
3.2 Диаграмма компонентов информационной системы	38
3.3 Диаграмма развертывания информационной системы	39
3.4 Реализация генерации отчетов.....	42
4 Контроль качества информационной системы автоматизации набора в учебный центр ИТ-компании.....	45
4.1 Тестирование	45
4.1.1 Диаграмма состояний и переходов	45
4.1.2 Модульное тестирование.....	46
4.2 Расчет метрик кода.....	48
4.3 Планирование и оценка бюджета	54
Заключение	57
Список использованных источников	59
Приложение А. Глоссарий	61
Приложение Б. Скрипты для создания объектов базы данных и наполнения базы данных	64
Приложение В. Исходные коды приложения	69

Введение

В наше время все больше ИТ-компаний открывают наборы в учебные центры подготовки специалистов. Основная цель – обучить потенциальных сотрудников технологиям, которые используются в компании, и принять на работу наиболее многообещающих и заинтересованных в трудоустройстве студентов. Кроме того, учебные центры играют не последнюю роль в продвижении компании на рынке.

Обязательный минимум шагов для организации набора в учебный центр состоит из сбора и хранения заявок, продвижения заявок по бизнес-процессу и сохранению информации о изменениях статусов заявок. Все это можно делать вручную, однако для компании этот вариант будет растратой трудовых ресурсов. Вместо того, чтобы каждый набор тратить время и денежные средства на обработку заявок вручную гораздо рациональнее разработать или приобрести информационную систему для автоматизации обработки заявок и использовать ее необходимое количество раз.

Информационная система по автоматизации процесса набора в учебный центр ИТ-компаний является подсистемой CRM-систем (сокращение от англ. Customer Relationship Management, т. е. система управления взаимоотношениями с клиентами) по управлению учебными центрами. Существует большое количество уже готовых решений, однако большинство из них являются универсальными для учебных центров различных направлений: музыкальные школы, спортивные секции и школы изучения иностранных языков. Соответственно, они не поддерживают функции, специфичные для обучения, организованного ИТ-компаниями, а также содержат функционал, неиспользуемый в данной сфере. Кроме того, большинство представленных на рынке решений имеют проприетарную лицензию.

Целью выпускной квалификационной работы является разработка информационной системы автоматизации процесса набора в учебный центр

ИТ-компании. Разработка включает в себя анализ требований, проектирование, реализацию и тестирование программного обеспечения.

1 Анализ предметной области и требований к информационной системе автоматизации набора в учебный центр ИТ-компании

1.1 Понятие CRM-систем

Аббревиатура CRM расшифровывается как «Customer Relationship Management», т. е. управление отношениями с клиентами. Сама идея управления отношениями с клиентами возникла в начале 1970-х годов. Первая CRM-система была разработана американской компанией Siebel CRM Systems в 1993 году, и спустя несколько лет CRM-системы обрели популярность благодаря продуктам компаний Siebel CRM Systems, Gartner и IBM. Следом свои CRM-системы выпустили и такие «гиганты» индустрии, как SAP SE и Oracle. В России же CRM-системы нашли свое распространение гораздо позднее.

В зарубежной и отечественной литературе содержится огромное множество определений CRM-систем. Наиболее полное определение CRM-систем дали польские исследователи Милан К., Скопур А. и Савета В. Так, CRM – это прежде всего понятие, которое ставит пользователя в центр деятельности компании с целью максимизации прибыли, которая достигается путём увеличения количества клиентов, а также их удержания, и включает в себя определение важных для пользователей ценностей, автоматизацию бизнес-процессов для повышения эффективности продаж в области маркетинга и обслуживания [1].

В [2] приведена классификация CRM-систем в соответствии с их основными функциями:

- операционные CRM-системы нацелены на упрощение работы с клиентами;
- аналитические CRM-системы предназначены для анализа текущей ситуации и дальнейшего принятия решений на основе сбора и анализа данных;

– комбинированные CRM-системы, соответственно, включают в себя элементы вышеперечисленных CRM-систем.

Несмотря на то, что первая CRM-система была разработана в начале девяностых годов прошлого века, до сих пор не существует стандартов по компонентам, из которых должна состоять CRM-система, или инструментам, которые должна содержать технология CRM. В [3] перечислены основные инструменты, которые может включать в себя технология управления взаимоотношения с клиентами:

- сбор единой базы клиентов;
- сбор истории отношений с клиентами;
- обмен сведениями между отделами компании без потери информации;
- автоматизация бизнес-процессов;
- интеграция бизнес-процессов;
- извлечение аналитических отчетов;
- анализ эффективности маркетинговых исследований;
- прогноз продаж;
- анализ удовлетворенности клиентов, фиксирование и разбор претензий;
- накопление и увеличение багажа знаний компании.

Поскольку CRM-системы являются инструментом повышения прибыли компаний, можно определить следующие критерии, которым должна соответствовать эффективная CRM-система [4]:

- оптимизация трудовых ресурсов;
- быстрый вход новых сотрудников в рабочий процесс;
- снижение риска падения продаж;
- анализ базы клиентов с формированием выборки по платежеспособности.

Так, можно сделать вывод, что основное назначение CRM-систем – это повышение прибыли компаний, которое достигается за счет

клиентоориентированности и автоматизации бизнес-процессов работы с клиентами.

1.2 Применение CRM-систем в организации образовательного процесса

Несмотря на то, что CRM-системы появились почти тридцать лет назад, их использование, и соответственно, разработка до сих пор являются востребованными на рынке. Так, в [5] рассмотрены преимущества использования CRM-систем в образовательных учреждениях, а также приведен перечень университетов, использующих CRM-системы, разработанные компаниями «Бизнес-навигатор», Naumen и Terrasoft. В этот перечень входят следующие передовые высшие учебные заведения: МГТУ имени Н. Э. Баумана, Высшая школа экономики и многие другие. В [6] описан процесс проектирования CRM-системы для оптимизации работы приемной кампании Сыктывкарского государственного университета. В [7] приведено описание разработанной конфигурации «Учебный центр» на платформе «1С: Предприятие 8.2» для автоматизации бизнес-процессов коммерческого учебного образовательного заведения.

Практически во всех работах отечественных авторов делается упор на использование CRM-систем для высших учебных заведений. В [7] описана CRM-система для использования в учебных центрах, но по сути, данное решение является надстройкой платформы «1С: Предприятие 8.2».

1.3 Понятие бизнес-процесса

При рассмотрении понятия CRM-систем можно заметить, насколько часто фигурирует термин бизнес-процесс. Можно дать следующее определение данному понятию. Бизнес-процессом является обоснованный, взаимосвязанный и последовательный набор действий, который, используя ресурсы фирмы, создает ценность и выдает оперативный и правильный результат клиенту [8].

В рамках разрабатываемой информационной системы бизнес-процессом будет являться процесс обработки заявок, поступающих от кандидатов в учебный центр ИТ-компании.

Существует огромное множество способов описания бизнес-процессов, например, такие как UML, BPEL и BPMN [9]. С. А. Потрясаев в [10] в качестве способа моделирования бизнес-процессов рассматривает именно нотацию BPMN, поскольку она позволяет удобно моделировать бизнес-процессы, а также понятна и техническим специалистам, и бизнес-пользователям.

В нотации BPMN существует несколько видов элементов, наиболее основные из них – это задачи, события и шлюзы.

Шлюзы в нотации BPMN обозначают условия, в зависимости от выполнения которых происходит продвижение заявки по бизнес-процессу. Шлюзы отображаются ромбами, в центре которых находятся пиктограммы, присущие конкретному виду шлюзов.

Задачи по своей сути определяют единицу работы. На диаграмме они отображаются прямоугольником с закругленными краями, в верхнем левом углу которого – пиктограмма, присущая определенному виду задач. Основные виды задач:

- сервисная – задача, являющаяся программной функцией, вызываемой автоматически при наступлении задачи, пиктограмма – шестеренка;
- пользовательская – задача, выполняемая человеком, пиктограмма – человек;
- отправка сообщения, пиктограмма – черный конверт;
- получение сообщения, пиктограмма – белый конверт.

События в BPMN отображаются кругами, в центре которого – пиктограмма, присущая определенному типу сообщений. Если пиктограмма отсутствует, то событие является пустым.

Виды событий в BPMN:

- начальное – определяют начало процесса;
- промежуточное – возникают во время работы процесса;
- конечное – определяют окончание процесса.

Кроме того, события могут быть привязаны к задачам, описанным выше, а также иметь свой тип. Типы событий, использованные в спроектированной диаграмме бизнес-процесса:

- сообщение – событие, порождающее или перехватывающее сообщение, пиктограмма – конверт;
- ошибка – событие, порождающее или перехватывающее ошибку, пиктограмма – молния;
- таймер – событие, срабатывающее по условию таймера, пиктограмма – часы.

Благодаря описанным выше возможностям проектирования бизнес-процессов в нотации BPMN является удобным и понятным не только для технических специалистов, но и для бизнес-пользователей. Исходя из вышесказанного, в качестве способа моделирования бизнес-процесса в разрабатываемой информационной системы была выбрана нотация BPMN.

1.4 Тестирование бизнес-процессов

Можно провести тестирование бизнес-процессов разработанной информационной системы автоматизации процесса набора в учебный центр ИТ-компании. Тестирование бизнес-процессов – это процедура исследования и испытания бизнес-процессов. Назначение данного вида тестирования состоит в следующем:

- демонстрация разработчикам, что спроектированный процесс соответствует предъявляемым требованиям;
- выявление ситуаций, в которых поведение процесса не соответствует согласованным требованиям.

Тестирование бизнес-процессов сводится к составлению тестовых сценариев на основе требований и проведению проверки процессов по этим

сценариям. В следствие того, что тестирование бизнес-процесса в целом является объемной задачей, зачастую этот процесс разбивают на модули и создают чек-листы для каждого отдельно взятого модуля. Типичными вариантами такой логики является создание отдельных чек-листов для:

- различных уровней функционального тестирования бизнес-процесса;
- отдельных функций производственного процесса;
- отдельных требований;
- типичных сценариев;
- функций процесса, которые наиболее подвержены рискам.

Для демонстрации принципов построения чек-листов можно воспользоваться логикой разбиения функций процесса по степени их важности на следующие категории:

- базовые функции, без которых производственный процесс не может состояться;
- дополнительные функции, выполняемые исполнителями процесса в их повседневной работе, оказывающие дополнительный эффект на эффективность и качество производимого продукта.

1.5 Платформы для разработки бизнес-процессов

В свое время одной из наиболее известных платформ для разработки бизнес-процессов была платформа IBM BPM [11]. Однако она имеет существенные недостатки:

- последняя выпущенная версия IBM BPM поддерживает в качестве языка разработки Java версии 8, которая была выпущена в 2014 году;
- проекты, разработанные на IBM BPM, разворачиваются на сервере IBM WebSphere, бесплатные версии которого очень сильно ограничены в функционале;
- проблемы с поддержкой модульного тестирования;

- проприетарная лицензия.

Альтернативой рассмотренной выше платформе для разработки бизнес-процессов является платформа Camunda Platform [12]. Она не содержит перечисленных недостатков платформы IBM BPM:

- поддержка в качестве языка разработки новых версий Java и других JVM-языков (например, Kotlin);
- поддержка модульного тестирования бизнес-процессов;
- помимо проприетарной лицензии существует бесплатная лицензия с более чем необходимым функционалом.

Camunda имеет интеграцию с фреймворком Spring, имеющего огромную популярность, развитое сообщество, а также, являющегося фактическим стандартом разработки веб-приложений на языках JVM. Кроме того, разработку приложений на данной платформе облегчают приложения, входящие в состав Camunda:

- Camunda Modeler – десктопное приложение для графического моделирования бизнес-процессов;
- Camunda Admin – веб-приложение для работы с пользователями и группами пользователей;
- Camunda Cockpit – веб-приложение для мониторинга бизнес-процессов;
- Camunda Tasklist – веб-приложение для работы с задачами бизнес-процессов.

На рисунке 1 представлен интерфейс десктопного приложения Camunda Modeler, служащего для графического моделирования бизнес-процессов. Благодаря дружественному интерфейсу данной программы ее могут использовать для работы не только технические специалисты, но и, к примеру, бизнес-аналитики.

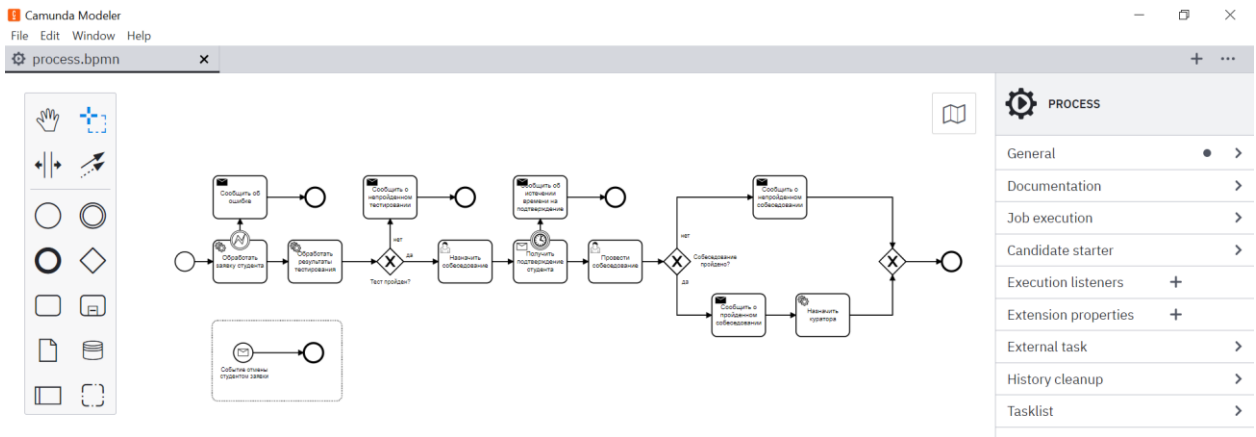


Рисунок 1 – Интерфейс приложения Camunda Modeler

На рисунке 2 представлен интерфейс веб-приложения Camunda Admin, служащего для работы с группами пользователей.

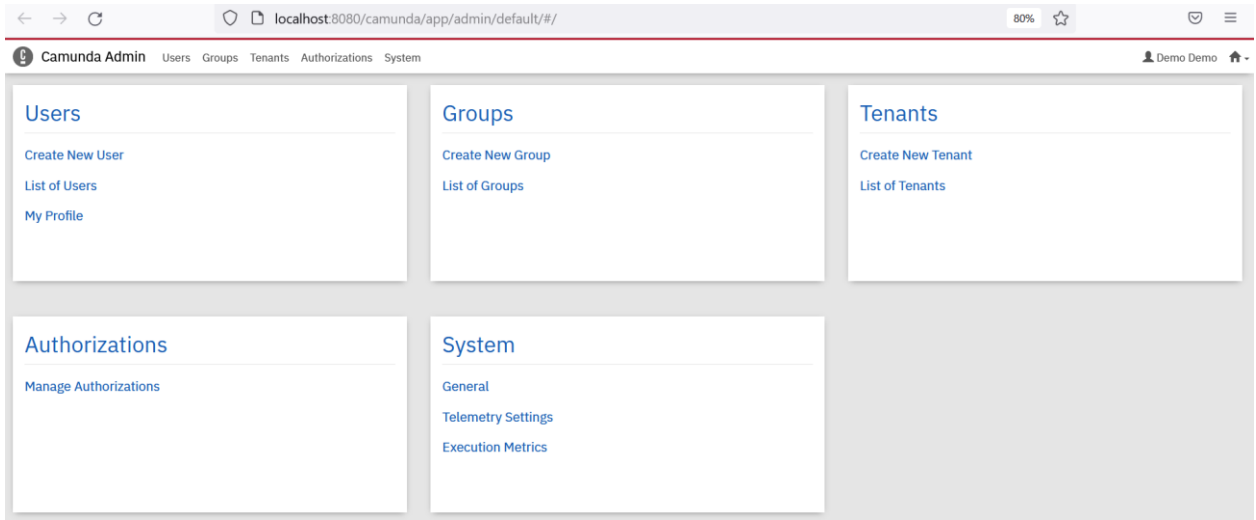


Рисунок 2 – Интерфейс приложения Camunda Admin

На рисунке 3 представлен интерфейс веб-приложения Camunda Cockpit, служащего для мониторинга работы бизнес-процессов. При помощи этого приложения можно узнать значения переменных бизнес-процесса. Кроме этого, данное приложение позволяет также увидеть задачи, в текущий момент назначенные на пользователей, и перейти по ссылке на них в приложение Camunda Tasklist.

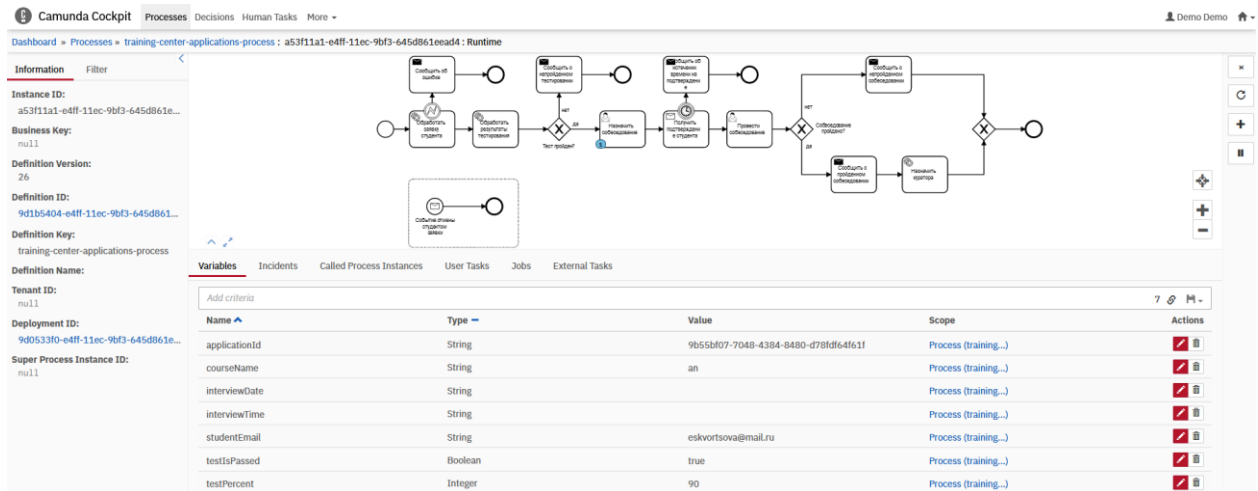


Рисунок 3 – Интерфейс приложения Camunda Cockpit

На рисунке 4 представлен интерфейс веб-приложения Camunda Tasklist, служащего для работы с пользовательскими задачами бизнес-процессов.

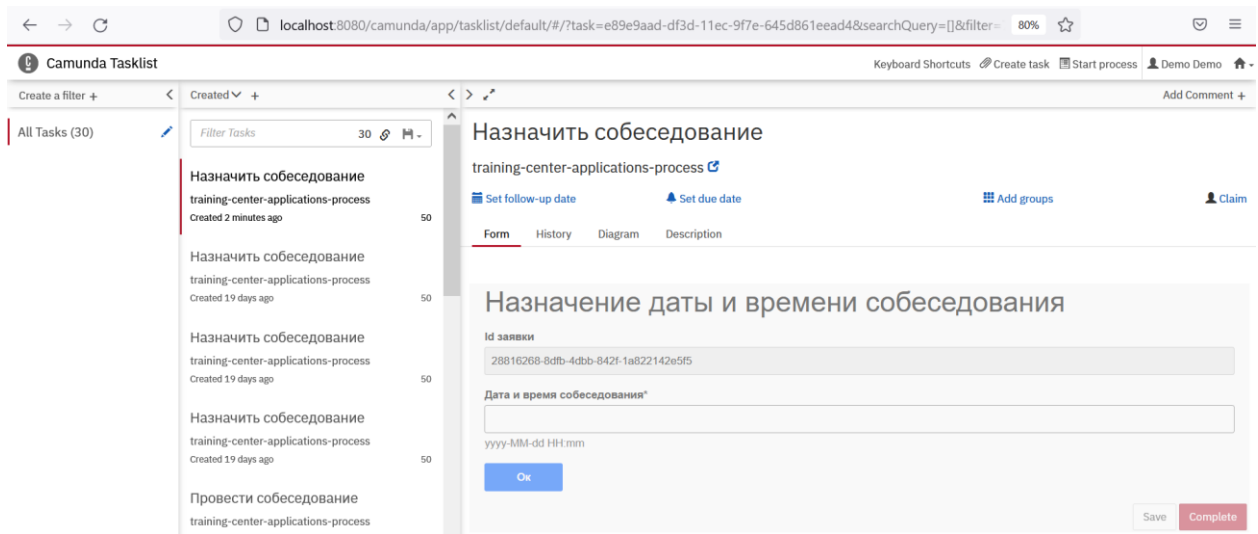


Рисунок 4 – Интерфейс приложения Camunda Tasklist

Приняв во внимание вышеперечисленные преимущества, в качестве платформы для разработки бизнес-процессов выбрана Camunda Platform.

1.6 Исследование CRM-систем для управления деятельностью учебного центра

1.6.1 Обзор программных продуктов аналогичного назначения

Проведен обзор имеющихся на рынке CRM-систем для управления учебными центрами. Основной упор сделан на функциональность, касающуюся набора студентов учебного центра.

Рассмотрим программный продукт «1С: Управление учебным центром». Согласно описанию, приведенному на интернет-странице [13], данный продукт «представляет собой комплексное решение задач управления, поддержки бизнес-процессов, учетных функций учебных центров любого профиля, вида финансирования и отраслевой направленности». Данное программное обеспечение реализует следующие функции, относящиеся к набору слушателей учебного центра:

- планирование и учет взаимодействия с физическими лицами и контрагентами;
- работа с физическими и юридическими лицами;
- регистрация заявок;
- печать и выдача свидетельств;
- предоставление и учет скидок;
- подготовка договоров и документов;
- защита персональных данных слушателей по 152-ФЗ;
- самостоятельная подача заявки на обучение клиентом;
- анализ заявок и набора.

Проанализировав предоставленные функции, можно сделать вывод, что в первую очередь программный продукт «1С: Управление учебным центром» нацелен на использование в коммерческих учебных центрах. Поэтому данный продукт не поддерживает логику набора студентов на конкурсной основе, что характерно для учебных центров ИТ-компаний, здесь же слушателями учебного центра являются все, кто записался и оплатил

выбранный курс. Кроме того, в перечне функций не указано сохранение истории взаимодействия с клиентом. Также использование данного продукта является платным.

В [14] приведено описание информационной системы управления деятельностью учебного центра, предназначенной для поддержки процессов управления деятельностью корпоративных и отраслевых учебных центров. Данная информационная система реализует следующие функции по учету слушателей:

- создание и редактирование информационной карточки слушателя;
- ведение истории общения сотрудников со слушателем;
- назначение и учет ознакомительных визитов;
- учет договоров на оказание образовательных услуг;
- учет маркетинговой информации о слушателях;
- учет результатов тестирования слушателей;
- учет истории обучения слушателя;
- учет счетов и платежей слушателей;
- учет посещаемости слушателя;
- учет отработок слушателя;
- учет пробных занятий;
- учет обучения слушателя за рубежом;
- учет баллов и скидок;
- автоматическое формирование печатных форм необходимых документов.

Данная информационная система предоставляет функцию, которая не поддерживалась рассмотренным ранее продуктом «1С: Управление учебным центром» – ведение истории взаимодействия со слушателем. Однако в этой информационной системе также нет поддержки конкурсного отбора студентов. Кроме того, в перечисленных функциях нет ничего о подаче заявки на обучение клиентом, а карточки клиентов заполняются вручную

после телефонного звонка от клиента или электронного письма. Подобный процесс набора в учебный центр сложно назвать автоматизированным, что является основным минусом данной информационной системы.

В [15] приведено описание CRM-системы для управления учебными центрами «Альфа CRM». Она универсальна для учебных центров, языковых и музыкальных школ, спортивных секций, при этом данную CRM-систему можно настроить под личные нужды. Помимо автоматизации продаж, учета и анализа финансов, «Альфа CRM» хранит всю историю взаимодействий с клиентами. Рассмотренная CRM-система не поддерживает описанную выше функциональность, специфичную для автоматизации бизнес-процессов в учебных центрах ИТ-компаний. Но благодаря настраиваемым тарифным планам, бесплатному пробному периоду и дружественному пользовательскому интерфейсу «Альфа CRM» пользуется большой популярностью для руководителей школ, кружков и секций, функционирующих на коммерческой основе.

1.6.2 Сравнительный анализ программных продуктов-аналогов

Для проведения сравнительного анализа программных продуктов-аналогов были выбраны следующие критерии:

- поддержка набора студентов в учебный центр на конкурсной основе;
- поддержка самостоятельной подачи заявок на обучение клиентами;
- поддержка ведения истории взаимодействия с клиентами;
- поддержка анализа заявок;
- бесплатность программного продукта.

В таблице 1 представлены коэффициенты значимости описанных выше критериев.

Таблица 1 – Коэффициенты значимости критериев оценки CRM-систем для управления деятельностью учебного центра

№	Критерий	Коэффициент значимости
1	Поддержка набора студентов в учебный центр на конкурсной основе	1
2	Поддержка самостоятельной подачи заявок на обучение клиентами	1
3	Поддержка ведения истории взаимодействия с клиентами	0,5
4	Поддержка анализа заявок	0,5
5	Бесплатность программного продукта	0,75

Для оценки соответствия сравниваемых программных продуктов критериям, приведенным выше, используется пятибалльная шкала, где 0 – «не соответствует» критерию, а 4 – «полностью соответствует». В таблице 2 приведено описание баллов.

Таблица 2 – Описание баллов шкалы соответствия

Балл	Описание
0	Функция не реализована, или продукт не соответствует требуемому качеству
1	Функция не реализована, но ее можно реализовать в рамках доработки продукта, или продукт в небольшой степени соответствует требуемому качеству

Продолжение таблицы 2

Балл	Описание
2	Функция реализована, но отличается от требуемой, или продукт частично соответствует требуемому качеству
3	Функция реализована, но немного отличается от требуемой, или продукт практически соответствует требуемому качеству
4	Функция реализована, полностью соответствует требуемой, или продукт полностью соответствует требуемому качеству

Для анализа были выбраны следующие программные продукты-аналоги:

- «1С: Управление учебным центром»;
- «Информационная система управления деятельностью учебного центра»;
- «Альфа CRM».

В таблице 3 представлена критериальная оценка продуктов-аналогов.

Таблица 3 – Критериальная оценка продуктов-аналогов

№	Критерий	Программный продукт №		
		1	2	3
1	Поддержка набора студентов в учебный центр на конкурсной основе	0	0	0
2	Поддержка самостоятельной подачи заявок на обучение клиентами	4	0	2

Продолжение таблицы 3

№	Критерий	Программный продукт №		
		1	2	3
3	Поддержка ведения истории взаимодействия с клиентами	0	4	3
4	Поддержка анализа заявок	4	0	0
5	Бесплатность программного продукта	0	0	0

Произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле:

$$O_{\text{пп}} = \sum_{i=1}^n (Z_i * K_i),$$

где $O_{\text{пп}}$ – интегральная оценка программного продукта;

n – количество критериев для проведения анализа программных продуктов-аналогов;

Z_i – значение выполнения критерия n ;

K_i – коэффициент значимости критерия.

В результате выполнения расчетов были получены следующие интегральные оценки сравниваемых программных продуктов:

- «1С: Управление учебным центром» – 6 баллов;
- «Информационная система управления деятельностью учебного центра» – 2 балла;
- «Альфа CRM» – 3,5 балла.

После выполнения сравнения программных продуктов аналогичного назначения, можно с уверенностью сделать вывод, что программный продукт «1С: Управление учебным центром» больше прочих соответствует заявленным критериям. Тем не менее, данный продукт не реализует наиболее важную функциональность – набор студентов в учебный центр на конкурсной основе, а кроме этого, является платным.

1.7 Постановка задачи на разработку информационной системы автоматизации процесса набора в учебный центр ИТ-компании

В рамках выпускной квалификационной работы была поставлена задача разработать информационную систему автоматизации процесса набора в учебный центр ИТ-компании. Функции, которые должна реализовывать система:

- прием заявок в учебный центр;
- перемещение заявок по бизнес-процессу;
- сохранение истории заявок;
- распределение зачисленных студентов по кураторам;
- генерация `xlsx`-документов со статистикой набора в учебный центр.

Разрабатываемая информационная система по сути будет являться связующим программным обеспечением между клиентским приложением, ориентированным на работу с пользователями-кандидатами в учебный центр, и базой данных. Информационная система в первую очередь нацелена на работу с администраторами учебного центра, и не предусматривает предоставление пользовательского интерфейса для регистрации кандидатов в учебный центр и отправки заявок.

1.8 Анализ функциональных требований к разрабатываемой информационной системе

Исходя из анализа предметной области были выделены следующие варианты использования:

- отправить заявку в учебный центр;
- отменить заявку в учебный центр;
- назначить собеседование;
- провести собеседование;
- получить `xlsx`-отчет.

На рисунке 5 представлена диаграмма вариантов использования.

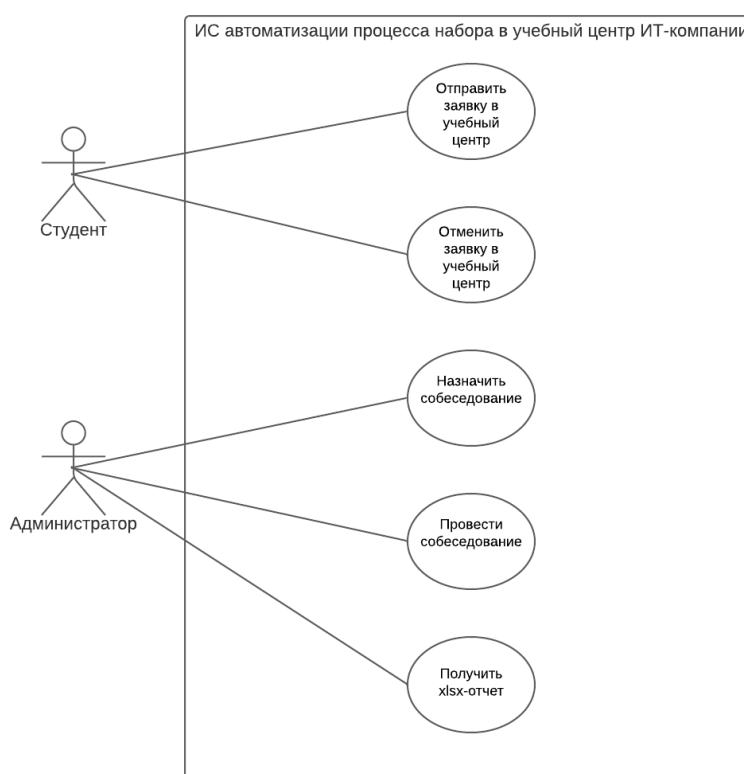


Рисунок 5 – Диаграмма вариантов использования

В системе существует две категории пользователей:

- студент – лицо, подавшее заявку в учебный центр через клиентское приложение;
- администратор – сотрудник учебного центра, ответственный за назначение и проведение собеседований.

1.9 Технологии и средства разработки

В качестве языка разработки выбран язык программирования Java версии 11. Этот выбор обусловлен высокой распространенностью языка Java и его востребованностью на рынке труда. Кроме того, язык Java является одним из наиболее общепризнанных инструментов для разработки веб-сервисов.

В качестве платформы для разработки бизнес-процессов выбрана Camunda Platform, рассчитанная на разработку на языках JVM. В настоящее

время Camunda является наиболее перспективной платформой для разработки бизнес-процессов и находит применение в разработке крупных коммерческих проектов.

Кроме этого, для использования в разработке информационной системы выбраны фреймворк Spring, являющийся стандартом разработки приложений на языках JVM, библиотека liquibase как средство для миграции и версионирования баз данных и библиотека Apache POI для генерации xlsx-документов. В качестве средства сборки проекта был выбран фреймворк Maven.

В качестве среды разработки выбрана IDE IntelliJ IDEA, являющаяся на данный момент наиболее распространенной интегрированной средой разработки на языках JVM.

В качестве СУБД выбрана реляционная СУБД PostgreSQL. В качестве инструмента для работы с базами данных был выбран DBeaver.

2 Проектирование информационной системы автоматизации процесса набора в учебный центр ИТ-компании

2.1 Базовая архитектура информационной системы

В качестве архитектуры информационной системы выбрана клиент-серверная архитектура. Клиент-серверная архитектура – это такая архитектура, нагрузка в которой распределена между заказчиками услуг, которых называют клиентами, и поставщиками услуг, которых называют серверами.

На рисунке 6 представлена диаграмма архитектуры информационной системы.

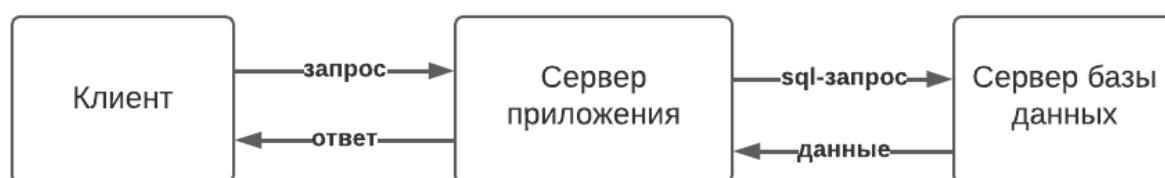


Рисунок 6 – Диаграмма архитектуры информационной системы

Клиент отправляет HTTP-запрос на сторону сервера приложения. Клиентом может являться браузер или любое приложение, умеющее отправлять HTTP-запросы.

Сервер приложения обрабатывает полученные HTTP-запросы от клиента. После этого сервер приложения обращается к серверу базы данных с sql-запросом.

Сервер базы данных отдает данные серверу приложения. Сервер приложения возвращает полученные данные в виде HTTP-ответа клиенту.

2.2 Проектирование структуры данных информационной системы

2.2.1 Разработка концептуальной модели базы данных

Модель «сущность-связь» – модель данных, которая позволяет описывать концептуальные схемы предметной области [16]. На основе

анализа предметной области в виде модели «сущность-связь» была построена концептуальная модель базы данных, представленная на рисунке 7.



Рисунок 7 – Концептуальная модель базы данных

Краткая спецификация сущностей базы данных:

- студент – пользователь, который отправил заявку в учебный центр;
- куратор – специалист ИТ-компании, ответственный за курирование прошедших в учебный центр студентов, один куратор может курировать несколько студентов (максимальное количество указано в поле «количество курируемых студентов»), но у одного студента может быть не больше одного куратора;
- заявка – заявка, которую отправляет студент для зачисления в учебный центр, студент может отправить несколько заявок, но не более одной на направление;
- направление – направление деятельности, на которую студент подает заявку для обучения, также соответствует специальности куратора, например, разработка, аналитика или тестирование;

- статус заявки – статусы, которыми могут обладать заявки, к примеру, оформлена, отменена и так далее;
- история заявки – отражает изменение статуса конкретной заявки;
- интервью – назначаемое студенту интервью после прохождения входного тестирования;
- реклама – отражает информацию о том, откуда кандидат узнал о наборе в учебный центр;
- настройки – параметры набора в учебный центр, которые можно указать до запуска информационной системы, к примеру, максимальное количество студентов на направление или минимальный порог прохождения входного тестирования, являются более приоритетными, чем настройки, указанные по умолчанию в конфигурационном файле информационной системы.

2.2.2 Разработка логической модели базы данных

В результате анализа требований и разработки концептуальной модели базы данных была разработана логическая модель базы данных в виде диаграммы «сущность-связь», представленная на рисунке 8.

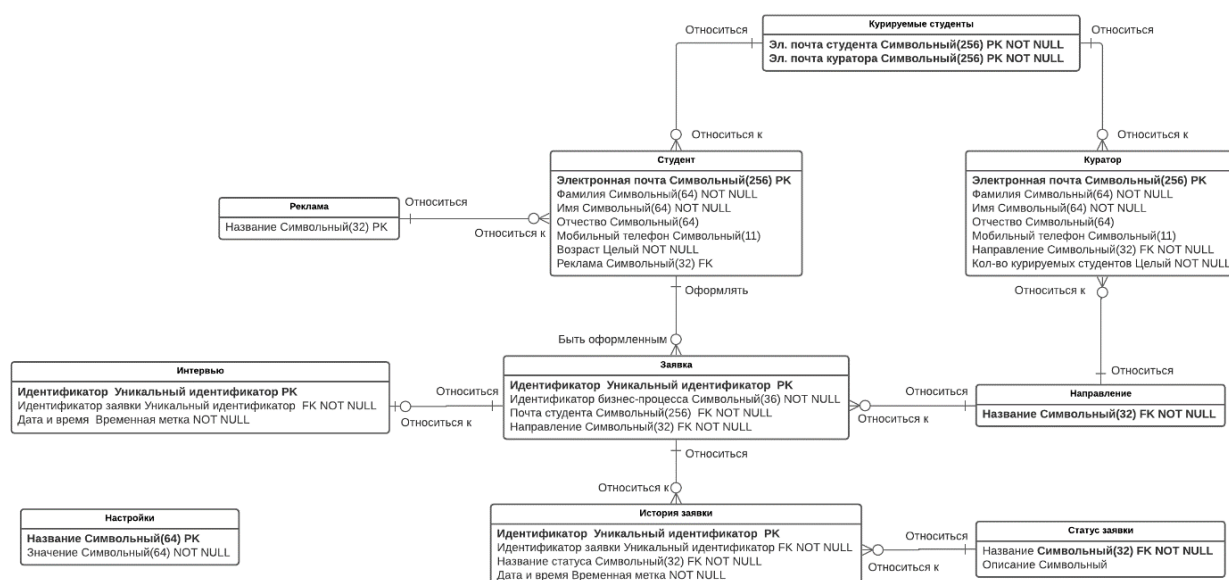


Рисунок 8 – Логическая модель базы данных

Были выделены отношения и их назначение, связи отношений и их характеристики, состав и параметры атрибутов отношений.

2.2.3 Разработка физической модели базы данных

По логической модели базы данных была разработана физическая модель базы данных в виде диаграммы «сущность-связь», представленная на рисунке 9.

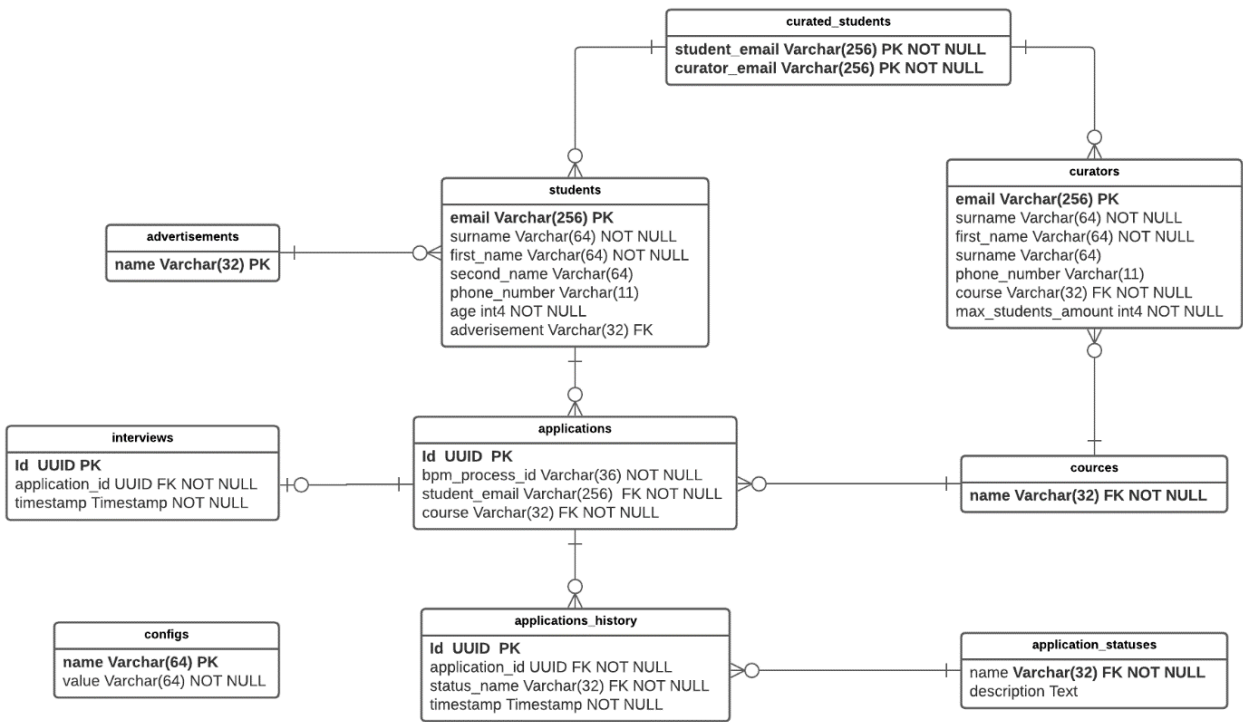


Рисунок 9 – Физическая модель базы данных

Физическая модель данных была разработана в соответствии с нотацией СУБД PostgreSQL, выбранной в качестве СУБД информационной системы автоматизации процесса набора в учебный центр ИТ-компании.

2.3 Проектирование информационной системы

2.3.1 Диаграмма бизнес-процесса

В приложении для графического моделирования бизнес-процессов Camunda Modeler в нотации BPMN был смоделирован бизнес-процесс обработки заявки в учебный центр (рисунок 10).

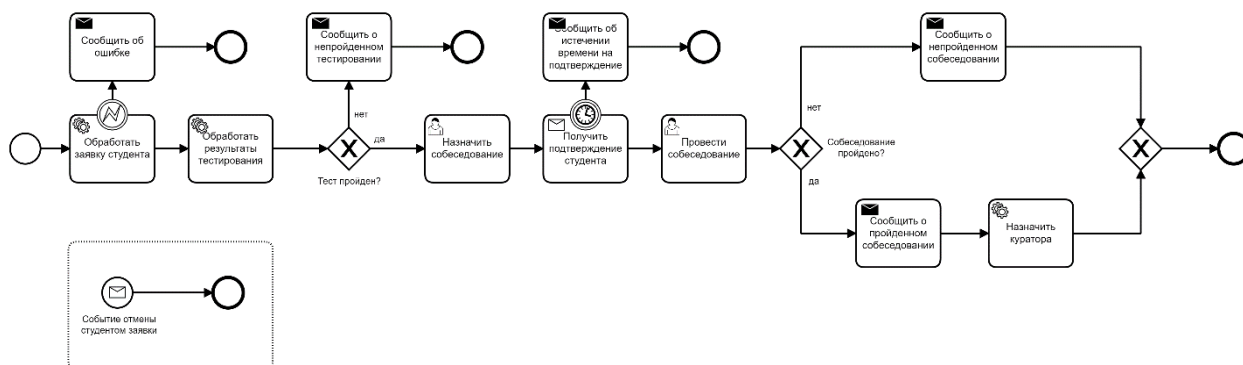


Рисунок 10 – Бизнес-процесс обработки заявки в учебный центр

В спроектированной диаграмме бизнес-процесса используются только шлюзы вида исключающее ИЛИ (XOR).

На диаграмме спроектированного бизнес-процесса можно увидеть следующие пустые события – начальное событие, обозначающие начало бизнес-процесса (пустой круг) и конечные события, обозначающие завершение бизнес-процесса (пустой круг с жирной границей).

Ниже приведено краткое описание бизнес-процесса.

После получения HTTP-запроса с данными заявки, информационная система инициирует запуск бизнес-процесса. Происходит обработка полученных данных, вызывается функция, написанная на языке Java.

Заявка является невалидной в том случае, если:

- студента, отправившего заявку, не существует в базе данных информационной системы;
- направления, на которое подана заявка, не существует в базе данных информационной системы;
- студент уже подал заявку на это направление.

Если полученная заявка является невалидной – система порождает сообщение о получении невалидной заявки и завершает бизнес-процесс.

Иначе заявка продвигается по бизнес-процессу и попадает в сервисную задачу по обработке результатов тестирования, происходит вызов Java-функции.

Если порог тестирования, значение, указанное в конфигурационном файле приложения или в таблице с конфигурациями набора в учебный центр (configs), не пройден, то система порождает сообщение о непройденном входном тестировании и завершает бизнес-процесс.

Иначе заявка продвигается по бизнес-процессу и попадает в пользовательскую задачу назначения собеседования. Задача назначена на администратора учебного центра и, после того, как администратор назначит дату и время собеседования в веб-приложении Camunda Tasklist, заявка попадет в задачу ожидания студентом подтверждения собеседования.

Информационная система слушает HTTP-запрос о подтверждении заявки. Если студент не подтвердит заявку за определенный срок, указанный в конфигурационном файле приложения или в таблице с конфигурациями набора в учебный центр (configs), то система порождает сообщение об истечении времени на подтверждение собеседования и завершает бизнес-процесс.

Иначе заявка продвигается по бизнес-процессу и попадает в пользовательскую задачу проведения собеседования. После того, как администратор назначит результат прохождения собеседования в веб-приложении Camunda Tasklist заявка продвигается дальше по бизнес-процессу в зависимости от результатов собеседования.

Если собеседование не пройдено, система порождает сообщение о непройденном собеседовании и завершает бизнес-процесс.

Если собеседование пройдено, система порождает сообщение о пройденном собеседовании, и заявка попадает в сервисную задачу назначения студенту куратора, вызывается функция, написанная на Java. После ее выполнения бизнес-процесс завершается.

В любой момент пользователь может отменить подачу заявки в учебный центр. Информационная система слушает HTTP-запрос и порождает событие отмены заявки. Бизнес-процесс слушает это событие и при его получении завершает свою работу.

2.3.2 Диаграмма классов информационной системы

На рисунке 11 представлена диаграмма классов информационной системы.

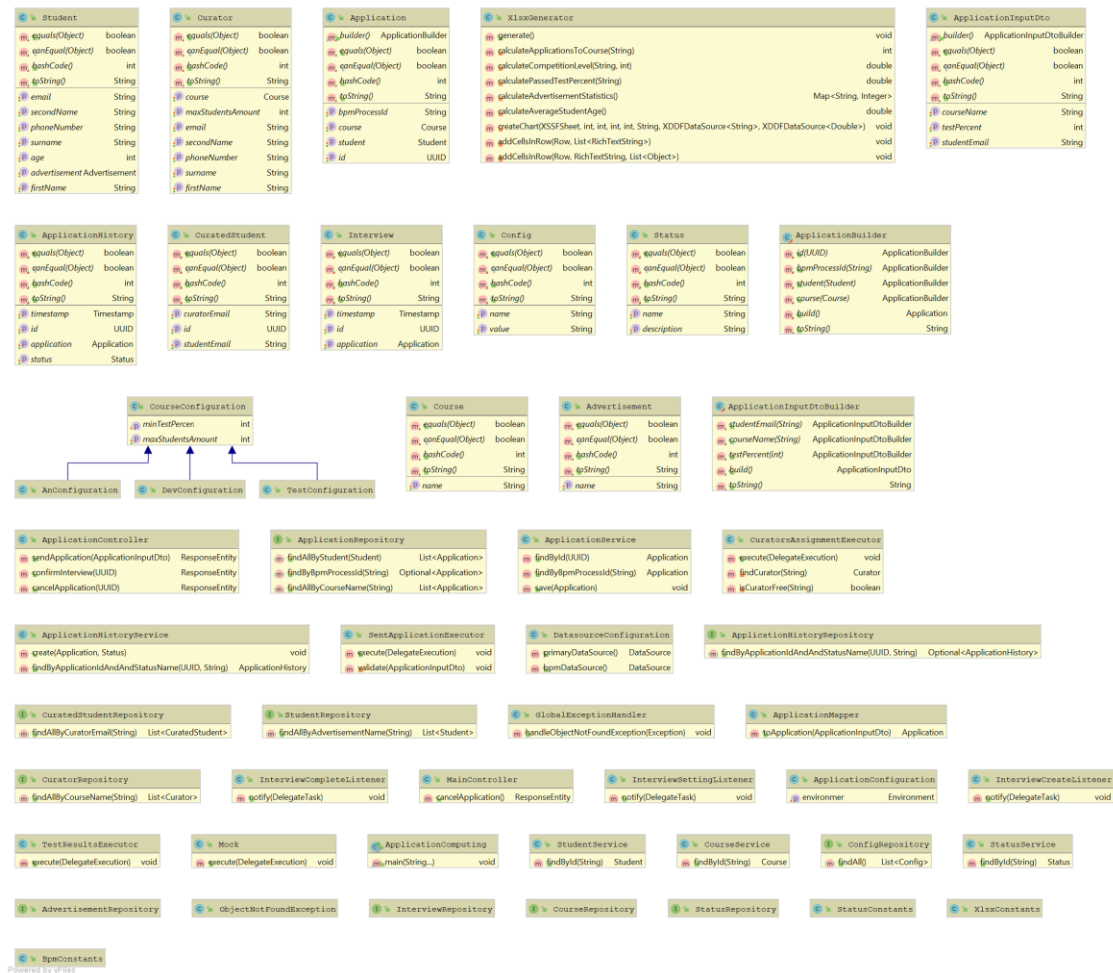


Рисунок 11 – Диаграмма классов

Краткая спецификация наиболее значимых классов информационной системы:

- Curator, ApplicationHistory, Application, Advertisement, Student, Interview, CuratedStudent, Status, Config, Course – классы, являющиеся эквивалентами сущностям базы данных;
- ApplicationInputDto – DTO-класс (Data Transfer Object) с информацией о приходящей в систему заявке;

- `ApplicationMapper` – класс, служащий для преобразования DTO-класса с информацией о приходящей в систему заявке в сущность заявки;
- классы с суффиксом `Constants` – классы, содержащие строковые константы названий заданий бизнес-процесса, статусов заявок, а также строковые константы для генерации `xlxs`-отчетов;
- классы с суффиксом `Configuration` – классы, считывающие параметры работы информационной системы из конфигурационного файла `application.yaml`;
- `ObjectNotFoundException` – пользовательский класс исключений;
- `GlobalExceptionHandler` – класс-обработчик исключений, возникающих во время работы информационной системы;
- классы с суффиксом `Controller` – классы-обработчики HTTP-запросов к системе;
- классы с суффиксом `Service` – классы, реализующие бизнес-логику работы с Java-объектами, эквивалентными сущностям базы данных;
- классы с суффиксом `Repository` – интерфейсы, необходимые для работы ORM Spring Data;
- `Mock` и классы с суффиксом `Executor` – классы с функциями, выполняемыми сервисными заданиями бизнес-процесса Camunda;
- Классы с суффиксом `Listener` – классы-обработчики событий;
- `XlsxGenerator` – класс, реализующий генерацию `xlsx`-отчетов;
- `WorkflowTest` – класс, содержащий модульные тесты.

2.3.3 Проектирование REST API

Поскольку разрабатываемая информационная система будет являться связующим программным обеспечением между клиентским приложением, ориентированным на работу с пользователями-кандидатами в учебный центр, и базой данных, она будет предоставлять API (сокращение от англ. Application Programming Interface, т. е. программный интерфейс приложения). API – описание способов, при помощи которых внешние системы могут

взаимодействовать с разрабатываемой информационной системой. Поскольку информационная система автоматизации процесса набора в учебный центр ИТ-компании является веб-сервисом, предоставляемым API будет являться список HTTP-запросов, которые информационная система умеет обрабатывать.

Спроектированное API будет соответствовать принципам REST (сокращение от англ. Representational State Transfer, т. е. передача самоописываемого состояния). REST – общепринятый архитектурный стиль взаимодействия приложений в сети.

В таблице 4 приведена спецификация REST API информационной системы.

Таблица 4 – Спецификация REST API

№	Назначение HTTP-запроса	HTTP-метод	URI
1	Отправка заявки в учебный-центр	POST	/application
2	Подтверждение интервью	POST	/application/confirm-interview/{id}
3	Отмена заявки	POST	/application/cancel-application/{id}
4	Генерация xlsx-отчета	POST	/generate-xlsx

Запросы с префиксом /application обрабатываются классом ApplicationController. Запрос на генерацию xlsx-отчета обрабатывается классом MainController.

В листинге 1 представлен пример JSON-объекта, являющегося телом запроса на отправку заявки.

Листинг 1 – Пример JSON-объекта

```
{"studentEmail": "iivanov@mail.ru",  
"courseName": "test",  
"testPercent": "49"}
```

Для запросов на подтверждение интервью и отмену заявки в учебный центр {id} в пути запроса является значением уникального идентификатора заявки в учебный центр.

Запрос на генерацию xlsx-отчета не предусматривает никакой информации в теле запроса.

3 Реализация информационной системы автоматизации набора в учебный центр ИТ-компании

3.1 Структура базы данных

База данных training-center-applications, хранящая сущности информационной системы, была создана в среде pgAdmin на стороне СУБД PostgreSQL.

Для описания структуры базы данных использовалась библиотека Liquibase, являющаяся системой управления версиями баз данных. Для описания структуры использовался формат SQL. В приложении Б (листинг Б.1 – Б.10) приведены исходные коды для создания таблиц базы данных.

Полученная структура базы данных представлена на рисунке 12.



Рисунок 12 – Структура базы данных

Таблицы databasechangelog и databasechangeloglock служат библиотеке Liquibase для контроля примененных изменений, они создаются автоматически и поэтому не были спроектированы ранее.

Для наполнения базы данных тестовыми данными так же использовалась библиотека Liquibase. В приложении Б (листинг Б.11 – Б.15) приведены исходные коды для наполнения базы данных тестовыми данными.

Кроме базы данных training-center-applications, существует база данных training-center-bpm, содержащая таблицы, необходимые для работы платформы для разработки бизнес-процессов Camunda.

3.2 Диаграмма компонентов информационной системы

Для удобства навигации по разрабатываемому проекту Java-классы со схожей логикой работы были перенесены в отдельные Java-пакеты. На рисунке 13 представлена диаграмма компонентов.

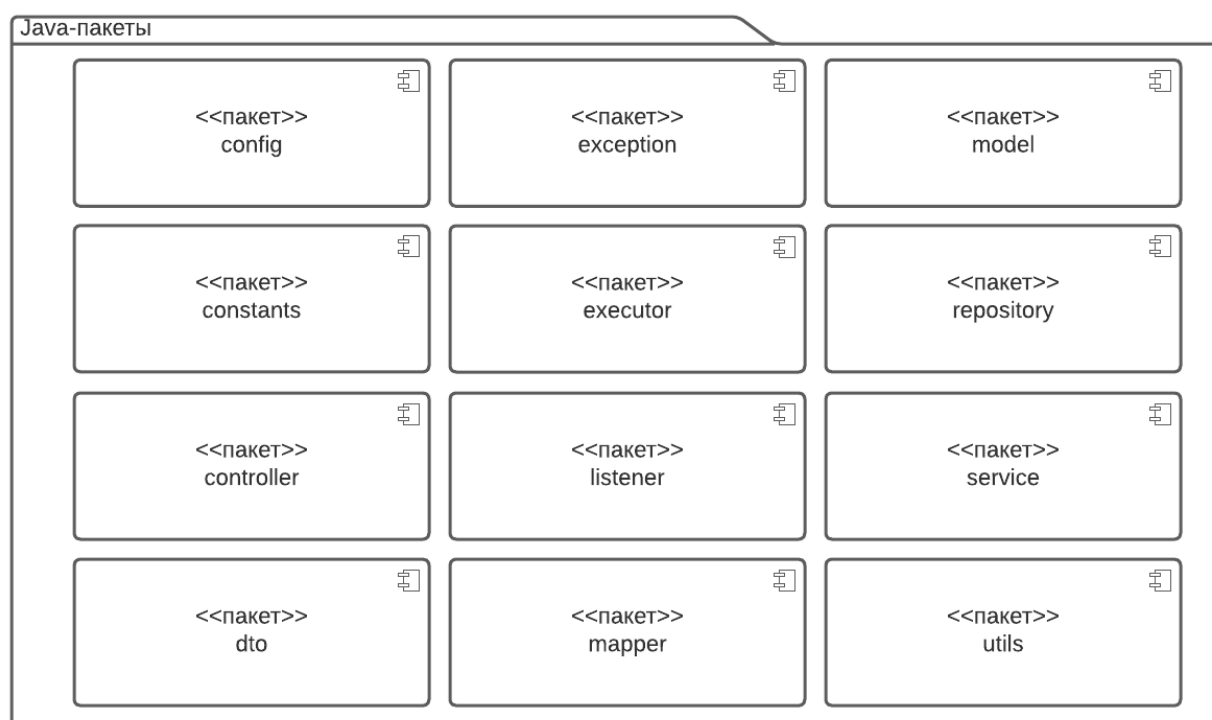


Рисунок 13 – Диаграмма компонентов

Краткая спецификация компонентов информационной системы:

- пакет `config` содержит классы, считывающие параметры работы информационной системы из конфигурационного файла;
- пакет `constants` включает классы, содержащие строковые константы названий заданий бизнес-процесса и статусов заявок;
- пакет `controller` содержит классы-обработчики HTTP-запросов к системе;
- пакет `dto` содержит DTO-класс (Data Transfer Object) с информацией о приходящей в систему заявке;
- пакет `exception` включает пользовательские классы исключений и класс-обработчик этих исключений;
- пакет `executor` содержит Java-код, выполняемый сервисными заданиями бизнес-процесса Camunda;
- пакет `listener` содержит классы-обработчики событий;
- пакет `mapper` содержит класс, служащий для преобразования DTO-класса с информацией о приходящей в систему заявке в сущность заявки;
- пакет `model` содержит Java-классы, являющиеся эквивалентами сущностям базы данных;
- пакет `repository` содержит интерфейсы, необходимые для работы ORM Spring Data;
- пакет `service` содержит классы, реализующие бизнес-логику работы с Java-объектами, эквивалентными сущностям базы данных;
- пакет `utils` содержит вспомогательные классы-утилиты.

В приложении В приведены листинги наиболее значимых классов информационной системы.

3.3 Диаграмма развертывания информационной системы

Информационная система была упакована в Docker-контейнер. Каждый контейнер содержит в себе все, что необходимо для работы приложения: код, используемые библиотеки и среду исполнения. Docker-контейнеры – это

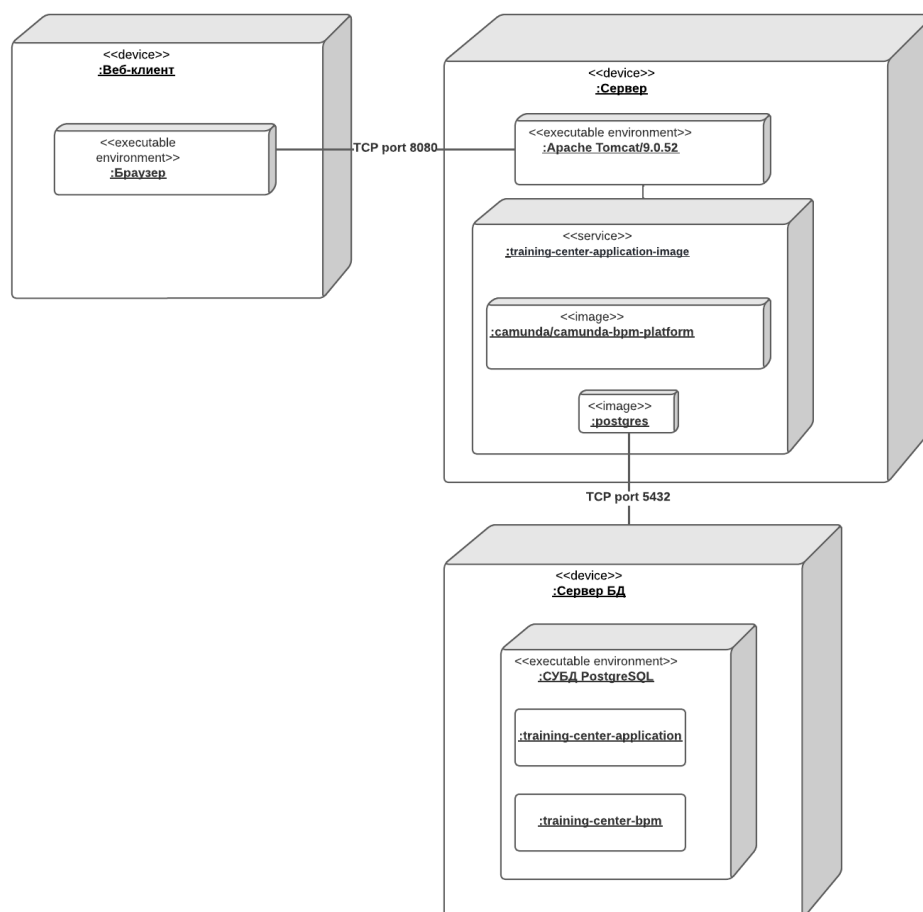


Рисунок 15 – Диаграмма развертывания информационной системы

На диаграмме представлены следующие узлы информационной системы:

- клиент;
- сервер приложения, содержащий сервер Tomcat и развернутый докер-контейнер, содержащий контейнеры платформы Camunda и СУБД PostgreSQL;
- сервер базы данных, содержащий СУБД PostgreSQL с двумя базами данных: training-center-application, содержащая сущности информационной системы, и training-center-bpm, содержащая таблицы для работы Camunda Platform.

3.4 Реализация генерации отчетов

Генерация отчетов о наборе в учебный центр была реализована при помощи библиотеки Apache POI для генерации xlsx-документов (листинг В.32).

Информационная система слушает POST-запрос по адресу [сервер:порт]/generate-xlsx, производит расчет статистики, генерацию документа training_center_statistics.xlsx и сохраняет сгенерированный документ в корневую директорию проекта.

Отчет содержит в себе два листа: один со статистикой набора, а второй с перечнем принятых студентов и назначенных им кураторов.

На рисунке 16 представлен лист со статистикой набора.

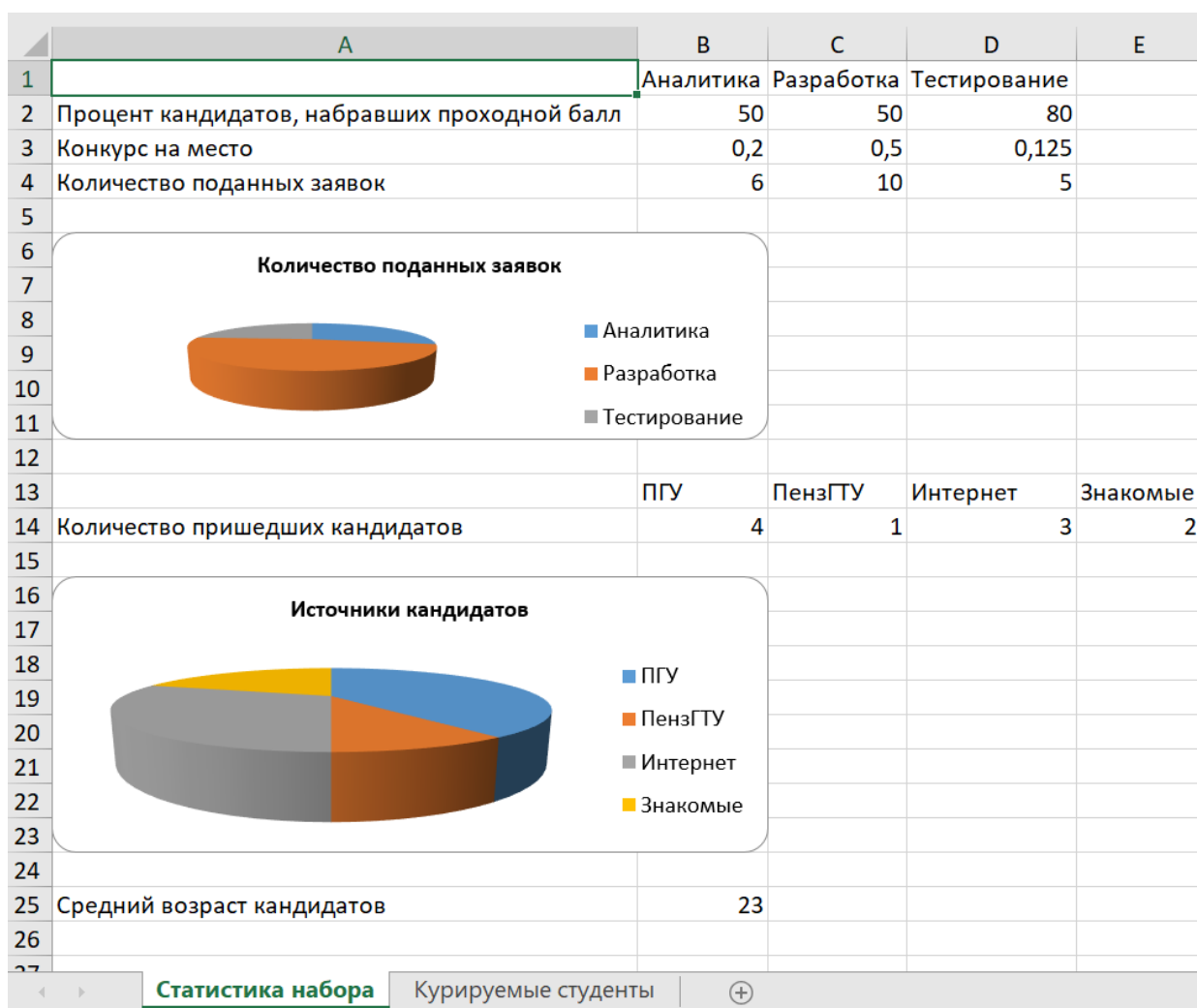


Рисунок 16 – Лист со статистикой набора

Рассчитываемая статистика:

- процент кандидатов, набравших проходной балл – характеризует эффективность входных тестов, при слишком большом или маленьком проценте, тест следует скорректировать;
- конкурс на место и количество поданных заявок на направление подготовки – характеризует интерес кандидатов к направлениям обучения;
- метрики источников кандидатов – анализ источников позволит использовать только наиболее эффективные площадки для привлечения кандидатов;
- средний возраст кандидата – является характеристикой, позволяющей определить портрет потенциального кандидата.

Для наилучшей наглядности были построены круговые диаграммы для количества заявок на направления обучения и количества кандидатов, подавших заявки в учебный центр благодаря различным платформам для привлечения в учебный центр.

Исходя из полученной статистики, можно сделать следующие выводы:

- входное тестирование по направлениям аналитика и разработка следует упростить;
- наиболее востребованное направление обучения – разработка;
- наиболее эффективная площадка для привлечения в учебный центр – ПГУ;
- средний возраст кандидата – двадцать три года, соответственно, учебный центр наиболее интересен для выпускников ВУЗов или людей, проработавших не в ИТ-сфере несколько лет.

На рисунке 17 представлен лист с курируемыми студентами. В таблице указаны почты студентов и их кураторов, а также направления обучения студентов.

	A	B	C
1	Студент	Куратор	Направление
2	gkuznetsov@mail.ru	aaleksandrov@trainingcenter.ru	Тестирование
3	ssemenova@mail.ru	aaleksandrov@trainingcenter.ru	Тестирование
4	aantonov@mail.ru	aaleksandrov@trainingcenter.ru	Тестирование
5	gkuznetsov@mail.ru	vvasnetsov@trainingcenter.ru	Аналитика
6	aartemiev@mail.ru	aaleksandrov@trainingcenter.ru	Тестирование
7	ppetrov@mail.ru	kmihailova@trainingcenter.ru	Разработка
8	eskvortsova@mail.ru	kmihailova@trainingcenter.ru	Разработка
9	iivanov@mail.ru	osidorova@trainingcenter.ru	Разработка
10	ssidorov@mail.ru	osidorova@trainingcenter.ru	Разработка
11	ddanilov@mail.ru	osidorova@trainingcenter.ru	Разработка
12	aartemiev@mail.ru	vvasnetsov@trainingcenter.ru	Аналитика
13	isidorov@mail.ru	vvasechkin@trainingcenter.ru	Аналитика
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			

Статистика набора
Курируемые студенты
+

Рисунок 17 – Лист с курируемыми студентами

То, что некоторые студенты встречаются в списке несколько раз, обусловлено тем, что кандидаты могут отправлять заявки на учебу в нескольких направлениях сразу.

4 Контроль качества информационной системы автоматизации набора в учебный центр ИТ-компании

4.1 Тестирование

4.1.1 Диаграмма состояний и переходов

Заявка в учебный центр может находиться в следующих состояниях:

- создана;
- невалидна;
- тест не пройден;
- тест пройден;
- просрочено подтверждение интервью;
- интервью подтверждено;
- интервью пройдено;
- интервью не пройдено;
- отменена.

На рисунке 18 представлена диаграмма состояний и переходов для сущности заявка.

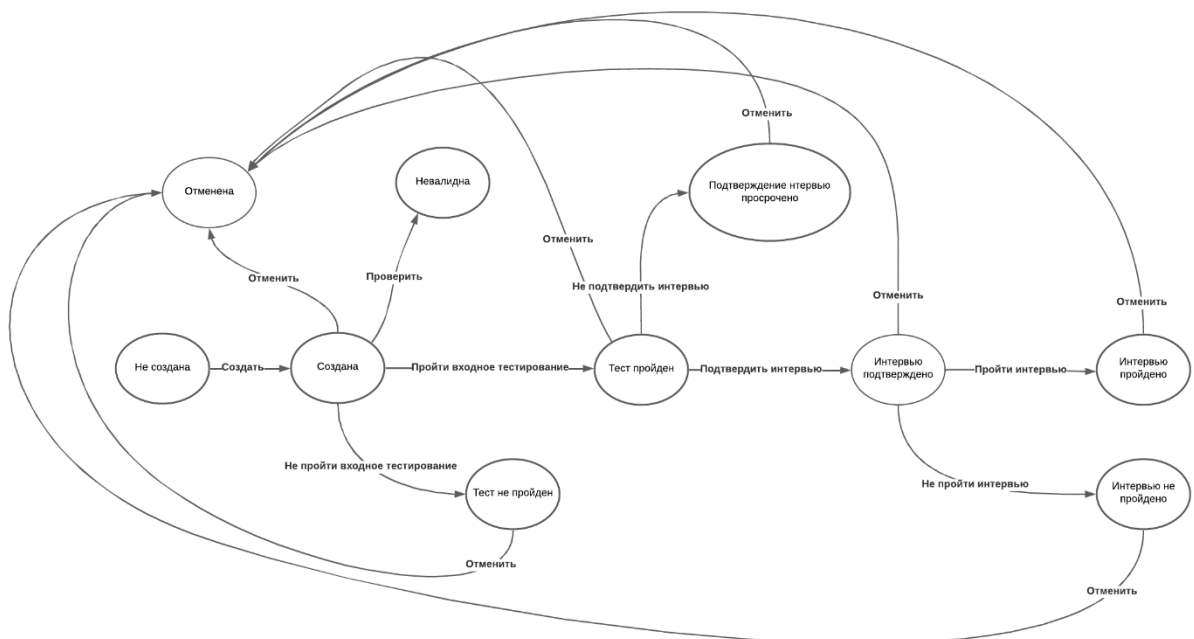


Рисунок 18 – Диаграмма состояний и переходов

Состояния на этой диаграмме обозначаются овалами, в центре которых – описание состояний, а переходы обозначены стрелками с подписями описания переходов.

Необходимо протестировать разработанную информационную систему таким образом, чтобы были пройдены маршруты через все состояния. Так, получилось шесть маршрутов:

- получение невалидной заявки (заявка создана и невалидна);
- непройденное входное тестирование (заявка создана и тест не пройден);
- неподтвержденное вовремя собеседование (заявка создана, тест пройден и интервью не подтверждено);
- непройденное собеседование (заявка создана, тест пройден, интервью подтверждено и интервью не пройдено);
- пройденное собеседование (заявка создана, тест пройден, интервью подтверждено и интервью пройдено);
- отмена заявки (заявка создана, находится в любом состоянии и отменена).

Необходимо уточнить, что при отмене заявки не происходит ее удаление из базы данных. В этом случае в таблицу истории заявок (`applications_history`) сохраняется информация о том, что заявка была отменена. Это реализовано таким образом с целью того, чтобы предотвратить ситуации повторной подачи заявок на направление, если пользователь, к примеру, не прошел входное тестирование, отменил заявку и решил снова подать заявку на это же направление с целью повторного прохождения входного тестирования.

4.1.2 Модульное тестирование

Для разрабатываемой информационной системы можно выделить следующие основные сценарии работы бизнес-процесса обработки заявок в учебный центр:

- получение невалидной заявки;
- непройденное входное тестирование;
- неподтвержденное вовремя собеседование;
- непройденное собеседование;
- пройденное собеседование;
- отмена заявки.

Для тестирования корректности работы информационной системы автоматизации процесса набора в учебный центр ИТ-компании были написаны модульные тесты на все вышеописанные сценарии работы бизнес-процесса.

Для написания модульных тестов на языке Java использовалась библиотека Junit и библиотеки, позволяющие писать модульные тесты для бизнес-процессов Camunda. В листинге В.33 представлен класс, содержащий модульные тесты на сценарии работы бизнес-процесса.

На рисунке 19 представлены результаты прохождения модульных тестов.

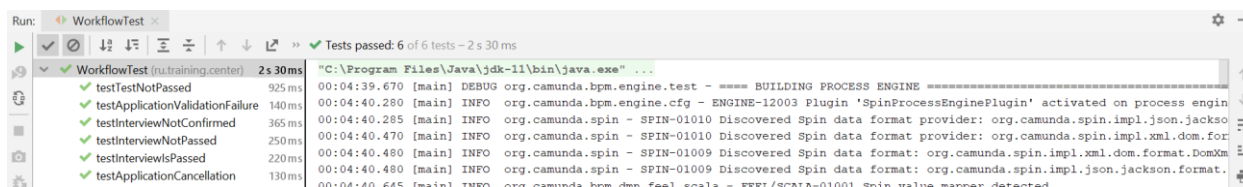


Рисунок 19 – Результаты прохождения модульных тестов

Все модульные тесты были пройдены.

Была использована библиотека для расчета процента тестового покрытия и генерации графического изображения покрытия модульными тестами бизнес-процесса обработки заявок в учебный центр. На рисунке 20 изображено сгенерированное библиотекой графическое изображение покрытия.

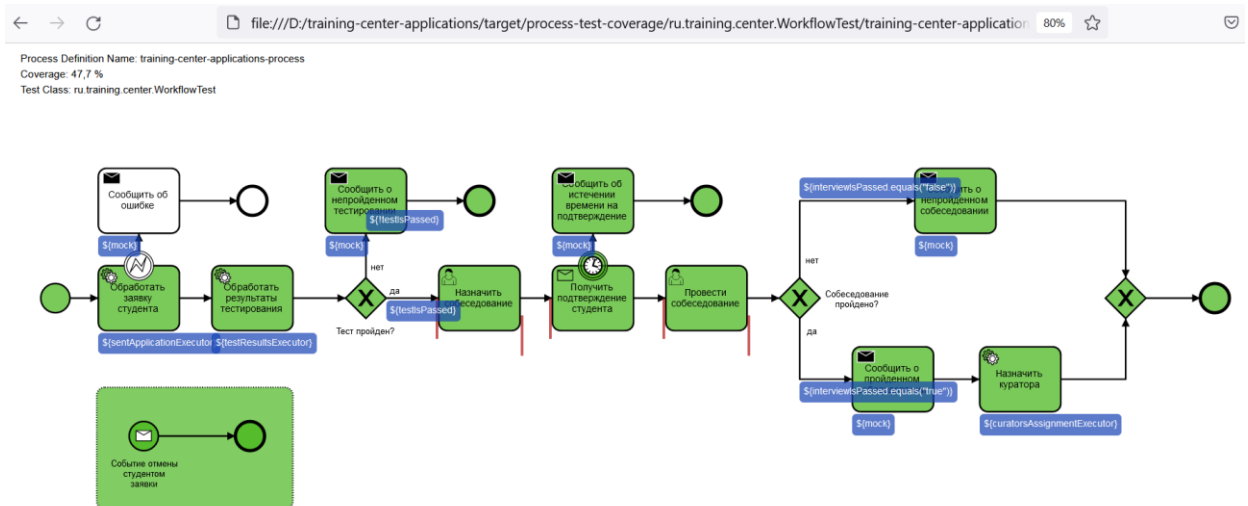


Рисунок 20 – Покрытие модульными тестами бизнес-процесса

Зеленым цветом выделены элементы бизнес-процесса, покрытые модульными тестами. Тест-кейс с невалидной заявкой в учебный центр был покрыт модульным тестом `testApplicationValidationFailure`, однако из-за особенностей работы вышеописанной библиотеки на рисунке он не покрашен. Рассчитанный библиотекой процент покрытия составил 47,7 процентов.

4.2 Расчет метрик кода

При помощи плагина в IDE IntelliJ IDEA рассчитана метрика LOC (сокращение от англ. Lines Of Code, т. е. количество строк кода). Результаты представлены на рисунке 21.

Таким образом, среднее количество строк кода в пакете составляет восемьдесят строк.

Были рассчитаны метрики устойчивости и абстрактности.

Метрика устойчивости для компоненты отражает количественную меру работы, которую необходимо произвести при изменении других зависимых компонент [18].

ru.training.center	14
ru.training.center.config	69
ru.training.center.constants	46
ru.training.center.controller	76
ru.training.center.dto	20
ru.training.center.exception	22
ru.training.center.executor	160
ru.training.center.listener	62
ru.training.center.mapper	20
ru.training.center.model	214
ru.training.center.repository	90
ru.training.center.service	96
ru.training.center.utils	155
Total	1 044
Average	80,31

Рисунок 21 – Расчет метрики LOC

Для вычисления устойчивости кода используется следующая формула:

$$I = \frac{Ce}{Ca + Ce},$$

где I – мера неустойчивости, которая изменяется в диапазоне от $[0, 1]$ (при $I = 0$ компонент максимально устойчив, соответственно, при $I = 1$ компонент максимально неустойчив);

Ce – количество классов вне данного компонента, зависящих от классов внутри компонента;

Ca – количество классов внутри данного компонента, зависящих от классов вне компонента.

Компоненты информационной системы (пакеты):

- config;
- constants;
- controller;
- dto;

- exception;
- executor;
- listener;
- mapper;
- model;
- repository;
- service;
- utils.

Для каждой компоненты была рассчитана метрика неустойчивости:

- $I_1 = \frac{6}{15+6} = 0,29;$
- $I_2 = \frac{0}{60+0} = 0;$
- $I_3 = \frac{27}{0+27} = 1;$
- $I_4 = \frac{0}{15+0} = 0;$
- $I_5 = \frac{0}{12+0} = 0;$
- $I_6 = \frac{89}{0+89} = 1;$
- $I_7 = \frac{17}{0+17} = 1;$
- $I_8 = \frac{11}{2+11} = 0,85;$
- $I_9 = \frac{0}{81+0} = 0;$
- $I_{10} = \frac{19}{27+19} = 0,41;$
- $I_{11} = \frac{35}{41+35} = 0,46;$
- $I_{12} = \frac{52}{2+52} = 0,96.$

Метрика A отражает меру абстрактности компонента.

Абстрактностью A называется величина

$$A = \frac{Na}{Nc},$$

где Nc – количество классов в компоненте;

Na – количество абстрактных классов в ней же.

Метрика A изменяется от 0 до 1, причем 0 означает, что в компоненте нет абстрактных классов, а 1, соответственно, что компонент состоит только из абстрактных классов.

Для каждой компоненты была рассчитана метрика абстрактности:

$$- A_1 = \frac{1}{6} = 0,17;$$

$$- A_2 = \frac{0}{3} = 0;$$

$$- A_3 = \frac{0}{2} = 0;$$

$$- A_4 = \frac{0}{1} = 0;$$

$$- A_5 = \frac{0}{2} = 0;$$

$$- A_6 = \frac{0}{4} = 0;$$

$$- A_7 = \frac{0}{3} = 0;$$

$$- A_8 = \frac{0}{1} = 0;$$

$$- A_9 = \frac{0}{10} = 0;$$

$$- A_{10} = \frac{10}{10} = 1;$$

$$- A_{11} = \frac{0}{5} = 0;$$

$$- A_{12} = \frac{0}{1} = 0.$$

Можно построить график, отложив на вертикальной оси меру абстрактности A , а на горизонтальной оси – меру неустойчивости I . Максимально «хорошие» компоненты будут располагаться в верхнем левом углу или в нижнем правом. Лучше всего, если изменчивые компоненты будут находиться на диагонали между точкой $(1, 0)$ и точкой $(0, 1)$, которую называют главной последовательностью.

На рисунке 22 изображены главная последовательность, зона неприятностей, где компонент является максимально устойчивым и при этом

не является абстрактным, и зона бесполезности, где компонент является и неустойчивым, и максимально абстрактным.

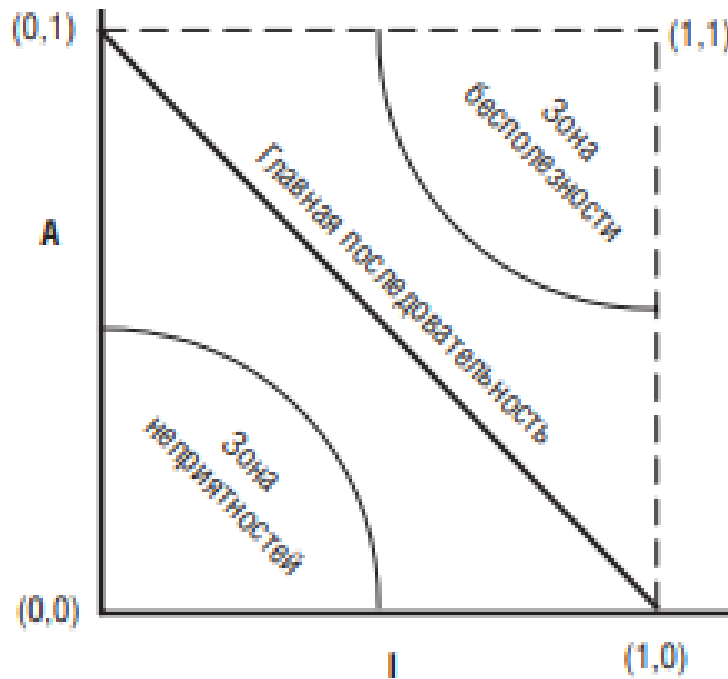


Рисунок 22 – Зоны исключения

Для вычисления количественной оценки используется формула вычисления расстояния до главной последовательности:

$$D = \left| \frac{(A + I - 1)}{\sqrt{2}} \right|,$$

где A – мера абстрактности;

I – мера неустойчивости.

Было вычислено расстояние до главной последовательности:

- $D_1 = \left| \frac{0.17+0.29-1}{\sqrt{2}} \right| = 0,39;$
- $D_2 = \left| \frac{0+0-1}{\sqrt{2}} \right| = 0,71;$
- $D_3 = \left| \frac{0+1-1}{\sqrt{2}} \right| = 0;$
- $D_4 = \left| \frac{0+0-1}{\sqrt{2}} \right| = 0,71;$
- $D_5 = \left| \frac{0+0-1}{\sqrt{2}} \right| = 0,71;$
- $D_6 = \left| \frac{0+1-1}{\sqrt{2}} \right| = 0;$

- $D_7 = \left| \frac{0+1-1}{\sqrt{2}} \right| = 0;$
- $D_8 = \left| \frac{0+0.85-1}{\sqrt{2}} \right| = 0,11;$
- $D_9 = \left| \frac{0+0-1}{\sqrt{2}} \right| = 0,71;$
- $D_{10} = \left| \frac{1+0.41-1}{\sqrt{2}} \right| = 0,29;$
- $D_{11} = \left| \frac{0+0.46-1}{\sqrt{2}} \right| = 0,39;$
- $D_{12} = \left| \frac{0+0.96-1}{\sqrt{2}} \right| = 0,03.$

Для рассчитанных метрик устойчивости и абстрактности логических компонент информационной системы был построен соответствующий график (рисунок 23).

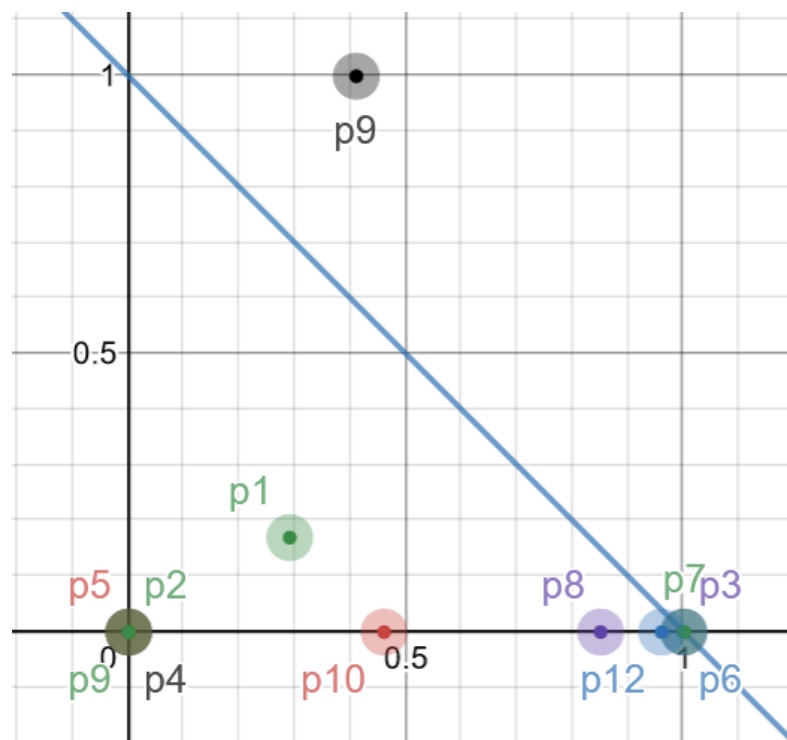


Рисунок 23 – Отображение рассчитанных метрик на графике

Пакеты config, constants, dto, exception, model и repository попали в зону неприяности. Эти пакеты не являются максимально устойчивыми и конкретными, таким образом, их нельзя расширить и сложно изменить. Однако это обусловлено тем, что данные часть этих пакетов отвечает за взаимодействие с базой данной при помощи ORM-средств (model и

repository), а остальные пакеты являются вспомогательными для остальных пакетов информационной системы.

Пакеты controller, executor и listener попали на главную последовательность, соответственно, они являются расширяемыми и изменяемыми.

4.3 Планирование и оценка бюджета

Для планирования и оценки бюджета разрабатываемой информационной системы были использованы средства программы для управления проектами Microsoft Project.

На рисунке 24 представлены ресурсы, использованные при разработке информационной системы.

	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на использ.	Начисление	Базовый календарь
1	Аналитик	Трудовой		A		100%	30 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
2	Разработчик	Трудовой		P		100%	35 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
3	Тестировщик	Трудовой		T		100%	30 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
4	Компьютер	Материальный	шт.	K			50 000,00 Р		0,00 Р	Пропорциональное	
5	Интернет	Затраты		I						Пропорциональное	

Рисунок 24 – Лист ресурсов

Были выделены три трудовых ресурса (аналитик, разработчик и тестировщик), в качестве материального ресурса указан компьютер, а также в качестве затрат – стоимость Интернета.

Проект разбит на задачи, на которые были назначены описанные выше ресурсы. В результате выделения задач и ресурсов была построена диаграмма Ганта, представленная на рисунке 25. Данная диаграмма является иллюстрацией графика работ по разработке информационной системы автоматизации процесса набора в учебный центр ИТ-компании.

Сроки задач поставлены в соответствии с производственным календарем 2022 года.

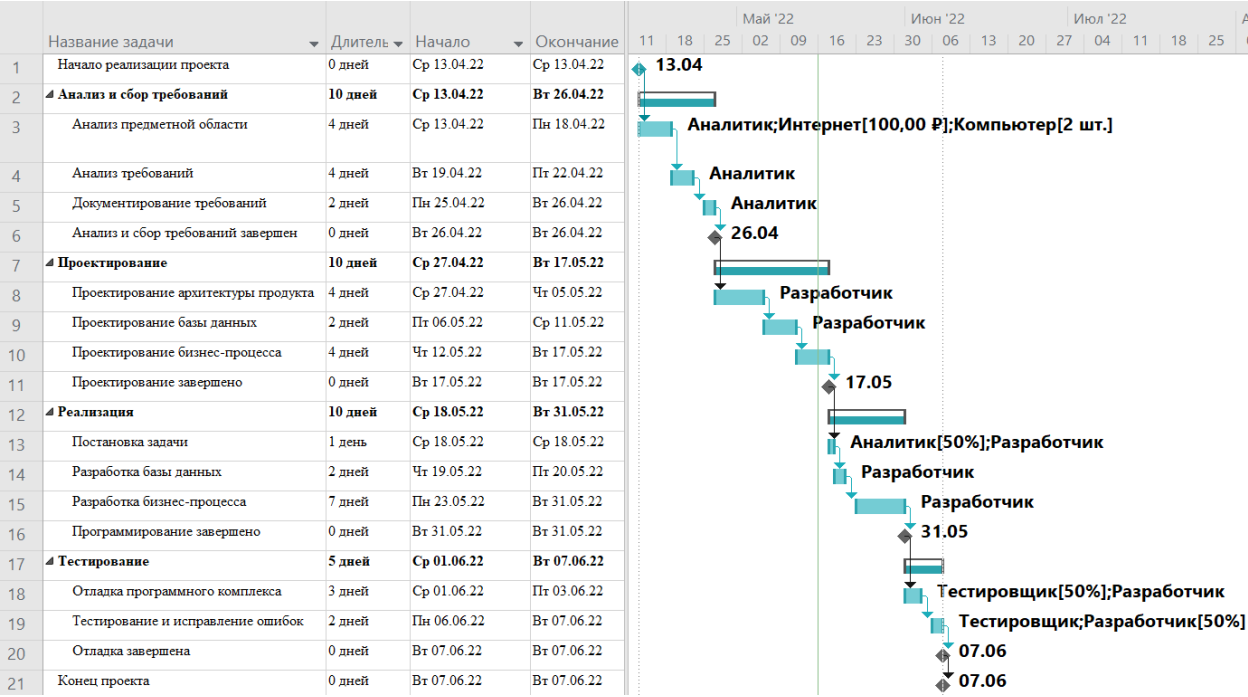


Рисунок 25 – Диаграмма Ганта

По полученному бизнес-плану был сгенерирован отчет по затратам (рисунок 26).

ОБЗОР ЗАТРАТ

СР 13.04.22 - ВТ 07.06.22



СОСТОЯНИЕ ЗАТРАТ
Состояние затрат для трудовых ресурсов.



ХОД ВЫПОЛНЕНИЯ И ЗАТРАТЫ
Ход выполнения в сравнении с затратами во времени. Превышение значения в строке "Совокупные затраты" над значением в строке "Процент завершения" указывает на возможное превышение бюджета.



РАСПРЕДЕЛЕНИЕ ЗАТРАТ
Распределение затрат между различными типами ресурсов.

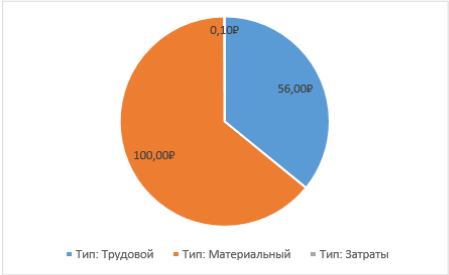


Рисунок 26 – Отчет по затратам

Длительность проекта составила 35 рабочих дней. Общее количество затрат – 156 100 рублей, причем большая часть затрат приходится на закупку двух компьютеров.

Заключение

В ходе выполнения выпускной квалификационной работы разработана информационная система автоматизации процесса набора в учебный центр ИТ-компании.

Подобрана и изучена научная литература по CRM-системам. Выяснено, что разработка CRM-систем для автоматизации работы учебных центров является перспективной и востребованной.

Исследованы понятие бизнес-процесса и различные виды нотаций для описания бизнес-процессов. В качестве нотации для описания бизнес-процессов разрабатываемой системы выбрана BPMN.

Рассмотрены платформы для разработки бизнес-процессов, в качестве платформы для реализации бизнес-процессов разрабатываемой системы выбрана Camunda Platform.

Проведен анализ CRM-систем для управления деятельностью учебных центров по интегральному и частным критериальным оценкам, выявлены их положительные и отрицательные стороны.

На основе проведенного анализа выделены функциональные и нефункциональные требования к системе.

Для разработки информационной системы выбрана архитектура системы. Также спроектирована концептуальная, логическая и физическая схема БД, отображающая основные сущности и связи в выбранной предметной области. Построена диаграмма бизнес-процесса обработки заявки в учебный центр. Построены диаграммы классов и компонентов информационной системы. Построена диаграмма развертывания.

Разработанное программное обеспечение протестировано. Выполнено модульное тестирование. Все тесты пройдены. Произведен расчет метрик кода.

Кроме того, проведено планирование и оценка бюджета проекта при помощи программы для управления проектами Microsoft Project.

Длительность проекта составила 35 рабочих дней. Общее количество затрат – 156 100 рублей.

Таким образом, в результате выполнения выпускной квалификационной работы разработана информационная система с возможностью автоматизации процесса набора в учебный центр ИТ-компании. Все требования к разработке информационной системы соблюдены.

Список использованных источников

1. Milan Krstic & Skorup Ana & Saveta Vukadinovic. From Product Customization To Customer Relationship Management // International Review. – 2014. – № 1 – 2.
2. Ключкова, А. В. CRM-система как инновационный инструмент повышения конкурентоспособности организации / А. В. Ключкова, А. А. Бебякина // Экономика и экологический менеджмент. – 2019. – №4. – С. 177 – 184.
3. Морозов Е.М. CRM-системы как средство автоматизации взаимодействия с клиентами // Современные научные исследования и инновации. 2015. № 3 [Электронный ресурс]. URL: <https://web.snauka.ru/issues/2015/03/49404> (дата обращения: 23.04.2022).
4. Захарова, Т. И. Сущность концепции управления взаимоотношениями с клиентами (CRM-системы) и ее роль в повышении эффективности деятельности в современных организациях / Т. И. Захарова, А. А. Иванов, О. П. Кокоулина, И. А. Халиль, А. А. Цимбалюк // Инновации и инвестиции. – 2021. – №6. – С. 44 – 51.
5. Юленков, С. Е. Роль современных CRM-систем в образовательных учреждениях / С. Е. Юленков, С. В. Котельникова, А. С. Касаткин // Решетневские чтения. – 2016. – №20. – С. 174 – 176.
6. Бабенко, В. В. Программные средства оптимизации бизнес-процесса «Приемная кампания ВУЗа» / В. В. Бабенко, Ю. В. Гольчевский, С. Т. Гуляева // Открытое образование. – 2015. – №3. – С. 57 – 62.
7. Коваленко, А. В. Автоматизация бизнес-процессов учебного центра / А. В. Коваленко, А. М. Пантелеева // Научный журнал КубГАУ. – 2015. – №110. – С. 388 – 400.
8. Елиферов, В. Г. Бизнес-процессы: Регламентация и управление: Учебник. / В. Г. Елиферов, В. В. Репин. – М.: ИНФРА-М, 2005. – 319 с.

9. Артамонов, И. В. Описание бизнес-процессов: вопросы стандартизации / И. В. Артамонов // Прикладная информатика. – 2011. – №3 (33). – С. 20 – 28.
10. Потрясаев, С. А. Комплексное моделирование сложных процессов на основе нотации BPMN / С. А. Потрясаев // Приборостроение. – 2016. – №11. – С. 913 – 920.
11. Обзор IBM Business Process Manager [Электронный ресурс]. URL: <https://www.ibm.com/docs/ru/bpm/8.6.0?topic=manager-business-process-overview> (дата обращения: 23.04.2022).
12. Обзор Camunda Platform [Электронный ресурс]. URL: <https://camunda.com/platform> (дата обращения: 23.04.2022).
13. Функциональные возможности продукта «1С: Управление учебным центром» [Электронный ресурс]. URL: <https://solutions.1c.ru/catalog/training-center/features> (дата обращения: 23.04.2022).
14. Функциональные возможности продукта «Информационная система управления деятельностью учебного центра» [Электронный ресурс]. URL: <https://sguits.ru/content/funkcionalnye-vozmozhnosti-0> (дата обращения: 23.04.2022).
15. Функциональные возможности продукта «Альфа CRM» [Электронный ресурс]. URL: <https://alfacrm.pro/> (дата обращения: 23.04.2022).
16. Крёнке, Д. Теория и практика построения баз данных. 8-е издание. / Д. Крёнке. – СПб.: Питер, 2003. – 800 с.
17. Документация Docker [Электронный ресурс]. URL: <https://docs.docker.com> (дата обращения: 14.06.2022).
18. Афанасьев, С. В. Метрики для объектно-ориентированного проектирования сложных систем. / С. В. Афанасьев, В. И. Воробьев // Вестник гражданских инженеров. СПб, издательство СГБГАСУ, 2005. – С. 108–114.

ПРИЛОЖЕНИЕ А
ГЛОССАРИЙ
(справочное)

CRM – это понятие, которое ставит пользователя в центр деятельности компании с целью максимизации прибыли, которая достигается путём увеличения количества клиентов, а также их удержания, и включает в себя определение важных для пользователей ценностей, автоматизацию бизнес-процессов для повышения эффективности продаж в области маркетинга и обслуживания.

Бизнес-процесс – обоснованный, взаимосвязанный и последовательный набор действий, который, используя ресурсы фирмы, создает ценность и выдает оперативный и правильный результат клиенту.

UML (сокращение от англ. Unified Modeling Language, т. е. унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения.

BPEL (сокращение от англ. Business Process Execution Language) – язык для описания бизнес-процессов и протоколов их взаимодействия между собой.

BPMN (сокращение от англ. Business Process Model and Notation, т. е. нотация и модель бизнес-процессов) – нотация для моделирования бизнес-процессов.

Camunda – платформа для разработки бизнес-процессов.

Клиент-серверная архитектура – это архитектура, нагрузка в которой распределена между заказчиками услуг, которых называют клиентами, и поставщиками услуг, которых называют серверами.

Модель «сущность-связь» – модель данных, которая позволяет описывать концептуальные схемы предметной области.

База данных (БД) – представленная в объективной форме совокупность самостоятельных материалов, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины.

Система управления базами данных (СУБД) – комплекс программ, позволяющих создание базы данных и манипулирование данными.

ORM (сокращение от англ. Object-Relational Mapping, т. е. объектно-реляционное отображение) – технология, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

HTTP (сокращение от англ. HyperText Transfer Protocol, т. е. протокол передачи гипертекста) – протокол передачи данных прикладного уровня.

API – описание способов, при помощи которых внешние системы могут взаимодействовать с разрабатываемой системой.

REST – общепринятый архитектурный стиль взаимодействия приложений в сети.

JSON (сокращение от англ. JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript.

Docker-контейнер – контейнер, содержащий в себе все, что необходимо для работы приложения: код, используемые библиотеки и среду исполнения.

Docker-образ – шаблон с инструкциями для создания контейнера.

Тестирование бизнес-процессов – это процедура исследования и испытания бизнес-процессов.

Метрика устойчивости – метрика, отражающая количественную меру работы, которую необходимо произвести при изменении других зависимых компонент.

Метрика абстрактности – метрика, отражающая меру абстрактности компонента.

ПРИЛОЖЕНИЕ Б
СКРИПТЫ ДЛЯ СОЗДАНИЯ ОБЪЕКТОВ БАЗЫ ДАННЫХ И
НАПОЛНЕНИЯ БАЗЫ ДАННЫХ
(рекомендуемое)

Листинг Б.1 – Файл 2022.05.01-create-configs-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_01_create_configs_table
CREATE TABLE IF NOT EXISTS configs (
    name VARCHAR(64) PRIMARY KEY NOT NULL,
    value VARCHAR(64) NOT NULL
)
```

Листинг Б.2 – Файл 2022.05.02-create-application-statuses-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_application_statuses_table
CREATE TABLE IF NOT EXISTS application_statuses (
    name VARCHAR(32) PRIMARY KEY NOT NULL,
    description TEXT
)
```

Листинг Б.3 – Файл 2022.05.02-create-applications-history-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_applications_history_table
CREATE TABLE IF NOT EXISTS applications_history (
    id UUID PRIMARY KEY NOT NULL,
    application_id UUID NOT NULL REFERENCES applications ON DELETE CASCADE,
    status_name VARCHAR(32) NOT NULL REFERENCES application_statuses,
    timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
)
```

Листинг Б.4 – Файл 2022.05.02-create-applications-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_applications_table
CREATE TABLE IF NOT EXISTS applications (
    id UUID PRIMARY KEY NOT NULL,
    bpm_process_id VARCHAR(36) NOT NULL,
    student_email VARCHAR(256) REFERENCES students ON DELETE SET NULL,
    course VARCHAR(32) NOT NULL REFERENCES courses ON DELETE SET NULL
)
```

Листинг Б.5 – Файл 2022.05.02-create-courses-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_courses_table
CREATE TABLE IF NOT EXISTS courses (
    name VARCHAR(32) PRIMARY KEY NOT NULL
)
```

Листинг Б.6 – Файл 2022.05.02-create-curated-students-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_curated_students_table
CREATE TABLE IF NOT EXISTS curated_students (
    id UUID PRIMARY KEY NOT NULL,
    curator_email VARCHAR(256) NOT NULL REFERENCES curators ON DELETE CASCADE,
    student_email VARCHAR(256) NOT NULL REFERENCES students ON DELETE CASCADE
)
```

Листинг Б.7 – Файл 2022.05.02-create-curators-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_curators_table
CREATE TABLE IF NOT EXISTS curators (
    email VARCHAR(256) PRIMARY KEY NOT NULL,
    surname VARCHAR(64) NOT NULL,
    first_name VARCHAR(64) NOT NULL,
    second_name VARCHAR(64),
    phone_number VARCHAR(11),
    course VARCHAR(32) NOT NULL REFERENCES courses ON DELETE SET NULL,
    max_students_amount INTEGER NOT NULL DEFAULT 1
)
```

Листинг Б.8 – Файл 2022.05.02-create-students-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_students_table
CREATE TABLE IF NOT EXISTS students (
    email VARCHAR(256) PRIMARY KEY NOT NULL,
    surname VARCHAR(64) NOT NULL,
    first_name VARCHAR(64) NOT NULL,
    second_name VARCHAR(64),
    phone_number VARCHAR(11),
    age INTEGER NOT NULL,
    advertisement VARCHAR(32) REFERENCES advertisements ON DELETE SET NULL
)
```

Листинг Б.9 – Файл 2022.05.05-create-interviews-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_05_create_interviews_table
CREATE TABLE IF NOT EXISTS interviews (
    id UUID PRIMARY KEY NOT NULL,
    application_id UUID NOT NULL REFERENCES applications ON DELETE CASCADE,
    timestamp TIMESTAMP NOT NULL
)
```

Листинг Б.10 – Файл 2022.05.17-create-advertisements-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_17_create_advertisements_table
CREATE TABLE IF NOT EXISTS advertisements (
    name VARCHAR(32) PRIMARY KEY NOT NULL
)
```

Листинг Б.11 – Файл 2022.05.03-fill-application-statuses-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_03_fill_application_statuses_table
INSERT INTO application_statuses (name) VALUES
    ('created'),
    ('test_passed'),
    ('test_not_passed'),
    ('interview_confirmed'),
    ('interview_not_confirmed'),
    ('interview_missed'),
    ('interview_passed'),
    ('interview_not_passed'),
    ('canceled')
```

Листинг Б.12 – Файл 2022.05.03-fill-courses-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_03_fill_courses_table
INSERT INTO courses VALUES
    ('an'),
    ('dev'),
    ('test')
```

Листинг Б.13 – Файл 2022.05.03-fill-curators-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_curators_table
INSERT INTO curators (email, surname, first_name, course,
max_students_amount) VALUES
    ('vvasnetsov@trainingcenter.ru', 'Васнецов', 'Василий', 'an', 2),
    ('vvasechkin@trainingcenter.ru', 'Васечкин', 'Василий', 'an', 3),
    ('epetrova@trainingcenter.ru', 'Петрова', 'Елена', 'an', 1),
    ('kmihailova@trainingcenter.ru', 'Михайлова', 'Ксения', 'dev', 2),
    ('osidorova@trainingcenter.ru', 'Сидорова', 'Оксана', 'dev', 3),
    ('aaleksandrov@trainingcenter.ru', 'Александров', 'Александр', 'test',
4),
    ('mmihailenko@trainingcenter.ru', 'Михайленко', 'Михаил', 'test', 3)
```

Листинг Б.14 – Файл 2022.05.03-fill-students-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_02_create_students_table
INSERT INTO students (email, surname, first_name, age, advertisement) VALUES
    ('eskvortsova@mail.ru', 'Скворцова', 'Екатерина', 21, 'ПГУ'),
    ('ppetrov@mail.ru', 'Петров', 'Петр', 27, 'Интернет'),
    ('iivanov@mail.ru', 'Иванов', 'Иван', 19, 'ПГУ'),
    ('ssidorov@mail.ru', 'Сидоров', 'Семен', 20, 'ПензГТУ'),
    ('ddanilov@mail.ru', 'Данилов', 'Даниил', 32, 'Знакомые'),
    ('ssemenova@mail.ru', 'Семенова', 'Ксения', 25, 'Интернет'),
    ('aartemiev@mail.ru', 'Артемьев', 'Артем', 18, 'ПГУ'),
    ('isidorov@mail.ru', 'Сидоров', 'Иван', 20, 'ПГУ'),
    ('gkuznetsov@mail.ru', 'Кузнецов', 'Георгий', 29, 'Знакомые'),
    ('aantonov@mail.ru', 'Антонов', 'Антон', 24, 'Интернет')
```

Листинг Б.15 – Файл 2022.05.17-fill-advertisements-table.sql

```
--liquibase formatted sql
--changeset skv.studies:2022_05_17_fill_advertisements_table
INSERT INTO advertisements (name) VALUES
    ('Интернет'),
    ('Знакомые'),
    ('ПГУ'),
    ('ПензГТУ')
```

Листинг Б.16 – Файл db.changelog-master.yml

```
databaseChangelog:
  - include:
      file: db/changelog/tables/2022.05.01-create-configs-table.sql
  - include:
      file: db/changelog/tables/2022.05.02-create-courses-table.sql
  - include:
      file: db/changelog/tables/2022.05.02-create-curators-table.sql
  - include:
      file: db/changelog/tables/2022.05.17-create-advertisements-table.sql
  - include:
      file: db/changelog/tables/2022.05.02-create-students-table.sql
```

```
- include:
  file: db/changelog/tables/2022.05.02-create-curated-students-table.sql
- include:
  file: db/changelog/tables/2022.05.02-create-applications-table.sql
- include:
  file: db/changelog/tables/2022.05.02-create-application-statuses-
table.sql
- include:
  file: db/changelog/tables/2022.05.02-create-applications-history-
table.sql
- include:
  file: db/changelog/inserts/2022.05.03-fill-application-statuses-
table.sql
- include:
  file: db/changelog/inserts/2022.05.03-fill-courses-table.sql
- include:
  file: db/changelog/inserts/2022.05.03-fill-curators-table.sql
- include:
  file: db/changelog/inserts/2022.05.17-fill-advertisements-table.sql
- include:
  file: db/changelog/inserts/2022.05.03-fill-students-table.sql
- include:
  file: db/changelog/tables/2022.05.05-create-interviews-table.sql
```

ПРИЛОЖЕНИЕ В
ИСХОДНЫЕ КОДЫ ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг В.1 – Файл DataSourceConfiguration.java

```
@Configuration
@EnableTransactionManagement
public class DataSourceConfiguration {
    @Bean
    @Primary
    @ConfigurationProperties(prefix="spring.datasource.primary")
    public DataSource primaryDataSource() {
        return DataSourceBuilder.create().build();
    }

    @Bean(name="camundaBpmDataSource")
    @ConfigurationProperties(prefix="spring.datasource.bpm")
    public DataSource bpmDataSource() {
        return DataSourceBuilder.create().build();
    }
}
```

Листинг В.2 – Файл ApplicationConfiguration.java

```
@Component
@AllArgsConstructor
public class ApplicationConfiguration implements EnvironmentAware {
    private ConfigRepository configRepository;

    @Override
    public void setEnvironment(Environment environment) {
        ConfigurableEnvironment configurableEnvironment =
        (ConfigurableEnvironment) environment;
        Map<String, Object> propertySource = new HashMap<>();
        configRepository.findAll().stream().forEach(config ->
        propertySource.put(config.getName(), config.getValue()));

        configurableEnvironment.getPropertySources().addAfter("systemEnvironment",
        new MapPropertySource("app-config", propertySource));
    }
}
```

Листинг В.3 – Файл CourseConfiguration.java

```
@Getter
@Setter
public abstract class CourseConfiguration {
    private int maxStudentsAmount;

    private int minTestPercent;
}
```

Листинг В.4 – Файл DevConfiguration.java

```
@ConfigurationProperties(prefix = "course.dev")
public class DevConfiguration extends CourseConfiguration {
}
```

Листинг В.5 – Файл BpmConstants.java

```
public class BpmConstants {
    public static final String PROCESS_KEY = "training-center-applications-
    process";

    public static final String STUDENT_EMAIL_VAR_NAME = "studentEmail";
}
```

```

public static final String COURSE_NAME_VAR_NAME = "courseName";

public static final String TEST_PERCENT_VAR_NAME = "testPercent";

public static final String APPLICATION_ID_VAR_NAME = "applicationId";

public static final String INTERVIEW_DATE_VAR_NAME = "interviewDate";

public static final String INTERVIEW_TIME_VAR_NAME = "interviewTime";

public static final String APPLICATION_VALIDATION_ERROR_CODE =
"ApplicationValidationError";

public static final String AN_COURSE_NAME = "an";

public static final String DEV_COURSE_NAME = "dev";

public static final String TEST_COURSE_NAME = "test";

public static final String TEST_IS_PASSED_VAR_NAME = "testIsPassed";

public static final String INTERVIEW_DATE_TIME_VAR_NAME =
"interviewDateTime";

public static final String INTERVIEW_CONFIRMATION_MESSAGE =
"interviewConfirmation";

public static final String INTERVIEW_IS_PASSED_VAR_NAME =
"interviewIsPassed";

public static final String APPLICATION_CANCELLATION_MESSAGE =
"applicationCancellation";
}

```

Листинг В.6 – Файл StatusConstants.java

```

public class StatusConstants {
    public static final String CREATED = "created";

    public static final String TEST_PASSED = "test_passed";

    public static final String TEST_NOT_PASSED = "test_not_passed";

    public static final String INTERVIEW_CONFIRMED = "interview_confirmed";

    public static final String INTERVIEW_PASSED = "interview_passed";

    public static final String INTERVIEW_NOT_PASSED = "interview_not_passed";

    public static final String CANCELED = "canceled";
}

```

Листинг В.7 – Файл XlsxConstants.java

```

public class XlsxConstants {
    public static final String STATISTICS_SHEET_NAME = "Статистика набора";

    public static final String AN = "Аналитика";

    public static final String DEV = "Разработка";

    public static final String TEST = "Тестирование";

    public static final String APPLICATIONS_AMOUNT = "Количество поданных

```

```

заявок";

    public static final String COMPETITION_LEVEL = "Конкурс на место";

    public static final String PASSED_TEST_PERCENT = "Процент кандидатов,
набравших проходной балл";

    public static final String STUDENTS_CAME_FROM_ADVERTISEMENT = "Количество
пришедших кандидатов";

    public static final String ADVERTISEMENT = "Источники кандидатов";

    public static final String AVERAGE_STUDENT_AGE = "Средний возраст
кандидатов";

    public static final String CURATED_STUDENTS = "Курируемые студенты";

    public static final String STUDENT = "Студент";

    public static final String CURATOR = "Куратор";

    public static final String COURSE = "Направление";

    public static final String XLSX_FILE_NAME =
"training_center_statistics.xlsx";
}

```

Листинг В.8 – Файл ApplicationController.java

```

@RestController
@RequestMapping("/application")
@AllArgsConstructor
public class ApplicationController {
    private final ProcessEngine processEngine;

    private final ApplicationService applicationService;

    private final ApplicationHistoryService applicationHistoryService;

    private final StatusService statusService;

    @PostMapping("")
    public ResponseEntity sendApplication(@RequestBody ApplicationInputDto
applicationInputDto) {
        Map<String, Object> variables = new HashMap<>();
        variables.put(STUDENT_EMAIL_VAR_NAME,
applicationInputDto.getStudentEmail());
        variables.put(COURSE_NAME_VAR_NAME,
applicationInputDto.getCourseName());
        variables.put(TEST_PERCENT_VAR_NAME,
applicationInputDto.getTestPercent());
        processEngine.getRuntimeService()
            .createProcessInstanceByKey(PROCESS_KEY)
            .setVariables(variables)
            .executeWithVariablesInReturn();
        return new ResponseEntity(HttpStatus.CREATED);
    }

    @PostMapping("/confirm-interview/{id}")
    public ResponseEntity confirmInterview(@PathVariable UUID id) {
        processEngine
            .getRuntimeService()
            .createMessageCorrelation(INTERVIEW_CONFIRMATION_MESSAGE)
    }
}

```

```

        .processInstanceId(applicationService.findById(id).getBpmProcessId())
        .correlate();
        applicationHistoryService.create(applicationService.findById(id),
            statusService.findById(INTERVIEW_CONFIRMED));
        return new ResponseEntity(HttpStatus.OK);
    }

    @PostMapping("/cancel-application/{id}")
    public ResponseEntity cancelApplication(@PathVariable UUID id) {
        processEngine
            .getRuntimeService()
            .messageEventReceived(APPLICATION_CANCELLATION_MESSAGE,
                applicationService.findById(id).getBpmProcessId());
        applicationHistoryService.create(applicationService.findById(id),
            statusService.findById(CANCELED));
        return new ResponseEntity(HttpStatus.OK);
    }
}

```

Листинг В.9 – Файл MainController.java

```

@RestController
@RequestMapping("/")
@AllArgsConstructor
public class MainController {
    private final XlsxGenerator xlsxGenerator;

    @PostMapping("/generate-xlsx")
    public ResponseEntity cancelApplication() throws IOException {
        xlsxGenerator.generate();
        return new ResponseEntity(HttpStatus.OK);
    }
}

```

Листинг В.10 – Файл ApplicationInputDto.java

```

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ApplicationInputDto {
    @NotNull
    @Email
    private String studentEmail;

    @NotNull
    private String courseName;

    @NotNull
    private int testPercent;
}

```

Листинг В.11 – Файл GlobalExceptionHandler.java

```

@ControllerAdvice
@Slf4j
public class GlobalExceptionHandler {
    @ExceptionHandler(ObjectNotFoundException.class)
    public void handleObjectNotFoundException(Exception ex) {
        log.info(ex.getMessage());
    }
}

```

Листинг В.12 – Файл ObjectNotFoundException.java

```
public class ObjectNotFoundException extends RuntimeException {
    public ObjectNotFoundException(String object, UUID id) {
        super(String.format("%s with id %s not found.", object, id));
    }

    public ObjectNotFoundException(String object, String id) {
        super(String.format("%s with id %s not found.", object, id));
    }
}
```

Листинг В.13 – Файл CuratorsAssignmentExecutor.java

```
@Component
@AllArgsConstructor
public class CuratorsAssignmentExecutor implements JavaDelegate {
    private final CuratorRepository curatorRepository;

    private final CuratedStudentRepository curatedStudentRepository;

    private final ApplicationHistoryService applicationHistoryService;

    @Override
    public void execute(DelegateExecution execution) {
        Application application = applicationHistoryService
            .findByApplicationIdAndAndStatusName(UUID.fromString(execution.getVariable(APPLICATION_ID_VAR_NAME).toString()), INTERVIEW_PASSED)
            .getApplication();
        Curator curator = findCurator(application.getCourse().getName());
        if (curator != null) {
            curatedStudentRepository.save(new
CuratedStudent(curator.getEmail(), application.getStudent().getEmail()));
        }

        private Curator findCurator(String course) {
            for (var curator : curatorRepository.findAllByCourseName(course)) {
                if (isCuratorFree(curator.getEmail())) {
                    return curator;
                }
            }
            return null;
        }

        private boolean isCuratorFree(String curator) {
            return curatedStudentRepository.findAllByCuratorEmail(curator).size()
<
curatorRepository.findById(curator).get().getMaxStudentsAmount();
        }
    }
}
```

Листинг В.14 – Файл SentApplicationExecutor.java

```
@Component
@AllArgsConstructor
public class SentApplicationExecutor implements JavaDelegate {
    private final ApplicationMapper applicationMapper;
```

```

private final ApplicationService applicationService;

private final ApplicationRepository applicationRepository;

private final StudentService studentService;

private final CourseService courseService;

private final ApplicationHistoryService applicationHistoryService;

private final StatusService statusService;

@Override
public void execute(DelegateExecution execution) {
    ApplicationInputDto applicationInputDto = ApplicationInputDto
        .builder()

        .studentEmail(execution.getVariable(STUDENT_EMAIL_VAR_NAME).toString())

        .courseName(execution.getVariable(COURSE_NAME_VAR_NAME).toString())
            .testPercent((Integer)
                execution.getVariable(TEST_PERCENT_VAR_NAME))
            .build();
        validate(applicationInputDto);

        Application application =
            applicationMapper.toApplication(applicationInputDto);
        application.setBpmProcessId(execution.getProcessInstanceId());
        applicationService.save(application);
        applicationHistoryService.create(application,
            statusService.findById(CREATED));
        execution.setVariable(APPLICATION_ID_VAR_NAME,
            application.getId().toString());
        execution.setVariable(INTERVIEW_DATE_VAR_NAME, "");
        execution.setVariable(INTERVIEW_TIME_VAR_NAME, "");
    }

    private void validate(ApplicationInputDto applicationInputDto) {
        if (studentService.findById(applicationInputDto.getStudentEmail()) ==
            null ||
            courseService.findById(applicationInputDto.getCourseName())
            == null ||
            applicationRepository

                .findAllByStudent(studentService.findById(applicationInputDto.getStudentEmail()
                    ()))
                    .stream()
                    .anyMatch(it ->
                        it.getCourse().getName().equals(applicationInputDto.getCourseName()))) {
            throw new BpmnError(APPLICATION_VALIDATION_ERROR_CODE);
        }
    }
}

```

Листинг В.15 – Файл TestResultsExecutor.java

```

@Component
@AllArgsConstructor
@DependsOn("applicationConfiguration")
public class TestResultsExecutor implements JavaDelegate {
    private final AnConfiguration anConfiguration;

    private final DevConfiguration devConfiguration;

```

```

private final TestConfiguration testConfiguration;

private final ApplicationService applicationService;

private final StatusService statusService;

private final ApplicationHistoryService applicationHistoryService;

@Override
public void execute(DelegateExecution execution) {
    boolean testIsPassed = false;
    switch
(applicationService.findByBpmProcessId(execution.getProcessInstanceId()).getC
ourse().getName()) {
        case AN_COURSE_NAME : {
            testIsPassed = (Integer)
execution.getVariable(TEST_PERCENT_VAR_NAME) >=
anConfiguration.getMinTestPercent();
            break;
        }
        case DEV_COURSE_NAME : {
            testIsPassed = (Integer)
execution.getVariable(TEST_PERCENT_VAR_NAME) >=
devConfiguration.getMinTestPercent();
            break;
        }
        case TEST_COURSE_NAME : {
            testIsPassed = (Integer)
execution.getVariable(TEST_PERCENT_VAR_NAME) >=
testConfiguration.getMinTestPercent();
            break;
        }
        default: break;
    }
    execution.setVariable(TEST_IS_PASSED_VAR_NAME, testIsPassed);
    Status status = testIsPassed ? statusService.findById(TEST_PASSED) :
statusService.findById(TEST_NOT_PASSED);

    applicationHistoryService.create(applicationService.findByBpmProcessId(execut
ion.getProcessInstanceId()), status);
}
}

```

Листинг В.16 – Файл InterviewCompleteListener.java

```

@Component
@AllArgsConstructor
public class InterviewCompleteListener implements TaskListener {
    private final StatusService statusService;

    private final ApplicationService applicationService;

    private final ApplicationHistoryService applicationHistoryService;

    @Override
    public void notify(DelegateTask delegateTask) {
        Status status =
delegateTask.getVariable(INTERVIEW_IS_PASSED_VAR_NAME).equals("true") ?
statusService.findById(INTERVIEW_PASSED)
: statusService.findById(INTERVIEW_NOT_PASSED);

        applicationHistoryService.create(applicationService.findByBpmProcessId(delega
teTask.getProcessInstanceId()), status);
    }
}

```

```

    }
}

```

Листинг В.17 – Файл InterviewCreateListener.java

```

@Component
public class InterviewCreateListener implements TaskListener {
    @Override
    public void notify(DelegateTask delegateTask) {
        delegateTask.setVariable(INTERVIEW_IS_PASSED_VAR_NAME, "false");
    }
}

```

Листинг В.18 – Файл InterviewSettingListener.java

```

@Component
@AllArgsConstructor
public class InterviewSettingListener implements TaskListener {
    private final ApplicationService applicationService;

    private final InterviewRepository interviewRepository;

    private static final String DATE_TIME_PATTERN = "yyyy-MM-dd HH:mm";

    @Override
    public void notify(DelegateTask delegateTask) {
        DateTimeFormatter formatter =
        DateTimeFormatter.ofPattern(DATE_TIME_PATTERN);
        interviewRepository.save(new
        Interview(applicationService.findByIdByBpmProcessId(delegateTask.getProcessInstance
        ceId()),

        Timestamp.valueOf(LocalDate.parse(delegateTask.getVariable(INTERVIEW_DATE
        _TIME_VAR_NAME).toString(), formatter))));
    }
}

```

Листинг В.19 – Файл ApplicationMapper.java

```

@AllArgsConstructor
@Component
public class ApplicationMapper {
    private final StudentRepository studentRepository;

    private final CourseRepository courseRepository;

    public Application toApplication(ApplicationInputDto applicationInputDto)
    {
        return Application
            .builder()

            .student(studentRepository.findById(applicationInputDto.getStudentEmail()).ge
            t())

            .course(courseRepository.findById(applicationInputDto.getCourseName()).get())
            .build();
    }
}

```

Листинг В.20 – Файл Application.java

```

@Entity
@Table(name = "applications")
@Data

```

```

@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Application {
    @Id
    @GeneratedValue
    private UUID id;

    private String bpmProcessId;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "student_email")
    @NonNull
    private Student student;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "course")
    @NonNull
    private Course course;
}

```

Листинг В.21 – Файл ApplicationHistory.java

```

@Entity
@Table(name = "applications_history")
@Data
@NoArgsConstructor
@AllArgsConstructor
@RequiredArgsConstructor
public class ApplicationHistory {
    @Id
    @GeneratedValue
    private UUID id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "application_id")
    @NonNull
    private Application application;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "status_name")
    @NonNull
    private Status status;

    @NonNull
    private Timestamp timestamp;
}

```

Листинг В.22 – Файл Student.java

```

@Entity
@Table(name = "students")
@Data
@NoArgsConstructor
public class Student {
    @Id
    @NonNull
    private String email;

    @NonNull
    private String surname;

    @NonNull
    @Column(name = "first_name")

```

```

private String firstName;

@Column(name = "second_name")
private String secondName;

@Column(name = "phone_number")
private String phoneNumber;

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "advertisement")
@NotNull
private Advertisement advertisement;

@NotNull
private int age;
}

```

Листинг В.23 – Файл ApplicationRepository.java

```

@Repository
public interface ApplicationRepository extends CrudRepository<Application,
UUID> {
    List<Application> findAllByStudent(Student student);

    Optional<Application> findByBpmProcessId(String bpmProcessId);

    List<Application> findAllByCourseName(String course);
}

```

Листинг В.24 – Файл ApplicationHistoryRepository.java

```

@Repository
public interface ApplicationHistoryRepository extends
CrudRepository<ApplicationHistory, UUID> {
    Optional<ApplicationHistory> findByApplicationIdAndAndStatusName (UUID
application, String status);
}

```

Листинг В.25 – Файл StudentRepository.java

```

@Repository
public interface StudentRepository extends CrudRepository<Student, String> {
    List<Student> findAllByAdvertisementName(String advertisement);
}

```

Листинг В.26 – Файл ApplicationService.java

```

@Service
@AllArgsConstructor
public class ApplicationService {
    private final ApplicationRepository applicationRepository;

    public Application findById(UUID id) {
        return applicationRepository
            .findById(id)
            .orElseThrow(() -> new
ObjectNotFoundException(Application.class.getSimpleName(), id));
    }

    public Application findByBpmProcessId(String id) {
        return applicationRepository
            .findByBpmProcessId(id)
            .orElseThrow(() -> new
ObjectNotFoundException(Application.class.getSimpleName(), id));
    }
}

```

```

    }

    public void save(Application application) {
        applicationRepository.save(application);
    }
}

```

Листинг В.27 – Файл ApplicationHistoryService.java

```

@Service
@AllArgsConstructor
public class ApplicationHistoryService {
    private final ApplicationHistoryRepository applicationHistoryRepository;

    public void create(Application application, Status status) {
        applicationHistoryRepository.save(new ApplicationHistory(application,
status,
            new Timestamp(System.currentTimeMillis())));
    }

    public ApplicationHistory findByIdAndAndStatusName (UUID id,
String status) {
        return
applicationHistoryRepository.findByIdAndAndStatusName(id, status)
            .orElseThrow(() -> new
ObjectNotFoundException(Application.class.getSimpleName(), id));
    }
}

```

Листинг В.28 – Файл StudentService.java

```

@Service
@AllArgsConstructor
public class StudentService {
    private final StudentRepository studentRepository;

    public Student findById(String id) {
        return studentRepository
            .findById(id)
            .orElseThrow(() -> new
ObjectNotFoundException(Student.class.getSimpleName(), id));
    }
}

```

Листинг В.29 – Файл ApplicationComputing.java

```

@SpringBootApplication
@EnableConfigurationProperties({AnConfiguration.class,
DevConfiguration.class, TestConfiguration.class})
public class ApplicationComputing {

    public static void main(String... args) {
        SpringApplication.run(ApplicationComputing.class, args);
    }
}

```

Листинг В.30 – Файл Dockerfile

```

FROM openjdk:11
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]

```

Листинг В.31 – Файл docker-compose.yml

```
version: '3'

services:
  camunda:
    image: camunda/camunda-bpm-platform:${DISTRO:-latest}
    ports:
      - "8080:8080"
    restart: unless-stopped

  postgres:
    image: postgres
    volumes:
      - ./docker-postgresql-multiple-databases:/docker-entrypoint-initdb.d
    environment:
      - POSTGRES_MULTIPLE_DATABASES="training-center-applications",
"training-center-bpm"
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
```

Листинг В.32 – Файл XlsxGenerator.java

```
@Component
@AllArgsConstructor
@DependsOn("applicationConfiguration")
public class XlsxGenerator {
    private final ApplicationRepository applicationRepository;

    private final ApplicationHistoryRepository applicationHistoryRepository;

    private final StudentRepository studentRepository;

    private final AdvertisementRepository advertisementRepository;

    private final CuratedStudentRepository curatedStudentRepository;

    private final CuratorService curatorService;

    private final AnConfiguration anConfiguration;

    private final DevConfiguration devConfiguration;

    private final TestConfiguration testConfiguration;

    public void generate() throws IOException {
        XSSFWorkbook workbook = new XSSFWorkbook();
        CreationHelper creationHelper = workbook.getCreationHelper();

        XSSFSheet statisticsSheet =
workbook.createSheet(STATISTICS_SHEET_NAME);

        Row row = statisticsSheet.createRow(0);
        addCellsInRow(row,
Arrays.asList(creationHelper.createRichTextString(""),
creationHelper.createRichTextString(AN),
creationHelper.createRichTextString(DEV),
creationHelper.createRichTextString(TEST)));

        row = statisticsSheet.createRow(1);
        addCellsInRow(row,
```

```

creationHelper.createRichTextString(PASSED_TEST_PERCENT),
Arrays.asList(calculatePassedTestPercent(AN_COURSE_NAME),
              calculatePassedTestPercent(DEV_COURSE_NAME),
              calculatePassedTestPercent(TEST_COURSE_NAME)));

        row = statisticsSheet.createRow(2);
        addCellsInRow(row,
creationHelper.createRichTextString(COMPETITION_LEVEL), Arrays.asList(
              calculateCompetitionLevel(AN_COURSE_NAME,
anConfiguration.getMaxStudentsAmount()),
              calculateCompetitionLevel(DEV_COURSE_NAME,
devConfiguration.getMaxStudentsAmount()),
              calculateCompetitionLevel(TEST_COURSE_NAME,
testConfiguration.getMaxStudentsAmount())));

        row = statisticsSheet.createRow(3);
        addCellsInRow(row,
creationHelper.createRichTextString(APPLICATIONS_AMOUNT), Arrays.asList(
              calculateApplicationsToCourse(AN_COURSE_NAME),
              calculateApplicationsToCourse(DEV_COURSE_NAME),
              calculateApplicationsToCourse(TEST_COURSE_NAME)));

        createChart(statisticsSheet, 0, 5, 2, 11, APPLICATIONS_AMOUNT,
              XDDFDataSourcesFactory.fromStringCellRange(statisticsSheet,
new CellRangeAddress(0, 0, 1, 3)),
              XDDFDataSourcesFactory.fromNumericCellRange(statisticsSheet,
new CellRangeAddress(3, 3, 1, 3)));

        row = statisticsSheet.createRow(12);

        Row advertisementValuesRow = statisticsSheet.createRow(13);

advertisementValuesRow.createCell(0).setCellValue(creationHelper.createRichTextString(STUDENTS_CAME_FROM_ADVERTISEMENT));
        int i = 0;
        for (var advertisementStatistic :
calculateAdvertisementStatistics().entrySet()) {
            i++;

row.createCell(i).setCellValue(creationHelper.createRichTextString(advertisementStatistic.getKey()));

advertisementValuesRow.createCell(i).setCellValue(advertisementStatistic.getValue());
        }

        createChart(statisticsSheet, 0, 15, 2, 23, ADVERTISEMENT,
              XDDFDataSourcesFactory.fromStringCellRange(statisticsSheet,
new CellRangeAddress(12, 12, 1, i)),
              XDDFDataSourcesFactory.fromNumericCellRange(statisticsSheet,
new CellRangeAddress(13, 13, 1, i)));

        row = statisticsSheet.createRow(24);
        addCellsInRow(row,
creationHelper.createRichTextString(AVERAGE_STUDENT_AGE),
Collections.singletonList(calculateAverageStudentAge()));

        XSSFSheet curatedStudentsSheet =
workbook.createSheet(CURATED_STUDENTS);

        row = curatedStudentsSheet.createRow(0);
        addCellsInRow(row,
Arrays.asList(creationHelper.createRichTextString(STUDENT),
creationHelper.createRichTextString(CURATOR),

```

```

        creationHelper.createRichTextString(COURSE)));

    i = 0;
    for (var curatedStudent : curatedStudentRepository.findAll()) {
        i++;
        row = curatedStudentsSheet.createRow(i);
        addCellsInRow(row,
Arrays.asList(creationHelper.createRichTextString(curatedStudent.getStudentEmail()),

creationHelper.createRichTextString(curatedStudent.getCuratorEmail()),
        creationHelper.createRichTextString(getCourseName(
curatorService.findById(curatedStudent.getCuratorEmail()).getCourse().getName
()))));
    }
    try (OutputStream fileOut = new FileOutputStream(XLSX_FILE_NAME)) {
        workbook.write(fileOut);
    }
}

private int calculateApplicationsToCourse(String course) {
    return applicationRepository.findAllByCourseName(course).size();
}

private double calculateCompetitionLevel(String course, int maxAmount) {
    return (double) calculateApplicationsToCourse(course) / maxAmount;
}

private double calculatePassedTestPercent(String course) {
    int passed = 0;
    List<Application> applications =
applicationRepository.findAllByCourseName(course);
    for (var a : applications) {
        if
(applicationHistoryRepository.findByIdAndAndStatusName(a.getId(),
TEST_PASSED).isPresent()) {
            passed++;
        }
    }
    return applications.size() == 0 ? 0 : (double) passed /
applications.size() * 100;
}

private Map<String, Integer> calculateAdvertisementStatistics() {
    Map<String, Integer> statistics = new HashMap<>();
    for (var a : advertisementRepository.findAll()) {
        statistics.put(a.getName(),
studentRepository.findAllByAdvertisementName(a.getName()).size());
    }
    return statistics
        .entrySet()
        .stream()

.sorted(Collections.reverseOrder(Map.Entry.comparingByValue()))
        .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue, (e1, e2) -> e1, HashMap::new));
}

private String getCourseName(String course) {
    switch (course) {
        case AN_COURSE_NAME:
            return AN;
        case DEV_COURSE_NAME:

```

```

        return DEV;
    case TEST_COURSE_NAME:
        return TEST;
    default:
        return "";
    }
}

private double calculateAverageStudentAge() {
    int age = 0;
    for (var student: studentRepository.findAll()) {
        age += student.getAge();
    }
    return age / ((List<Student>) studentRepository.findAll()).size();
}

private void createChart(XSSFSheet sheet, int firstColumn, int firstRow,
int secondColumn, int secondRow,
                        String title, XDDFDataSource<String> keys,
XDDFDataSource<Double> values) {
    XSSFDrawing drawing = sheet.createDrawingPatriarch();
    XSSFClientAnchor anchor = drawing.createAnchor(0, 0, 0, 0,
firstColumn, firstRow, secondColumn,
secondRow);
    XSSFChart chart = drawing.createChart(anchor);
    chart.setTitleText(title);
    chart.setTitleOverlay(false);
    XDDFChartLegend legend = chart.getOrAddLegend();
    legend.setPosition(LegendPosition.TOP_RIGHT);
    XDDFChartData chartData = chart.createData(ChartTypes.PIE3D, null,
null);
    chartData.setVaryColors(true);
    chartData.addSeries(keys, (XDDFNumericalDataSource<? extends Number>)
values);
    chart.plot(chartData);
}

private void addCellsInRow(Row row, List<RichTextString> values) {
    int i = 0;
    for (var value : values) {
        row.createCell(i).setCellValue(value);
        i++;
    }
}

private void addCellsInRow(Row row, RichTextString firstCellValue,
List<Object> values) {
    row.createCell(0).setCellValue(firstCellValue);
    int i = 1;
    for (var value : values) {
        if (value instanceof Double) {
            row.createCell(i).setCellValue((Double) value);
        } else if (value instanceof Integer) {
            row.createCell(i).setCellValue((Integer) value);
        }
        i++;
    }
}
}

```

Листинг В.33 – Файл WorkflowTest.java

```

public class WorkflowTest {
    @Rule

```

```

public ProcessEngineRule processEngineRule = new ProcessEngineRule();

@Rule
public final ExpectedException thrown = ExpectedException.none();

@Rule
@ClassRule
public static TestCoverageProcessEngineRule rule =
TestCoverageProcessEngineRuleBuilder.create().build();

private ProcessScenario processScenario = mock(ProcessScenario.class);

private static final String INTERVIEW_CONFIRMATION_TIMEOUT_PERIOD =
"P2DT1M";

private static final String INTERVIEW_CONFIRMATION_PERIOD = "P1DT1M";

@Deployment(resources = DEPLOYMENT)
public void testDeployment() {
}

@Before
public void init() throws Exception {
    DelegateExpressions.registerJavaDelegateMock(MOCK);
    final JavaDelegate mock =
DelegateExpressions.getJavaDelegateMock(MOCK);
    mock.execute(mock(DelegateExecution.class));
}

private void initDelegatesAfterTestIsPassed() throws Exception {
DelegateExpressions.registerJavaDelegateMock(SENT_APPLICATION_EXECUTOR);
    final JavaDelegate sentApplicationProcessor =
DelegateExpressions.getJavaDelegateMock(SENT_APPLICATION_EXECUTOR);
    sentApplicationProcessor.execute(mock(DelegateExecution.class));

    DelegateExpressions
        .registerJavaDelegateMock(TEST_RESULTS_EXECUTOR)
        .onExecutionSetVariable(TEST_IS_PASSED_VAR_NAME, true);
    final JavaDelegate testResultsProcessor =
DelegateExpressions.getJavaDelegateMock(TEST_RESULTS_EXECUTOR);
    testResultsProcessor.execute(mock(DelegateExecution.class));

DelegateExpressions.registerTaskListenerMock(INTERVIEW_SETTING_LISTENER);
    final TaskListener interviewSettingListener =
DelegateExpressions.getTaskListenerMock(INTERVIEW_SETTING_LISTENER);
    interviewSettingListener.notify(mock(DelegateTask.class));
}

private void initDelegatesAfterInterviewIsConfirmed() throws Exception {
DelegateExpressions.registerTaskListenerMock(INTERVIEW_CREATE_LISTENER);
    final TaskListener interviewCreateListener =
DelegateExpressions.getTaskListenerMock(INTERVIEW_CREATE_LISTENER);
    interviewCreateListener.notify(mock(DelegateTask.class));

DelegateExpressions.registerTaskListenerMock(INTERVIEW_COMPLETE_LISTENER);
    final TaskListener interviewCompleteListener =
DelegateExpressions.getTaskListenerMock(INTERVIEW_COMPLETE_LISTENER);
    interviewCompleteListener.notify(mock(DelegateTask.class));
}

```

```

@Test
@Deployment(resources = DEPLOYMENT)
public void testApplicationValidationFailure() throws Exception {
    thrown.expect(BpmnError.class);

    DelegateExpressions
        .registerJavaDelegateMock(SENT_APPLICATION_EXECUTOR)
        .onExecutionThrowBpmnError(APPLICATION_VALIDATION_ERROR_CODE);
    final JavaDelegate sentApplicationProcessor =
        DelegateExpressions.getJavaDelegateMock(SENT_APPLICATION_EXECUTOR);
    sentApplicationProcessor.execute(mock(DelegateExecution.class));

    ProcessInstance processInstance = processEngine()
        .getRuntimeService()
        .startProcessInstanceByKey(PROCESS_KEY);

    assertThat(processInstance)
        .isStarted()
        .hasPassed(PROCESS_APPLICATION_TASK)
        .hasPassed(INFORM_APPLICATION_NOT_VALID_TASK)
        .isEnded();
}

@Test
@Deployment(resources = DEPLOYMENT)
public void testTestNotPassed() throws Exception {

    DelegateExpressions.registerJavaDelegateMock(SENT_APPLICATION_EXECUTOR);
    final JavaDelegate sentApplicationProcessor =
        DelegateExpressions.getJavaDelegateMock(SENT_APPLICATION_EXECUTOR);
    sentApplicationProcessor.execute(mock(DelegateExecution.class));

    DelegateExpressions
        .registerJavaDelegateMock(TEST_RESULTS_EXECUTOR)
        .onExecutionSetVariable(TEST_IS_PASSED_VAR_NAME, false);
    final JavaDelegate testResultsProcessor =
        DelegateExpressions.getJavaDelegateMock(TEST_RESULTS_EXECUTOR);
    testResultsProcessor.execute(mock(DelegateExecution.class));

    ProcessInstance processInstance = processEngine()
        .getRuntimeService()
        .startProcessInstanceByKey(PROCESS_KEY);

    assertThat(processInstance)
        .isStarted()
        .hasPassed(PROCESS_APPLICATION_TASK)
        .hasPassed(PROCESS_TEST_RESULTS_TASK)
        .hasPassed(TEST_GATEWAY)
        .hasPassed(INFORM_TEST_NOT_PASSED_TASK)
        .isEnded();
}

@Test
@Deployment(resources = DEPLOYMENT)
public void testInterviewNotConfirmed() throws Exception {
    initDelegatesAfterTestIsPassed();

    when(processScenario.waitsAtUserTask(SET_UP_INTERVIEW_TASK)).thenReturn(task
    -> task.complete(withVariables(INTERVIEW_IS_PASSED_VAR_NAME, "false")));

    when(processScenario.waitsAtReceiveTask(WAIT_FOR_CONFIRMATION_TASK))
        .thenReturn((receiveTask) ->
    receiveTask.defer(INTERVIEW_CONFIRMATION_TIMEOUT_PERIOD,

```

```

receiveTask::receive));

    Scenario.run(processScenario)
        .startByKey(PROCESS_KEY)
        .execute();

    verify(processScenario).hasCompleted(PROCESS_APPLICATION_TASK);
    verify(processScenario).hasCompleted(PROCESS_TEST_RESULTS_TASK);
    verify(processScenario).hasCompleted(TEST_GATEWAY);
    verify(processScenario).hasCompleted(SET_UP_INTERVIEW_TASK);
    verify(processScenario).hasCanceled(WAIT_FOR_CONFIRMATION_TASK);
    verify(processScenario).hasCompleted(INFORM_CONFIRMATION_TIMEOUT_TASK);
}

@Test
@Deployment(resources = DEPLOYMENT)
public void testInterviewNotPassed() throws Exception {
    initDelegatesAfterTestIsPassed();

    initDelegatesAfterInterviewIsConfirmed();

    when(processScenario.waitsAtUserTask(SET_UP_INTERVIEW_TASK)).thenReturn(task
-> task.complete(withVariables(INTERVIEW_IS_PASSED_VAR_NAME, "false")));

    when(processScenario.waitsAtReceiveTask(WAIT_FOR_CONFIRMATION_TASK))
        .thenReturn((receiveTask) ->
receiveTask.defer(INTERVIEW_CONFIRMATION_PERIOD, receiveTask::receive));

    when(processScenario.waitsAtUserTask(DO_INTERVIEW_TASK)).thenReturn(TaskDeleg
ate::complete);

    Scenario.run(processScenario)
        .startByKey(PROCESS_KEY)
        .execute();

    verify(processScenario).hasCompleted(PROCESS_APPLICATION_TASK);
    verify(processScenario).hasCompleted(PROCESS_TEST_RESULTS_TASK);
    verify(processScenario).hasCompleted(TEST_GATEWAY);
    verify(processScenario).hasCompleted(SET_UP_INTERVIEW_TASK);
    verify(processScenario).hasCompleted(WAIT_FOR_CONFIRMATION_TASK);
    verify(processScenario).hasCompleted(DO_INTERVIEW_TASK);
    verify(processScenario).hasCompleted(INFORM_INTERVIEW_NOT_PASSED_TASK);
}

@Test
@Deployment(resources = DEPLOYMENT)
public void testInterviewIsPassed() throws Exception {
    initDelegatesAfterTestIsPassed();

    initDelegatesAfterTestIsPassed();

    DelegateExpressions.registerJavaDelegateMock(CURATORS_ASSIGNMENT_EXECUTOR);
    final JavaDelegate curatorsAssignmentExecutor =
    DelegateExpressions.getJavaDelegateMock(CURATORS_ASSIGNMENT_EXECUTOR);
    curatorsAssignmentExecutor.execute(mock(DelegateExecution.class));

    when(processScenario.waitsAtUserTask(SET_UP_INTERVIEW_TASK)).thenReturn(task
-> task.complete(withVariables(INTERVIEW_IS_PASSED_VAR_NAME, "true")));

    when(processScenario.waitsAtReceiveTask(WAIT_FOR_CONFIRMATION_TASK))

```

```

        .thenReturn((receiveTask) ->
receiveTask.defer(INTERVIEW_CONFIRMATION_PERIOD, receiveTask::receive));

when(processScenario.waitsAtUserTask(DO_INTERVIEW_TASK)).thenReturn(TaskDeleg
ate::complete);

        Scenario.run(processScenario)
            .startByKey(PROCESS_KEY)
            .execute();

        verify(processScenario).hasCompleted(PROCESS_APPLICATION_TASK);
        verify(processScenario).hasCompleted(PROCESS_TEST_RESULTS_TASK);
        verify(processScenario).hasCompleted(TEST_GATEWAY);
        verify(processScenario).hasCompleted(SET_UP_INTERVIEW_TASK);
        verify(processScenario).hasCompleted(WAIT_FOR_CONFIRMATION_TASK);
        verify(processScenario).hasCompleted(DO_INTERVIEW_TASK);
        verify(processScenario).hasCompleted(INFORM_INTERVIEW_PASSED_TASK);
        verify(processScenario).hasCompleted(ASSIGN_CURATOR_TASK);
    }

    @Test
    @Deployment(resources = DEPLOYMENT)
    public void testApplicationCancellation() throws Exception {
        initDelegatesAfterTestIsPassed();

        ProcessInstance processInstance = processEngine()
            .getRuntimeService()
            .startProcessInstanceByKey(PROCESS_KEY);

        processEngine().getRuntimeService().createMessageCorrelation(APPLICATION_CANC
ELLATION_MESSAGE)
            .correlate();

        assertThat(processInstance)
            .isStarted()
            .hasPassed(APPLICATION_CANCELLATION_TASK)
            .isEnded();
    }
}

```

Листинг В.34 – Файл TestConstants.java

```

public class TestConstants {
    static final String DEPLOYMENT = "process.bpmn";

    static final String MOCK = "mock";

    static final String SENT_APPLICATION_EXECUTOR =
"sentApplicationExecutor";

    static final String TEST_RESULTS_EXECUTOR = "testResultsExecutor";

    static final String INTERVIEW_SETTING_LISTENER =
"interviewSettingListener";

    static final String INTERVIEW_CREATE_LISTENER =
"interviewCreateListener";

    static final String INTERVIEW_COMPLETE_LISTENER =
"interviewCompleteListener";

    static final String PROCESS_APPLICATION_TASK = "process-application";
}

```

```
    static final String INFORM_APPLICATION_NOT_VALID_TASK = "inform-  
application-not-valid";  
  
    static final String PROCESS_TEST_RESULTS_TASK = "process-test-results";  
  
    static final String TEST_GATEWAY = "test-gateway";  
  
    static final String INFORM_TEST_NOT_PASSED_TASK = "inform-test-not-  
passed";  
  
    static final String SET_UP_INTERVIEW_TASK = "set-up-interview";  
  
    static final String WAIT_FOR_CONFIRMATION_TASK = "wait-for-confirmation";  
  
    static final String INFORM_CONFIRMATION_TIMEOUT_TASK = "inform-  
confirmation-timeout";  
  
    static final String DO_INTERVIEW_TASK = "do-interview";  
  
    static final String INFORM_INTERVIEW_NOT_PASSED_TASK = "inform-interview-  
not-passed";  
  
    static final String CURATORS_ASSIGNMENT_EXECUTOR =  
"curatorsAssignmentExecutor";  
  
    static final String INFORM_INTERVIEW_PASSED_TASK = "inform-interview-  
passed";  
  
    static final String ASSIGN_CURATOR_TASK = "assign-curator";  
  
    static final String APPLICATION_CANCELLATION_TASK = "application-  
cancellation";  
}
```