Федеральное государственное автономное
образовательное учреждение
высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Легалов
подпись инициалы, фамилия
« _____» _____ 2016 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
Код и наименование направления

Система автоматизации контроля уровня квалификации персонала
Тема

Пояснительная записка

Руководитель _____ профессор, д.т.н А. И. Легалов
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Е. В. Ророт
подпись, дата инициалы, фамилия

Нормоконтролер _____ В. И. Иванов
подпись, дата инициалы, фамилия

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время обучение и повышение квалификации персонала приобретает особое значение и становится неотъемлемой частью успешно функционирующего предприятия. Учитываю всю специфику рынка IT – технологий, особенностью которого является быстрые и частые изменения, как внешних условий предприятия, так и внутренних, можно констатировать, что развитие системы профессионально обучения в компании определяет не только ее успешное развитие, но и выживаемость на рынке. Для того чтобы развивать сотрудников, необходимо умение их оценивать, на базе этой оценки можно строить планы по развитию персонала. В IT – секторе основной капитал компании – это высококвалифицированные специалисты, поэтому руководству жизненно необходимо знать, какую работу и кому из сотрудников можно поручить. Также и сами сотрудники, работающие в IT – индустрии уделяют много внимания своему профессиональному развитию, поэтому для адекватной оценки навыков, нужны адаптированные под потребности отрасли методы. Чем более индивидуальные услуги предоставляет компания своим клиентам, тем более квалифицированные специалисты должны в ней работать. Поэтому компании нужны собственные разработки, стандартные методы оценки персонала им не подходят, к своим внутренним ресурсам они должны подходить так же уважительно, как и к внешним. При этом компания должна уметь не только объективно оценивать текущий уровень квалификации каждого сотрудника, но и разрабатывать для него план дальнейшего профессионального развития и карьерного роста.

Для выполнения задачи по обучению и развитию персонала как раз и необходимо иметь возможность использовать вспомогательный ресурс в виде системы, которая будет помогать контролировать обучение и развитие персонала, а также содержать в себе модули, отвечающие за прохождение сотрудниками обучающего материала, который может быть представлен в виде внешних и внутренних ресурсов. Так же система должна позволять

отслеживать, планировать рост и квалификацию, как самим сотрудником, так и ответственным людям за обучение персонала в компании.

Существующие аналоги систем обучения и развития персонала, как на российском рынке, так и на зарубежном, могут отвечать описанным критериям, но главным отличием большинства систем является то, что они не могут соответствовать специфики именно IT-компаний, компаний занимающихся разработкой программного обеспечения. Все эти системы подогнаны под усреднённые задачи и профессии. Примерами таких систем являются WebTutur[7] и MoorepayHr [8]. Обзор этих систем представлен в разделе 1.6.

Целью выпускной квалификационной работы является разработка веб-ориентированной системы, которая позволит компании производить обучение персонала посредством выбора сотрудником курса, а также реализация функциональности, позволяющей осуществлять контроль за прохождением обучения.

Задачи, которые должен будет решать программный продукт:

1. Обучение сотрудников компании.
2. Тестирование персонала.
3. Управление планированием и развитием карьеры сотрудников, как со стороны руководства, так и со стороны самих сотрудников.
4. Контроль знаний сотрудников.
5. Разработка обучающих материалов.

Для достижения поставленных целей необходимо будет решить следующие задачи:

1. Предоставление пользовательского интерфейса для сотрудников компании:
    - Регистрация;
    - Авторизация;
    - Управление учетной записью пользователя.

2. Для сотрудников ответственных за обучение должна быть реализована функциональность планирования обучения, которая будет включать в себя:
    - Добавление курсов, курсы создаются из материалов, по каждому курсу должен быть итоговый тест;
    - Добавление материалов, материалу может соответствовать тест;
    - Просмотр списка сотрудников;
    - Просмотр информации о сотруднике (ФИО, должность, специализация, квалификация, достижения);
    - Контроль прохождения курсов у сотрудников;
    - Формирование и анализ результатов обучения сотрудников, подготовка отчетов;
    - Ведение базы учебных программ;
    - Импорт базы содрудников из excel.
3. Для сотрудников, проходящих обучение, реализовать следующую функциональность:
    - Возможность просматривать свой учебный план.
    - Выбор курсов для обучения
    - Возможность изучать курсы, материалы.
    - Прохождение тестов по курсам, материалам.
    - Просмотр информации о себе (ФИО, должность, специализация, квалификация, достижения).
    - Просмотр своего карьерного роста.

Первый раздел выпускной квалификационной работы посвящен описанию особенностей систем контроля и управления персонала, в нем так же рассматривается некоторые уже существующие системы обучения и контроля персонала.

Второй раздел посвящен краткому описанию архитектуры разрабатываемого приложения.

В третьем разделе рассматривается программная реализация проекта.

В четвертом разделе показана функциональная часть выпускной квалификационной работы.

1 Особенности систем контроля и управления персоналом

1.1 Основные функции

Система управления персоналом — это совокупность приемов, методов, технологий организации работы с персоналом [1].

Первоначальным этапом проектирования и формирования системы управления персоналом организации является формулировка целей данной системы. Для различных организаций цели системы управления персоналом варьируются в зависимости от характера деятельности организации, объемов производства, стратегических задач и т.д. Обобщение опыта зарубежных и отечественных организаций позволяет сформулировать главную цель системы управления персоналом организации как обеспечение организации персоналом, их эффективное использование, профессиональное и социальное развитие. На рисунке 1.1 показана структура целей системы управления персоналом организации.

Рисунок 1.1 – Цели системы управления персоналом

Система управления персоналом делиться на функции:

Функция планирования персонала заключается в разработке кадровой политики и стратегии управления персоналом; анализе кадрового потенциала организации и рынка труда; организации кадрового планирования и прогнозирования потребности в персонале; поддержании взаимосвязей с внешними источниками, обеспечивающими организацию кадрами.

Функция оценки, обучения и развития персонала заключается в осуществлении обучения, переподготовки и в повышении квалификации персонала; введении в должность и адаптации новых работников; организации и проведении мероприятий по оценке персонала; управлении развитием карьеры.

Функция информационного обеспечения управления персоналом заключается в ведении учета и статистики персонала; информационном и техническом обеспечении системы управления персоналом; обеспечении персонала необходимой для работы научно-технической информацией.

1.2 Особенности обучения персонала

1.2.1 Цели обучения персонала

Обучение персонала — важная составляющая успешности большинства фирм. Появление новых технологий, внедрение в производство новой техники и оборудования требуют соответствующей квалификации работников. Своевременное овладение работников знаниями, умениями и навыками обеспечит эффективное развитие и поддержание конкурентоспособности организации [2].

Важность непрерывного образования подтверждают следующие основные факторы.

1. Внедрение новой техники, технологии, производство современных товаров, рост коммуникационных возможностей.

2. Мир превращается в рынок с высоким уровнем конкуренции между странами. Страны, имеющие современную систему инженерного труда и программы непрерывного образования, лидируют в условиях этой конкуренции.
3. Непрерывные и быстрые изменения в технологии и информатике требуют непрерывного обучения персонала.
4. Для фирмы более эффективно и экономично повышение отдачи от уже работающих сотрудников на основе их непрерывного обучения, чем привлечение новых работников.

Цели обучения, которые преследует работодатель:

- организация и формирование персонала управления;
- овладение умением определять, понимать и решать проблемы;
- воспроизводство персонала;
- интеграция персонала;
- гибкое формирование персонала;
- адаптация;
- внедрение нововведений.

Цели непрерывного образования с позиции работника:

- поддержание на соответствующем уровне и повышение профессиональной квалификации;
- приобретение профессиональных знаний вне сферы профессиональной деятельности;
- приобретение профессиональных знаний о поставщиках и потребителях продукции, банках и других организациях, влияющих на работу фирмы;
- развитие способностей в области планирования и организации производства.

1.2.2 Виды обучения персонала

Различаются три вида обучения: подготовка, повышение квалификации и переподготовка персонала.

Подготовка персонала — планомерное и организованное обучение и выпуск квалифицированных кадров для всех областей человеческой деятельности, владеющих совокупностью специальных знаний, умений, навыков и способов общения [3].

Повышение квалификации персонала — обучение кадров в целях усовершенствования знаний, умений, навыков и способов общения в связи с ростом требований к профессии или повышением в должности [3].

Переподготовка персонала — обучение кадров в целях освоения новых знаний, умений, навыков и способов общения в связи с овладением новой профессией или изменившимися требованиями к содержанию и результатам труда [3].

Предметом обучения являются:

- знания — получение теоретических, методических и практических знаний, необходимых работнику для выполнения своих обязанностей на рабочем месте;
- умения — способность выполнять обязанности, закрепленные за работником на конкретном рабочем месте;
- навыки — высокая степень умения применять полученные знания на практике, навыки предполагают такую меру освоения работы, когда вырабатывается сознательный самоконтроль;
- способы общения (поведения), форма жизнедеятельности личности — совокупность действий и поступков индивида в процессе общения с окружающей действительностью, выработка характера поведения, соответствующего требованиям, предъявляемым рабочим местом, социальные отношения, коммуникабельность.

Виды и методы обучения персонала:

Обучение без отрыва от производства осуществляется в обычной рабочей обстановке: обучаемый использует настоящие рабочие инструменты, оборудование, документацию или материалы, которые он будет использовать и после завершения курса обучения. При этом обучаемый работник рассматривается как частично производительный работник.

Обучение с отрывом от производства проводится вне рабочего места, как правило, с использованием специально упрощенных учебных инструментов и оборудования. Обучаемый работник не считается производительной единицей с момента начала обучения, его работа начинается с выполнения упражнений.

Методы обучения персонала на рабочем месте представлены в таблице 1.1.

Таблица 1.1 – Методы обучения персонала на рабочем месте

Методы обучения	Характерные особенности метода
Направленное приобретение опыта	Систематическое планирование обучения на рабочем месте, основу планирования составляет индивидуальный план профессионального обучения, в котором изложены цели обучения.
Производственный инструктаж	Общая информация, введение в специальность, адаптация, ознакомление обучающегося с новой рабочей обстановкой.
Смена рабочего места (ротация)	Получение знаний и приобретение опыта в результате систематической смены рабочего места. В результате этого за определенный промежуток времени создается представление о многогранности деятельности и производственных задач
Использование работников в качестве ассистентов, стажеров	Обучение и ознакомление работника с проблемами высшего и качественно иного порядка задач при одновременном принятии на себя некоторой доли ответственности.

Продолжение таблицы 1.1 – Методы обучения персонала на рабочем месте

Наставничество	Сотрудничество наставника и обучающегося, когда наставник обеспечивает непрерывную, беспристрастную обратную связь и периодически проверяет уровень исполнения работы наставляемых.
Подготовка в проектных группах	Сотрудничество, осуществляемое в учебных целях в проектных группах, создаваемых на предприятии для разработки крупных, ограниченных сроком задач.

Методы обучения персонала вне рабочего места представлены в таблице 1.2.

Таблица 1.2 – Методы обучения персонала вне рабочего места

Методы обучения	Характерные особенности метода
Чтение лекций	Пассивный метод обучения, используется для изложения теоретических и методических знаний, практического опыта.
Программированные курсы обучения	Более активный метод обучения, эффективен для получения теоретических знаний.
Конференции, семинары, беседы «за круглым столом», экскурсии, дискуссии, встречи с руководством	Активный метод обучения, участие в дискуссиях развивает логическое мышление и вырабатывает способы поведения в различных ситуациях.

Продолжение таблицы 1.2 – Методы обучения персонала вне рабочего места

Деловые игры	Обучение манере вести себя в различных производственных ситуациях, при ведении переговоров, причем обладатели ролей должны вырабатывать альтернативные точки зрения.
Тренинг	Ежедневное обучение, в ходе которого один инструктирует или тренирует другого относительно основ его деятельности путем интенсивного обучения.
Самостоятельное обучение	Наиболее простой вид обучения, для которого не требуется ни инструктор, ни специальное помещение, ни определенное время: обучающийся учится там, тогда и так, как ему удобно.

Кроме обучения на рабочем месте и вне его, возможно сочетание того и другого метода. К таким формам обучения можно отнести:

• опытное или эмпирическое обучение — обучение путем самостоятельной работы, но в некотором логическом порядке;

• демонстрация и практика под руководством — обучающий показывает стажеру, как делать, затем обучающий дает возможность сделать это самому работнику, но под его руководством;

• программируемое обучение — книга или машина, которая «ведет» читателя и периодически проверяет его знания постановкой вопросов;

• обучение с помощью компьютера — собственно программируемое обучение путем взаимодействия с компьютером, использование сети Интернет;

• обучение действием — обучение в ходе выполнения действий, например, участие вместе с другими в разработке проекта или группового задания, или работа «во втором составе» другого подразделения.

1.3 Развитие персонала

Развитие персонала — система взаимосвязанных действий, включающих выработку стратегии, прогнозирование и планирование потребности в персонале, управление карьерой и профессиональным ростом, организацию процесса адаптации, обучения, тренинга, формирование организационной культуры [4].

Развитие персонала является систематическим процессом, ориентированным на формирование сотрудников, отвечающих потребностям предприятия, и, в то же время, на изучение и развитие производительного и образовательного потенциала сотрудников предприятия.

Развитие персонала включает следующий комплекс мер:

- профессиональное обучение;
- переподготовка и повышение квалификации кадров;
- ротация;
- делегирование полномочий;
- планирование карьеры персонала в организации.

Профессиональное развитие представляет собой процесс подготовки сотрудника к выполнению новых производственных функций, занятию новых должностей, решению новых задач.

Цели развития персонала:

- повышение трудового потенциала работников для решения личных задач и задач в области функционирования и развития организации
- повышение эффективности труда;
- снижение текучести кадров;
- подготовка необходимых руководящих кадров;
- воспитание молодых способных сотрудников;
- достижение большей независимости рынка труда;
- адаптация к новым технологиям;

- рост социальных качеств сотрудников и их удовлетворенности трудом.

Меры по развитию персонала:

- сохранение работоспособности;
- адаптация персонала к изменяющимся условиям;
- подготовка сотрудников к выполнению более сложных задач;
- организация психологической помощи сотрудникам, работающим в условиях повышенных рисков.

Основные принципы развития персонала:

- целостность системы развития, преемственность различных видов и форм развития персонала;
- опережающий характер обучения и развития на основе прогноза научно- технического развития и условий развития организации;
- гибкость различных форм развития, возможность их использования на отдельных этапах развития;
- профессиональное и социальное стимулирование развития человеческих ресурсов;
- построение системы развития персонала с учетом конкретных возможностей организации, социально-экономических условий его функционирования.

Факторы, влияющие на необходимость развития персонала в современных условиях:

- серьезная конкуренция на различных рынках в условиях глобализации экономики;
- бурное развитие новых информационных технологий;
- системное, комплексное решение вопросов управления человеческими ресурсами и всех стратегических задач на основе единой программы деятельности организации;
- необходимость разработки стратегии и организационной культуры организации;

• участие всех линейных руководителей в реализации единой кадровой политики и решения стратегических задач организации;

• наличие широкой специализированной сети консультационных фирм по различным направлениям развития человеческих ресурсов.

1.4 Планирование карьеры

Карьера – это результат осознанной позиции и поведения человека в области трудовой деятельности, связанный с должностным или профессиональным ростом [5].

Карьеру – траекторию своего служебного движения – человек строит сам, сообразуясь с особенностями внутри и вне организационной реальности и главное — со своими собственными целями, желаниями и установками [5].

Планирование и контроль деловой карьеры заключаются в том, что с момента принятия работника в организацию и до предполагаемого увольнения с работы необходимо организовать планомерное горизонтальное и вертикальное продвижение работника по системе должностей или рабочих мест.

Планирование карьеры — одно из направлений кадровой работы в организации, ориентированное на определение стратегии и этапов развития и продвижения специалистов [6].

Продвижение по службе определяется не только личными качествами работника (образование, квалификация, отношение к работе, система внутренних мотиваций), но и объективными, в частности:

• высшая точка карьеры — высший пост, существующий в конкретной рассматриваемой организации;

• длина карьеры — количество позиций на пути от первой позиции, занимаемой индивидуумом в организации, до высшей точки;

• показатель уровня позиции — отношение числа лиц, занятых на следующем иерархическом уровне, к числу лиц, занятых на том

иерархическом уровне, где находится индивидуум в данный момент своей карьеры;

• показатель потенциальной мобильности — отношение (в некоторый определенный период времени) числа вакансий на следующем иерархическом уровне к числу лиц, занятых на том иерархическом уровне, где находится индивидуум.

Движение персонала по профессиональной лестнице:

Служебно-профессиональное продвижение - серия поступательных перемещений по различным должностям, способствующая развитию, как организации, так и личности.

Система служебно-профессионального продвижения - совокупность средств и методов должностного продвижения персонала, применяемых в различных организациях.

Продвижение персонала состоит из следующих процедур:

1.Повышение в должности или квалификации, когда служащий замещает более высокую должность, а рабочий получает новый разряд.

2. Перемещение, когда работник переводится на другое равноценное рабочее место (цех, отдел, служба) в силу производственной необходимости или изменения характера труда.

3. Понижение, когда в связи с изменением его потенциала работник переводится на более низкую должность или по результатам аттестации на более низкий разряд для рабочего.

4. Увольнение с предприятия, когда работник полностью меняет место работы в связи с неудовлетворенностью условиями труда или несоответствия занимаемому рабочему месту.

1.5 Система квалификаций разработчиков ПО в компании Aspirity

На рисунке 1.2 показана система квалификаций разработчиков, которая используется в компании Aspirity.

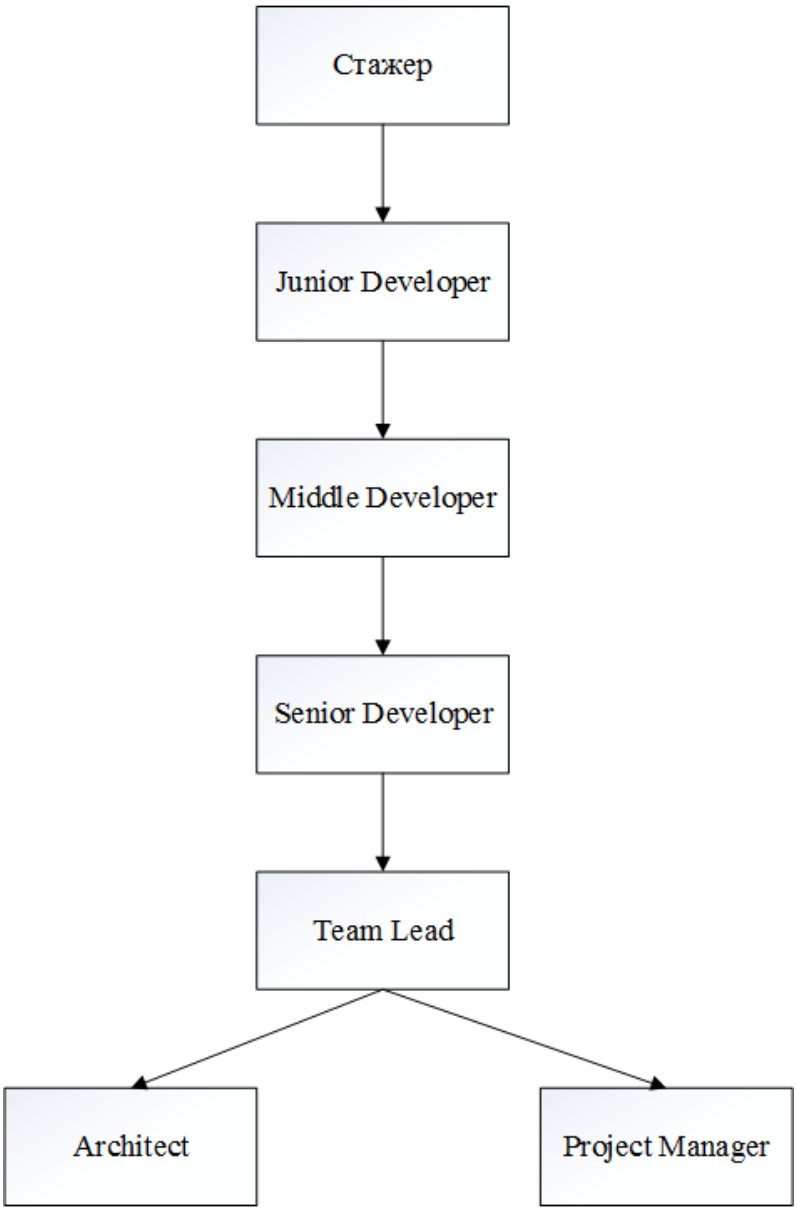

Рисунок 1.2 – Система квалификаций

1.6 Обзор существующих систем

1.6.1 WebTutor

WebTutor — система комплексной автоматизации HR-процессов [7]. Систему WebTutor отличает модульный подход, позволяющий создавать на базе набора программных модулей гибко настраиваемые системы, функционал которых зависит от задач, стоящих перед заказчиком.

Основные модули WebTutor:

1. Подбор персонала - в данном модуле реализуются функции, позволяющие руководителям подразделений через веб-интерфейс размещать вакансии, знакомиться с резюме, назначать тестирование и т.п., а работникам кадровых служб – вести поиск кандидатов и работу с ними.

2. Портал знаний - позволяет вести работу с персоналом, управлять процессом накопления знаний в компании, предоставить сотрудникам компании доступ к различным информационным материалам, содержит систему управления контентом портала (CMS), а также включает в себя ряд функций, необходимых для построения полнофункционального корпоративного портала.

3. Обучение и развитие:

- Обучение персонала с помощью электронных учебных курсов
- Автоматизацию деятельности корпоративного или внешнего учебного центра
- Проведение обучающих семинаров через интернет
- Выявление, воспитание и удержание талантливых сотрудников
- Создание электронных курсов

4. Оценка персонала - автоматизирует процесс оценки, включая планирование и проведение оценочных процедур и аттестаций. Модуль поддерживает разные виды оценки, включая оценку по компетенциям,

оценку эффективности деятельности, управление по целям, оценку должностей.

5. Тестирование:

- Редактор тестов, обладающий следующими возможностями:
- Поддержка форматированных тестовых вопросов (таблицы, графика, видео и т.п.)
- Поддержка различных типов тестовых вопросов
- Поддержка адаптивной методики тестирования
- Гибкие возможности настройки тестов
- Создание и ведение базы вопросов неограниченного объема
- Автоматическое создание тестов из отобранного набора вопросов
- Контроль прохождения тестов
- Получение отчетов о результатах тестирования

Интерфейс портала, представленный на рисунке 1.3, предназначен для организации единой точки доступа пользователей (участников бизнес-процессов, автоматизируемых программным комплексом) к различным ресурсам и сервисам программного комплекса.

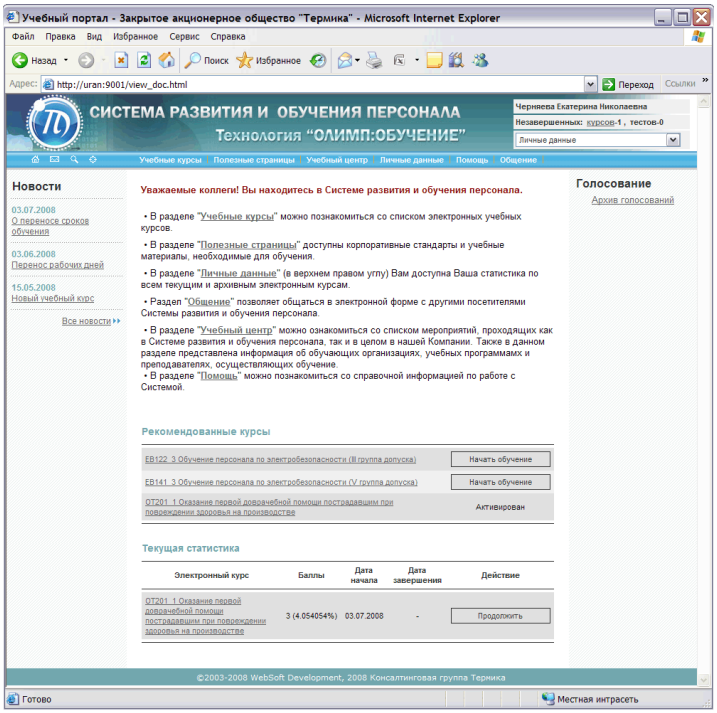

Рисунок 1.3 – Интерфейс портала WebTutor

С помощью интерфейса Портал пользователи в соответствии со своей функциональной ролью выполняют различные действия, например:

- прохождение тестирования и обучения;
- заполнение различных форм при участии в оценочных процедурах;
- получение информации;
- управление процессами обучения, тестирования, оценки персонала;
- получение отчетов.

Интерфейс администратора, представленный на рисунке 1.4, предназначен для решения основных задач по администрированию программного комплекса WebTutor.

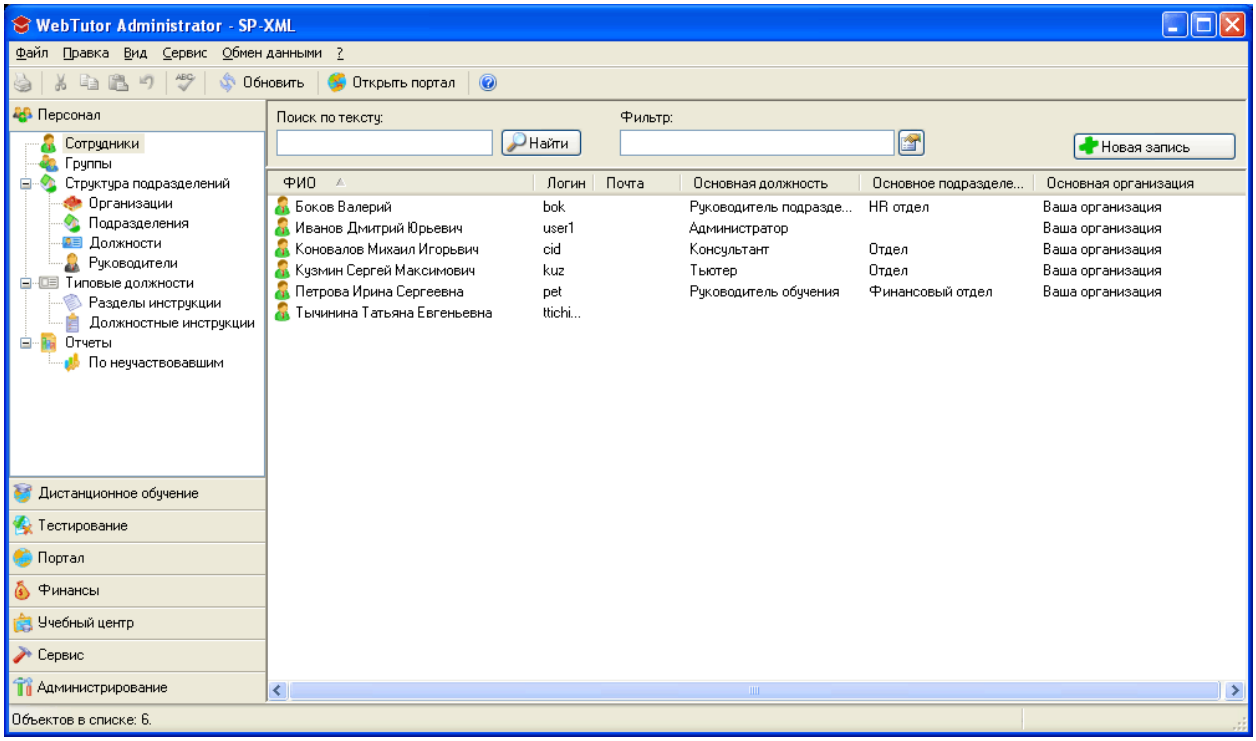

Рисунок 1.4 – Интерфейс администратора WebTutor

1.6.2 MoorepayHr

Moorepayhr – система разработана для малого и среднего бизнеса [8]. Система является модульной. Она имеет ряд модулей, которые способны

помочь управлению сотрудником с момента его найма на работу до момента его ухода из компании.

Модули [8]:

- Personnel
- Employee Self Service
- Safety
- Performance Management
- Talent Management
- Learning
- External Personnel
- Recruitment
- Remuneration
- Reporting
- Customisable Workflow
- Time & Attendance

Сервис поставляется через облачные технологии, не требуется загрузки или установки, и какого либо программного обеспечения и дополнительного оборудования.

2 Архитектура разрабатываемой системы

2.1 Общая Архитектура приложения

Разрабатываемое приложение использует трехуровневую архитектурную модель, представленную на рисунке 2.1.

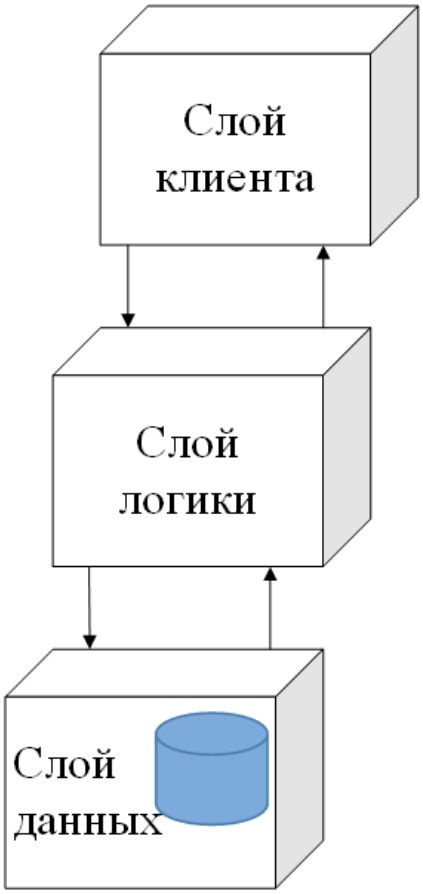

Рисунок 2.1 – Структура приложения

Слой клиента: самый верхний уровень приложения с интерфейсом пользователя. Главная функция интерфейса предоставление задач и результатов, понятных пользователю. Данный слой работает под управлением браузера.

Слой логики: этот слой координирует программу, обрабатывает команды, выполняет логические решения и вычисления, выполняет расчеты.

Он также перемещается и обрабатывает данные между двумя окружающими слоями. Работает под управлением веб – сервера.

Слой данных: здесь хранится информация и извлекается из базы данных и файловой системы. Информация отправляется в логический слой для обработки и в конечном итоге возвращается пользователю.

Приложение разработано с использованием фреймворка Express.JS.

Разрабатываемая система использует клиент – серверную модель взаимодействия. Работа системы изображена на рисунке N. Взаимодействие клиентского уровня и серверного происходит с помощью GET и POST запросов. GET запросы используются только для получения данных и не изменяют состояние системы. При помощи POST запросов происходит изменения данных хранящихся на сервере.

Для того чтобы сервер взаимодействовал с сервером базы данных MongoDB, используется библиотека Mongoose [11].

Mongoose – это ORM для MongoDB сделанная под node.js.

ORM [12] (англ. Object-Relational Mapping, рус. объектно-реляционное отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии. При использовании ORM запросы выполняются при помощи собственного синтаксиса, что позволяет абстрагироваться от конкретных реализация СУБД.

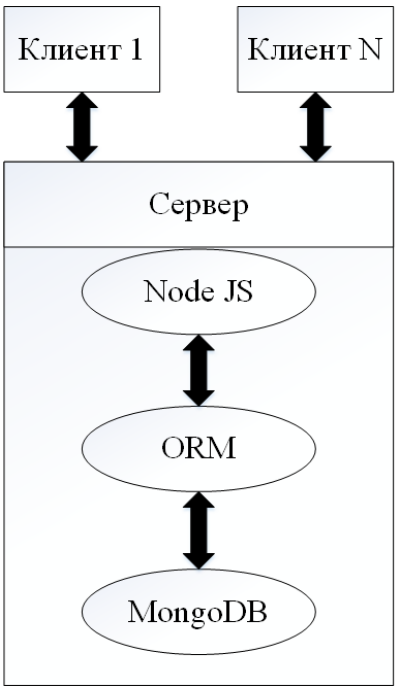

Рисунок 2.2 – Структурная схема

2.2 Средства разработки

Для разработки сервера приложения будет использоваться платформа Node.js с использованием фреймворка Express.js.

Node.JS – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения [9].

В качестве СУБД используется MongoDB.

MongoDB — документно-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Написана на языке C++ [10].

MongoDB реализует новый подход к построению баз данных, где нет таблиц, схем, запросов SQL, внешних ключей и многих других вещей, которые присущи объектно-реляционным базам данных.

В отличие от реляционных баз данных MongoDB предлагает документно-ориентированную модель данных, благодаря чему MongoDB работает быстрее, обладает лучшей масштабируемостью, ее легче использовать.

2.3 Описание функциональных возможностей системы

В системе имеется два основных вида пользователей это обычные сотрудники компании и администраторы, т.е. те, кто отвечают за работу системы. Администраторы в свою очередь делятся на роли, такие как директор (руководитель компании), HR – менеджер, составитель курсов обучения. Составителем курсов может быть обычный рядовой сотрудник компании, т.е. одновременно он может составлять курсы для обучения и быть обучаемым.

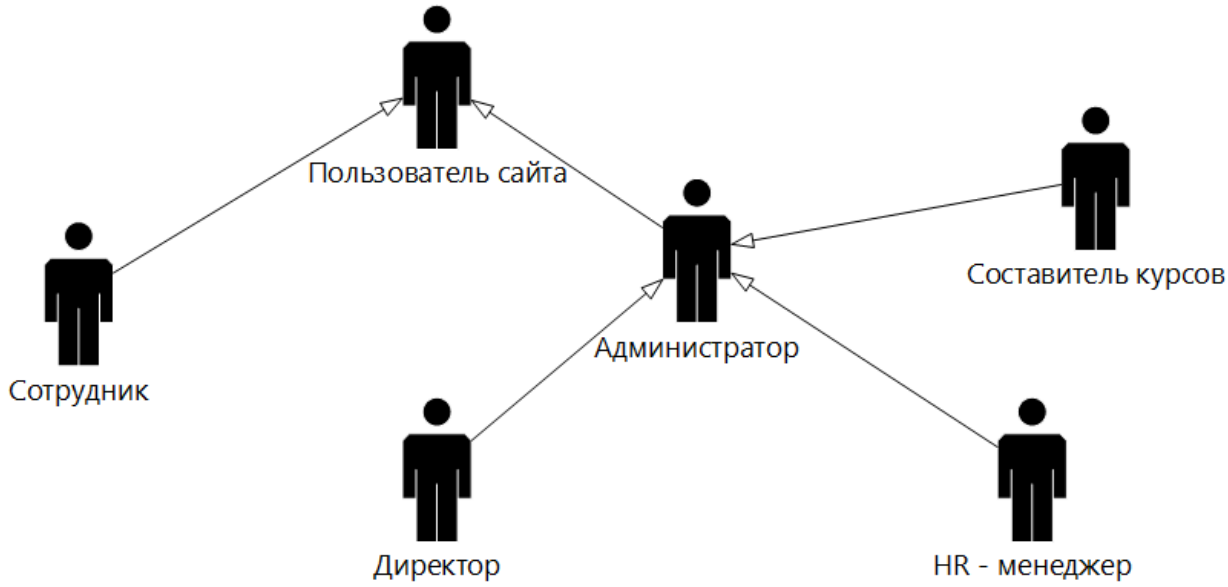

Рисунок 2.3 – Иерархия исполнитель – роль

На представленном выше рисунке 2.3 изображены несколько групп пользователей (называемые на языке UML «исполнителями» (actors)[14]), которые и будут работать с данной разрабатываемой системой. В данном случае самый общий тип это тип «Пользователь сайта», который расположен в самом верху диаграммы. Сплошная стрелка на диаграмме обозначает обобщение, т.е. имеется две основные специфические категории пользователей: посетители – сотрудники и посетители – администраторы. Характеристики, которые общие для обеих групп, будут прописаны

исполнителю «пользователь сайта», а привилегии специфические для пользователей – сотрудников и пользователей – администраторов, будут прописаны более мелкой соответствующей роли.

Функциональные возможности системы приведены в виде диаграммы прецедентов на рисунках 2.4 и 2.5.

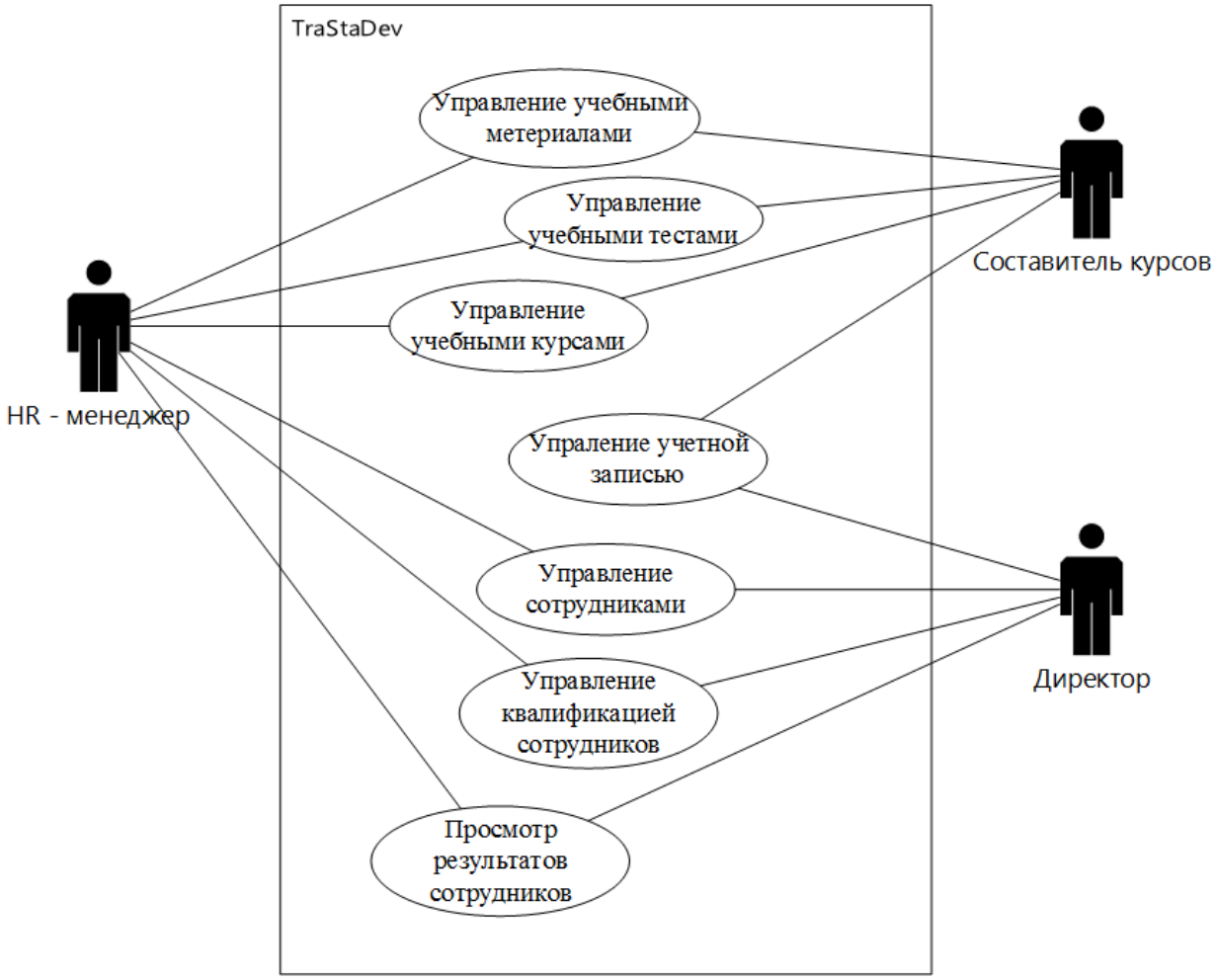

Рисунок 2.4 – Диаграмма прецедентов администратора

Прецедент «Управление учетной записью» позволяет пользователям системы изменять учетные данные после регистрации.

Прецедент «Управление учебными курсами» необходим для создания, удаления и редактирования учебных курсов. Учебный курс содержит в себе материалы, которые необходимо изучить сотруднику и тесты, которые дают возможность оценить степень изучения курса.

Прецедент «Управление учебными тестами» необходим для создания, удаления и редактирования учебных тестов. Учебный тест содержит в себе вопросы и задания, на которые должен ответить сотрудник, после изучения материала или курса.

Прецедент «Управление сотрудниками» позволяет приглашать в систему новых сотрудников либо удалять из системы существующих.

Прецедент «Управление квалификацией сотрудников» позволяет администратору на основе пройденных курсов, повышать квалификацию сотрудников.

Прецедент «Просмотр результатов сотрудников» позволяет администратору отслеживать процесс и результаты обучения каждого сотрудника.

Прецедент «Просмотр доступных курсов» позволяет сотрудникам просматривать курсы, которые доступны для изучения данным сотрудником.

Прецедент «Выбор курса обучения» дает возможность выбрать сотруднику курс, который он хотел бы изучить.

Прецедент «Тестирование» предоставляет возможность сотруднику пройти тест и узнать, какое количество баллов он набрал во время прохождения теста.

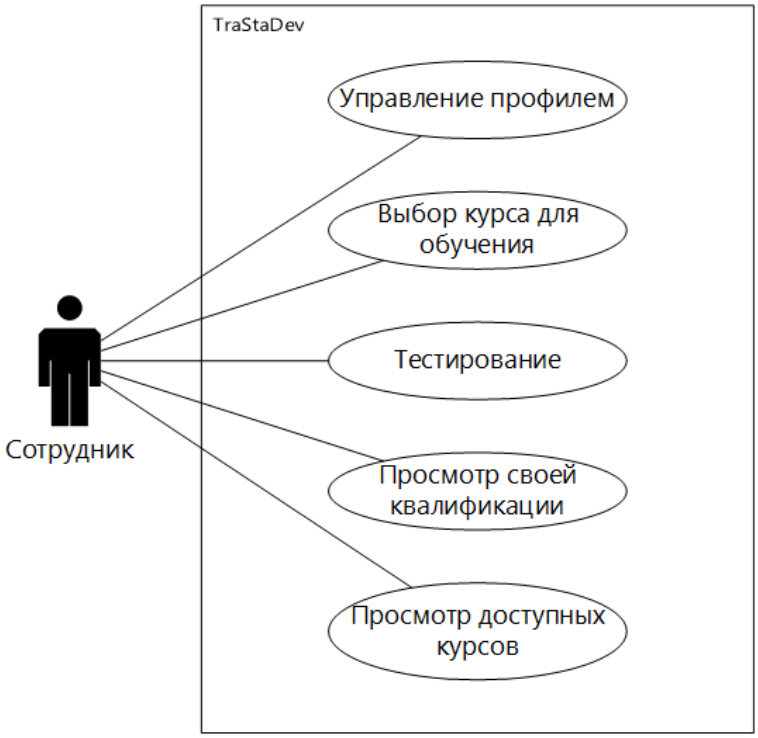

Рисунок 2.5 – Диаграмма прецедентов для сотрудников

Диаграмма активностей администратора представлена на рисунке А.1, а для сотрудников на рисунке 2.6.

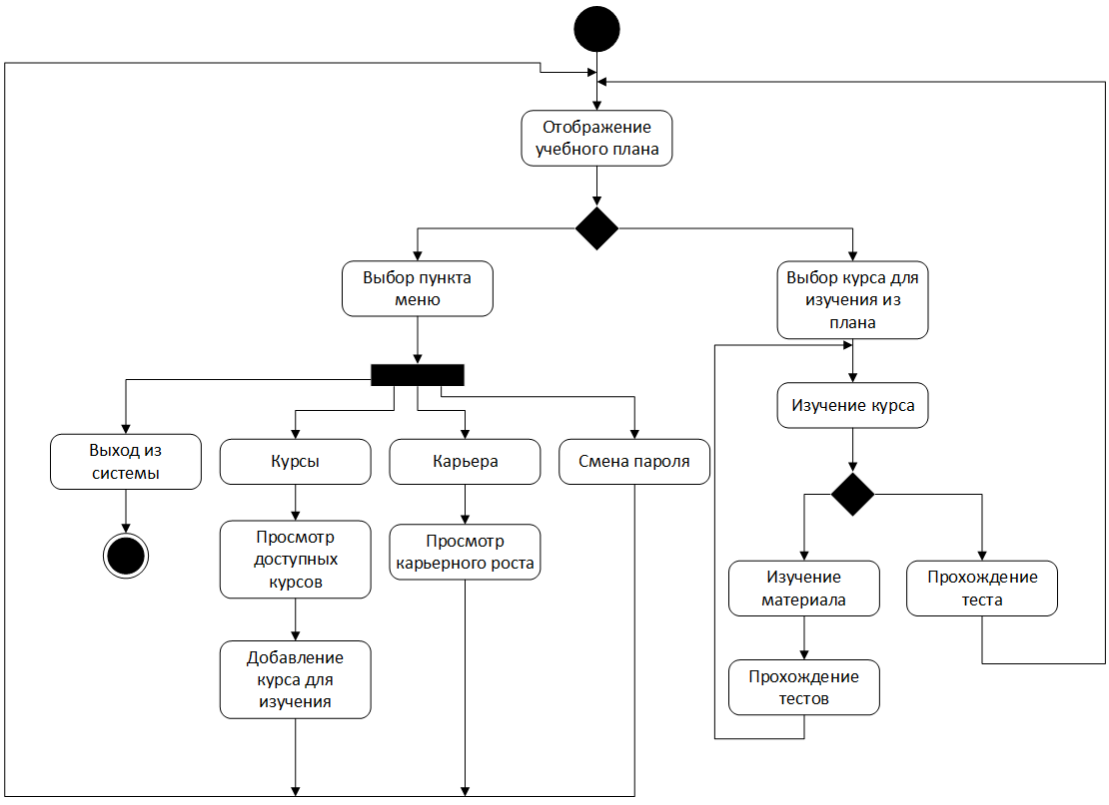

Рисунок 2.6 – Диаграмма деятельностей сотрудника

2.4 Архитектура базы данных

Для хранения данных используется структура базы данных, приведенная на рисунке 2.7.

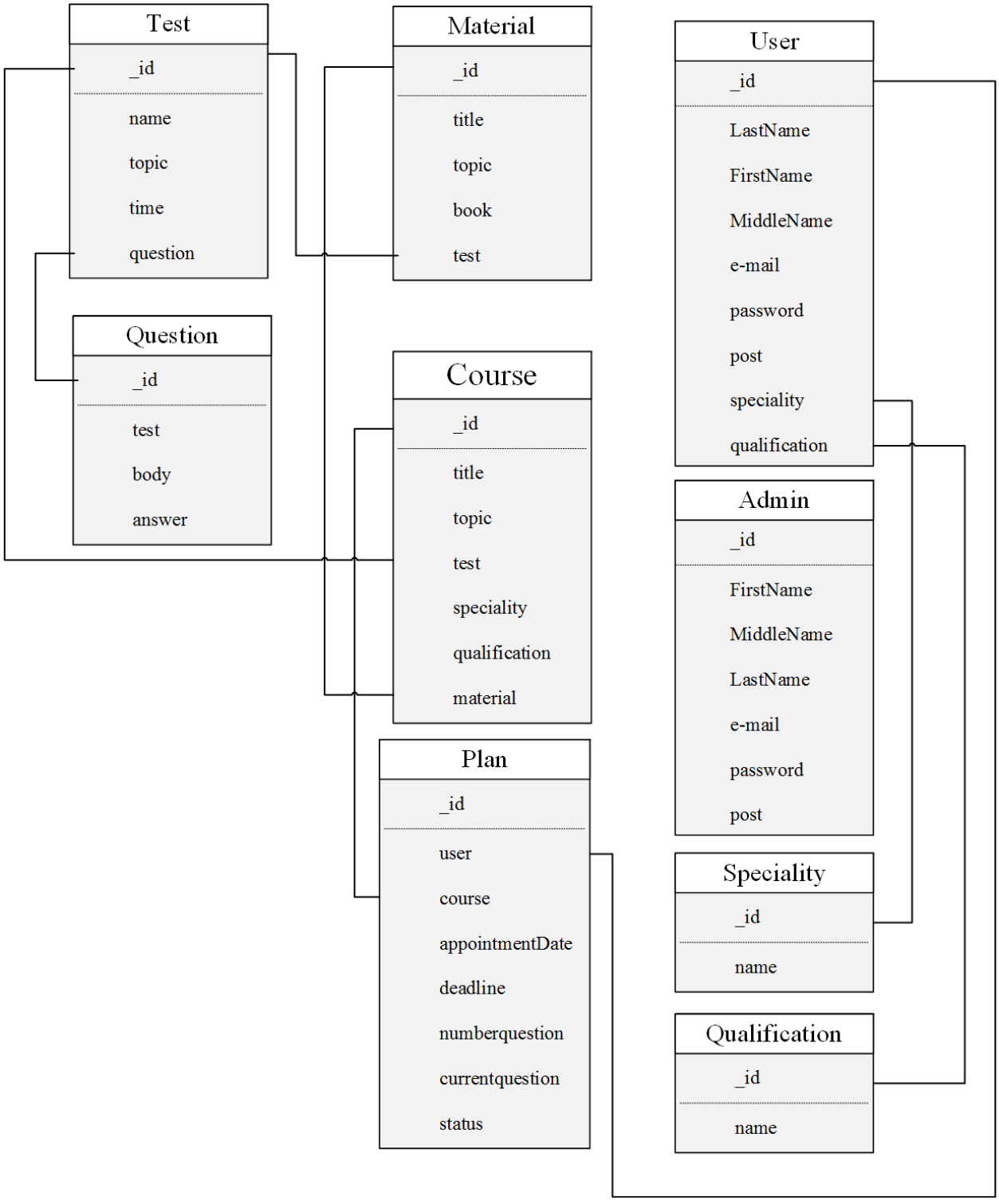

Рисунок 2.7 – Структура базы данных

Для обеспечения функционирования системы были выделены следующие сущности: Answer, Course, Users, Plan, Material, Question, Test. Каждой модели соответствует таблица в базе данных (Рисунок 2.6). Для работы с базой данных используется библиотека Mongoose, которая позволяет работать с базой данных MongoDB . После описания модели данных и последующем запуске приложения происходит, создания базы данных в соответствии с моделью данных.

Ниже приведен пример содержания сущности Users:

```
var mongoose = require('mongoose');
module.exports = mongoose.model('User',{
    username: {type: String, required: true},
    password: {type: String},
    firstName: {type: String, required: true},
    lastName: {type: String, required: true},
    middleName: {type: String, required: true},
    post: {type: String, required: true},
    speciality: {type: mongoose.Schema.Types.ObjectId, ref: 'Speciality', required: true},
    qualification: {type: mongoose.Schema.Types.ObjectId, ref: 'Qualification', required: true}
});
```

3 Реализация разрабатываемой системы

3.1 Разработка клиентской части

Для создания интерфейса приложения был выбран фреймворк Bootstrap. Bootstrap - свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. Сообществом разработчиков создано большое количество компонентов, плагинов, которые ускоряют разработку приложений и помогают создавать интерфейсы удобные для пользователей. Одной из главных особенностей Bootstrap является то, что он обеспечивает корректное отображение веб – страниц на различных устройствах, а это очень важно, так как в наше время многие пользователи используют для выхода в интернет планшеты и смартфоны.

Для разделения представления и логики приложения, в разрабатываемой системе используется JavaScript шаблонизатор – EJS. Общий принцип работы шаблонизатора представлен на рисунке 3.1.

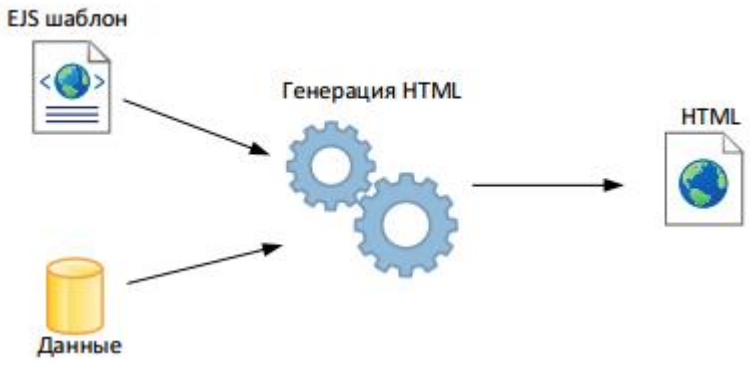

Рисунок 3.1 - Схема генерации HTML – документа

Далее приведен пример использования шаблонизации в разрабатываемом приложении:

```
router.get('/users', isAuthenticated, function(req, res){
   res.render('/users/index', { admin: req.admin });
  });
```

Методу render объекта res передаются два аргумента: первый – это путь до шаблона, который будет использоваться для генерации HTML – документа, вторым аргументом передаются данные, которые будут использованы при генерации HTML – страницы.

EJS шаблон (users/index.ejs):

```
<div class="ibox-content">
   <table class="table table-striped table-bordered table-hover dataTables-example">
      <thead>
      <tr>
         <th>ФИО</th>
      </tr>
      </thead>
      <tbody>
      <% for(var i = 0; i < users.length; i++) { %>
      <tr class="gradeX">
         <td><a href="/users/details/<%= users[i].id %>"><%= users[i].lastName %>
               <%= users[i].firstName %> <%= users[i].middleName %></a>
         </td>
      </tr>
      <% } %>
      </tbody>
   </table>
```

Во время генерации страницы выражения: <%= users[i].id %>,<%= users[i].lastName %>, <%= users[i].firstName %>, <%=users[i].middleName %> заменяются на значения, которые находятся в объекте users.

Помимо рассмотренных возможностей, EJS позволяет использовать частичные представления. Это бывает полезным для вынесения часто используемых частей разметки в отдельные файлы, а также для отображения результатов AJAX – запросов. Еще одной возможностью, которая часто используется, является использование JavaScript кода на странице шаблона. Это позволяет управлять процессом генерации в зависимости от переданных данных.

3.2 Аутентификация и авторизация пользователей

В качестве механизма аутентификации используется cookies – аутентификация. Механизм аутентификации включает в себя следующие этапы:

1. Пользователь вводит свои учетные данные на странице входа в систему и отправляет их POST – запросом на сервер.
2. Веб – сервер получает учетные данные, осуществляет аутентификацию и в случае её прохождения, осуществляет переадресацию на главную страницу web – приложения, прикрепив cookies уникальным идентификатором организованной сессии.
3. При последующей работе для каждого перехода по ресурсам приложения web – браузер автоматически отправляет соответствующий cookies с соответствующим идентификатором, полученным на шаге 2.
4. Сервер проверяет идентификатор в своей базе идентификаторов и при наличии в базе такого идентификатора, узнает пользователя и допускает его к запрашиваемому ресурсу (в случае наличия у него соответствующих прав доступа) без повторной аутентификации.

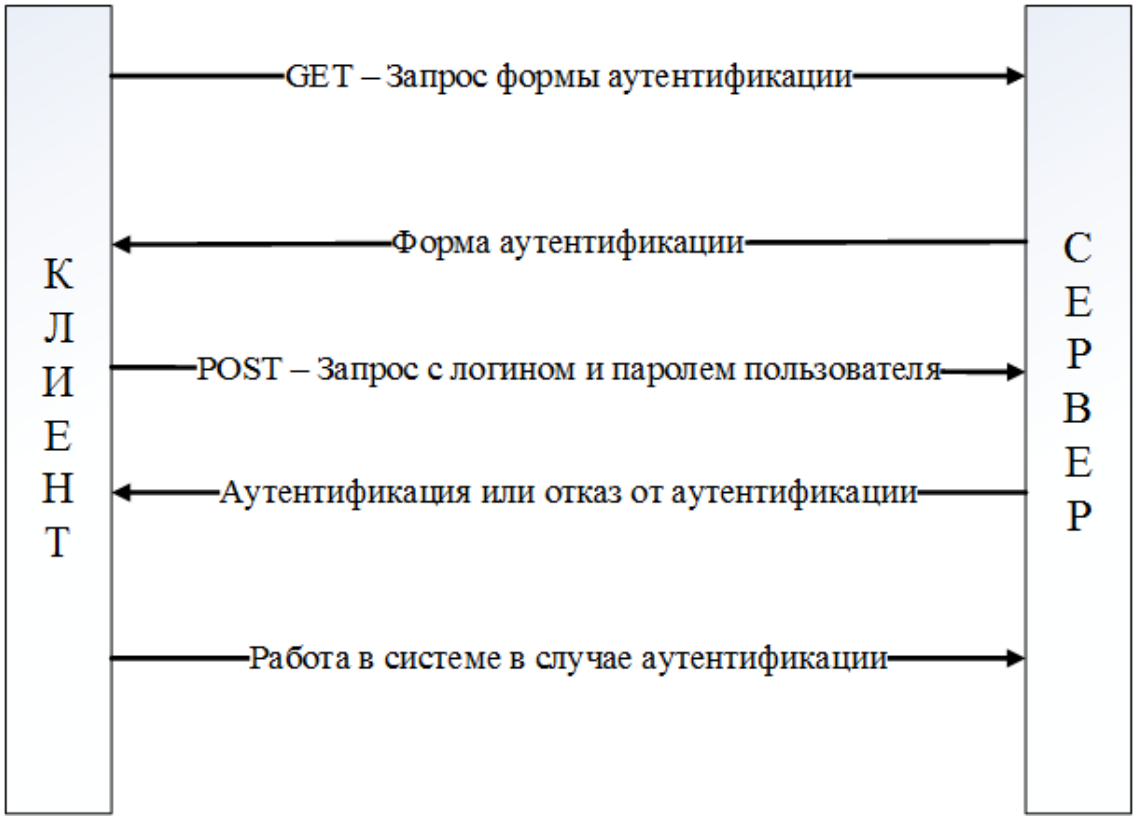

Рисунок 3.2 – Схема аутентификации пользователя

Авторизация пользователя происходит при помощи библиотек passport и passport – local.

Процесс авторизации, аутентифицированного пользователя включает в себя следующие этапы:

1. Пользователь делает запрос к методу, для доступа к которому необходимо обладать определенными правами доступа.
2. Сервер читает данные из сессии пользователя, в которых содержится информация о роли пользователя в системе.
3. В соответствии со списком контроля доступа сервер, производит проверку каждого правила. При успешном прохождении всех правил, пользователь получает доступ к запрашиваемому методу, в противном случае пользователь перенаправляется на страницу входа.

Код авторизации администратора приведен в приложении Д.

Переменная isValidPasswordAdmin используется для расшифровки паролей пользователя, так как пароли в базе данных хранятся в зашифрованном виде. Для зашифровки и расшифровки паролей используется функция bCrypt. bCrypt – адаптивная криптографическая функция формирования ключа, используемая для защищенного хранения паролей[13]. Функция bCrypt основана на шифре Blowfish.

Для хранения сессии авторизированного пользователя используется функция:

```
passport.serializeUser(function(user, done) {
    console.log('serializing user: ');console.log(user);
    done(null, user._id);
  });
```

Данная функция определяет, какие данные из объекта user, следует хранить в сессии. Результат данной функции будет прикреплен к сеансу как req.session.passport.user = {}.

3.3 Валидация данных

Для обеспечения корректного функционирования веб – приложения, все данные вводимые пользователем на страницах должны проверяться на допустимость используемых символов и на соответствие типов вводимых данных. Контроль ввода данных на стороне клиента является дополнительным контролем так как основным и наиболее важным этапом проверки данных, перед тем как данные будут сохранены в базе данных, является валидация на стороне сервера.

Для валидации на стороне клиента, используется внешняя JavaScript библиотека, так как использование стандартных атрибутов валидации, которые доступны в HTML 5 недостаточно.

При вводе данных, которые не проходят условиям валидации, пользователю выводится сообщение, информирующее его о некорректности, введённых данных и отправка данных на сервер запрещается.

Фрагмент кода, реализующий валидацию на стороне клиента, представлен в приложении Г.

3.4 Взаимодействие с базой данных

Как уже было описано в разделе 2.1 для взаимодействия системы с базой данных MongoDB используется ORM библиотека Mongoose.

Сначала прописываем серверу путь, по которому расположена созданная база данных.

```
module.exports = {
    'url' : 'mongodb://localhost/learndb'
}
```

В данном случае база данных называется learndb и расположена она по адресу localhost.

После веб – сервер должен подключиться к базе данных, это происходит путем:

```
var dbConfig = require('./db');
var mongoose = require('mongoose');
// Connect to DB
mongoose.connect(dbConfig.url);
```

Здесь объявляется две переменные, первая переменная отвечает за расположение адреса базы данных. Вторая переменная подключает библиотеку mongoose js, через которую и будет, происходит взаимодействие с базой данных. После этого идет просто подключение базы к приложению, через существующую функцию connect библиотеки mongoose, где в качестве параметра она принимает указанный путь до базы данных.

3.5 Взаимодействие клиентской части и веб – сервера

Взаимодействие клиентской части и веб – сервера происходит при помощи фреймворка ExpressJS. Данный фреймворк имеет специальную встроенную функцию router. При помощи, которой и настраивается маршрутизация. Маршрутизация определяет, как приложение отвечает на клиентский запрос к конкретному адресу (конечной точки) которым, является URI (или путь), и определенному методу запроса HTTP (GET, POST и т.д.) [15].

Прежде чем использовать функцию router, требуется подключить нужные библиотеки:

```
var express = require('express');
var router = express.Router();
```

Далее будут приведены код выполнения get и post запроса.

```
router.get('/users/details/:id', isAuthenticated, function(req, res){
   var id = req.url;
   id = id.substr(15);
   DetailsUser(id);
   res.render('users/details', {admin:req.user, 'userinfo':DetailsUser.userinfo});
  });
```

Данный кусок кода отвечает за маршрут «/users/details/:id», т.е. как будет реагировать приложение, когда пользователь отправит get запрос на данный адрес. При выполнении данного запроса, сервер с начало обработает функцию isAuthenticated, т.е. проверит, имеет ли данный авторизированный пользователь доступ к исполнению данного запроса. После того как сервер получит ответ о том что пользователю разрешен доступ, будет обрабатываться код который расположен внутри функции function(req, res). Там в переменная id, получает полный адрес запроса, т.к. «:id» не постоянная и меняется в зависимости от работы системы и действий пользователя.

После объявления переменной будет исполнена функция DetailsUser(id), которая в качестве аргумента принимает ранее объявленную переменную id. Данная функция отвечает за отображение детальной информации о сотруднике компании. Она будет подробней рассмотрена ниже.

После того как функция закончила свое выполнение, в работу вступает res.render, который в свою очередь и отвечает за отображение клиентской части приложения после выполнения запроса get. В качестве аргументов она принимает путь, где находится файл который нужно отобразить, так же render может принимать и переменные которые будут использованы при отображении клиентской части системы, в данном случае это данные о администраторе и данные о пользователе, по которому и нужно получить детальную информацию.

Далее будет представлен код файла, в котором эти переменные (admin и userinfo будут использоваться).

```
<div class="title-box">
            <h5><%= userinfo.lastName %> <%= userinfo.firstName %> <%= userinfo.middleName %></h5>
        </div>
        <div class="content">
          <dl class="dl-horizontal">
            <dt>Фамилия</dt>
            <dd><%= userinfo.lastName %></dd>
            <dt>Имя</dt>
            <dd><%= userinfo.firstName %></dd>
            <dt>Отчество</dt>
            <dd><%= userinfo.middleName %></dd>
            <dt>Email</dt>
            <dd><%= userinfo.username %></dd>
            <dt>Должность</dt>
            <dd><%= userinfo.post %></dd>
```

```
                <dt>Специализация</dt>
                <dd><%= userinfo.speciality %></dd>
                <dt>Квалификация</dt>
                <dd><%= userinfo.qualification %></dd>
              </dl>
            </div>
```

Выполнение post – запроса, post запрос позволяет отправлять какие то данные на сервер, для дальнейшей их обработки.

```
router.post('/users/create' , createUser, function(req, res){
   res.redirect('users');
 });
```

Он работает по похожему принципу, что и get, но только вместо render, здесь используется функция redirect, которая в случае успешного выполнения post запроса перенаправит пользователя по адресу «/users».

3.6 Регистрация сотрудников

Регистрировать сотрудников, как было рассмотрено в разделе 2.3 могут только либо директор, либо HR – менеджер. Далее будет рассмотрен код, который отвечает за регистрацию сотрудников в системе.

Для начала администратор отправляет get запрос на отображение формы регистрации сотрудника:

```
router.get('/users/create', isAuthenticated, function(req, res){
   DisplaySpeciality();
   res.render('users/create',{admin:req.user, specialities:
DisplaySpeciality.speciality});
 });
```

Функция DisplaySpeciality() отвечает за отображение доступных специальностей для сотрудников.

После заполнения всех полей администратором, администратор оправляет post запрос, на сохранение сотрудника в базе данных.

```
router.post('/users/create' , createUser, function(req, res){
   res.redirect('users');
  });
```

Функция createUser, как раз и отвечает за сохранение и создание сотрудника в базе данных. Код данной функции приведен в приложении В.

Переменная User отвечает за подключение за подключение модели данных user, которая взаимодействует с базой данных.

В переменную params заносятся параметры, которые будут сохраняться в базе данных, они берутся из клиентской части. Код клиентской части представлен в приложении Б.

Функция User.create отвечает за создание сотрудника в базе данных, в качестве параметров она принимает, она принимает переменную params, которая и будет сохраняться в базе и функцию. Функция отвечает за проверку на ошибки, в параметрах она принимает err и user, err – отвечает за ошибки при сохранении, а user это и есть создаваемый пользователь. Далее идет два условных оператора if, в первом условии проверяется, есть ли ошибки при создании user, второй оператор проверяет, имеется ли уже такой пользователь в базе данных. Если все два этих условия не выполняются, то происходит сохранения объекта user в базе данных, иначе будет показано сообщение об ошибке во время создания пользователя.

3.7 Отображение списка сотрудников

Список сотрудников отображается при отправке get – запроса по адресу «/users».

```
router.get('/users',isAuthenticated, function(req, res){
   OutputUser();
   res.render('users/index',{'admin': req.user, 'users':OutputUser.users});
```

```
});
```

В этом коде есть функция OutputUser(), которая отвечает за извлечение списка сотрудников из базы данных.

```
var User = require('../../models/user');
module.exports = function(){
   User
      .find()
      .populate('speciality')
      .populate('qualification')
      .exec(function(err, users){
      console.log(users);
      module.exports.users = users;
   })
};
```

User.find() отвечает за поиск объектов в базе данных, в данном случае так как не указаны параметры поиска, то он найдет все объекты которые есть в базе user.

User.populate(), отвечает за то чтобы соотнести _id специальности и квалификации сотрудника с _id из этих баз данных, в которых хранятся специальности и квалификации.

module.exports.users = users отвечает за передачу найденных объектов, которые и будут использоваться для отображения в таблице.

После выполнения функции OutputUser(), в действие вступает res.render, где в качестве параметра полученного будет использоваться user полученный в ходе выполнения функции.

Ниже представлен фрагмент клиентской части, которая и отвечает за отображения таблицы.

```
 table class="table table-striped table-bordered dataTables_wrapper">
                <thead>
                <tr>
```

```
                    <th>ФИО</th>
                    <th>e-mail</th>
                    <th>Должность</th>
                    <th>Специализация</th>
                    <th>Квалификация</th>
                </tr>
                </thead>
                <tbody>
                <% for(var i = 0; i<users.length; i++) { %>
                <tr>
                    <td><a href="/users/details/<%= users[i]._id %>"><%=
users[i].lastName %>
                            <%= users[i].firstName %> <%=
users[i].middleName%></a>
                    </td>
                    <td><%= users[i].username %></td>
                    <td><%= users[i].post %></td>
                    <td><%= users[i].speciality %></td>
                    <td><%= users[i].qualification %></td>
                </tr>
                <% } %>
                </tbody>
            </table>
```

3.8 Карьерный рост сотрудника

Карьерный рост сотрудников отображается в виде графика изображенного на рисунке 4.4.

Для отображения используется goggle charts[16]. Ниже представлен скрипт отображения графика.

```
<script type="text/javascript">
        google.charts.load("current", {packages:['corechart']});
        google.charts.setOnLoadCallback(drawChart);
        function drawChart() {

          var data = google.visualization.arrayToDataTable([
            ["Qualification", "Изученные курсы", { role: "style" } ],
            ["Стажер", <%= planIntern.length%>, "color: #228B22"],
            ["Junior Developer", <%= planJunior.length%>, "color: #006400"],
            ["Middle Developer", <%= planMiddle.length%>, "color: #00BFFF"],
            ["Senior Developer", <%= planSenior.length%>, "color: #1E90FF"],
            ["Team Lead", <%= planTeam.length%>, "color: #000080"],
            ["Architect", <%= planArchitect.length%>,"color: #FFA500"],
            ["Project Manager", 0, "color: #FF8C00"]
          ]);

          var view = new google.visualization.DataView(data);
          view.setColumns([0, 1,
            { calc: "stringify",
              sourceColumn: 1,
              type: "string",
              role: "annotation" },
            2]);

          var options = {
            title: "Карьера сотрудника: <%= userinfo.lastName %> <%= userinfo.firstName %> <%= userinfo.middleName %> ",
            width: 900,
```

```
                height: 550,
                bar: {groupWidth: "95%"},
                legend: { position: "none" },
            };
            var chart = new google.visualization.ColumnChart(document.getElementById("columnchart_values"));
            chart.draw(view, options);
        }
    </script>
```

Переменная data отвечает за параметры, которые будут изображены на графике. В ней указываются квалификации, количество изучаемых курсов по указанной квалификации. Так же указывается цвет, которым будет изображена квалификация. Переменная view отвечает за визуализацию данных представленных в переменной data. Функция chart.draw(view, options) отрисовывает данные, которые представлены ей в виде двух аргументов.

3.9 Учебный план сотрудника

Для отображения доступных курсов посылается get запрос:

```
router.get('/user/course', isAuthenticated, function(req, res){
  DisplayUserCourse(req, res);
  res.render('user/course', {'user': req.user, 'course': DisplayUserCourse.course});
 });
```

Функция DisplayUserCourse(req, res) отвечает за поиск доступных курсов для сотрудника. Код этой функции представлен ниже.

```
var Course = require('../../models/course');

module.exports = function(req, res){
  var IdSpeciality = req.user.speciality;
```

```
    var IdQualification = req.user.qualification;
    Course
        .find({'speciality': IdSpeciality, 'qualification': IdQualification})
        .exec(function(err, course){
            if (err){
                console.log('Произошла ошибка');
            }
            module.exports.course = course;
        })
}
```

После этого сотрудник может назначить один из доступных курсов для изучения. Для назначения курса выполняется post запрос.

```
router.post('/user/plan/create', CreatePlan, function(req, res){
    res.redirect('/user');
  });
```

Функция CreatePlan отвечает за добавление курса в план сотрудника. Данная функция представлена в приложении Е.

4 Функционирование системы

4.1 Отображение пользователей

Отображение пользователей по запросу «/users», после выполнения запроса пользователи отображаются в виде таблицы, в которой указаны их фамилия, имя, отчество и другие данные, которые подгружаются из базы данных. Отображение пользователей доступно только директору либо HR – менеджеру. На рисунке 4.1 показана как выглядит эта таблица.

Рисунок 4.1 – Таблица пользователей

Также в этом же окне имеется кнопка «Добавить сотрудника» при нажатие на которую администратор может добавить пользователя. Окно добавление сотрудника будет рассмотрено в следующем разделе.

4.2 Создание сотрудников

На рисунке 4.2 представлено окно создания сотрудников.

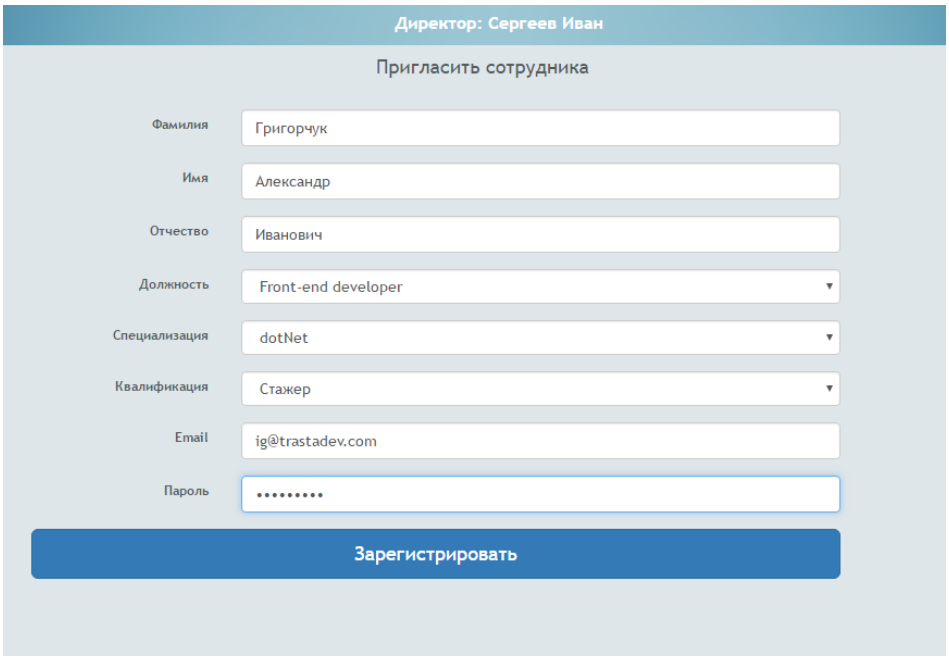

Рисунок 4.2 – Окно создания сотрудника

4.3 Создание курсов

Курсы создает либо HR – менеджер либо составитель курсов. Для этого в окне рисунка 4.3 вводятся название курса, тема, специальность и квалификация для которых создается данный курс. А также пишется список материалов по данному курсу и создается тест для оценки изученного материала данного курса.

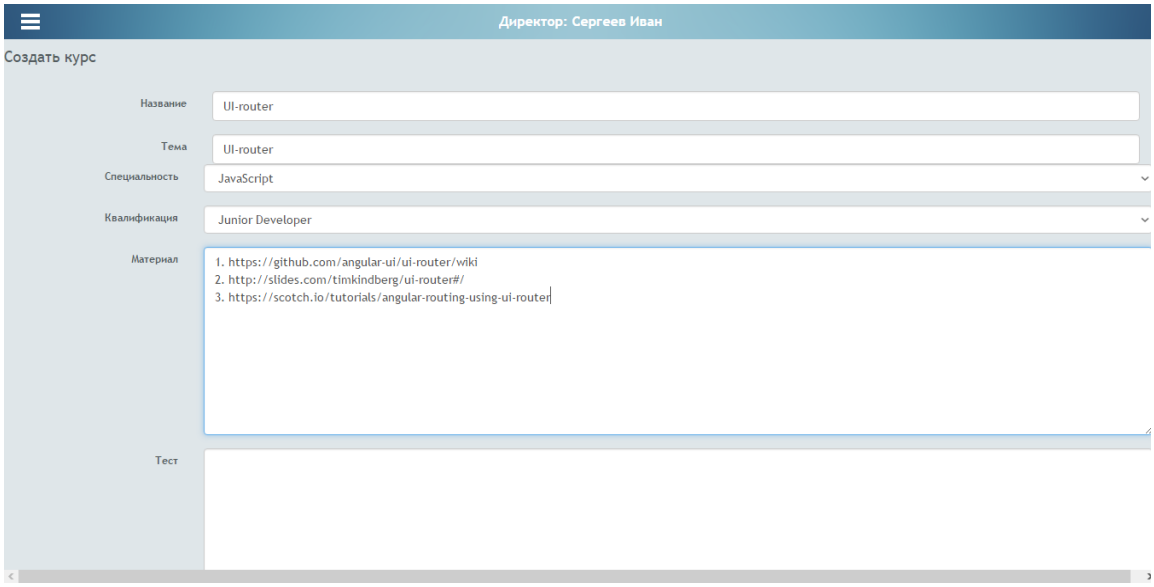

Рисунок 4.3 – Окно добавления курса

4.4 Отображение карьеры сотрудника

Карьеру сотрудника может посмотреть как сам сотрудник, так и любой пользователь из класса администраторов. Для того чтобы посмотреть карьеру нужно зайти в профиль сотрудника.

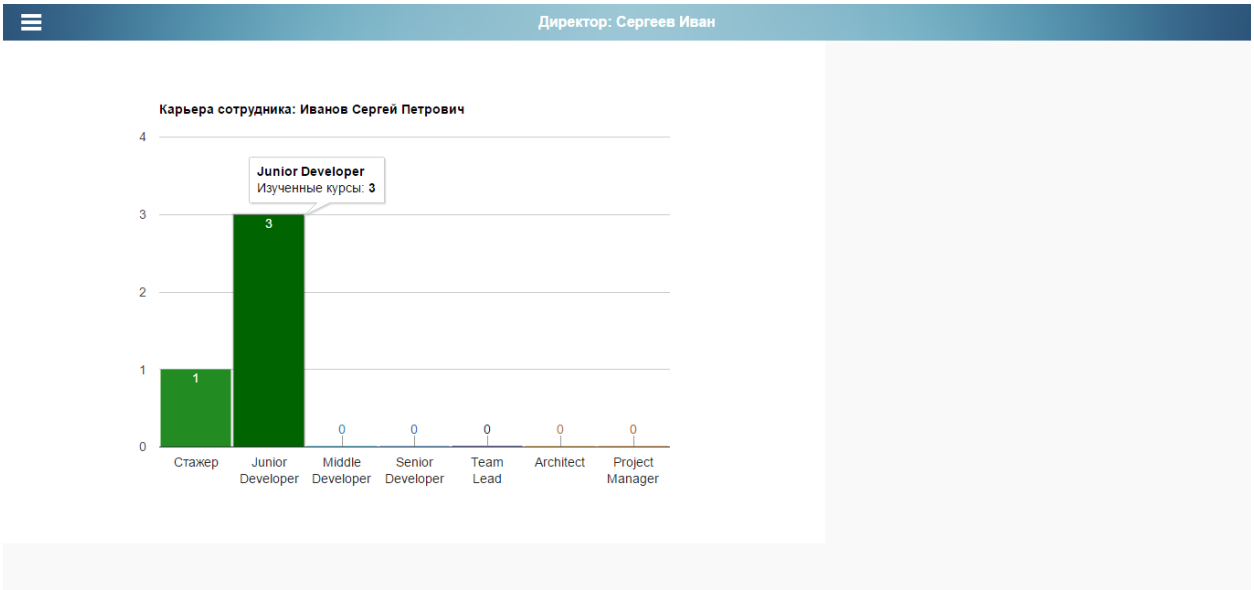

Рисунок 4.4 – График отображения карьерного роста сотрудника

4.5 Учебный план сотрудника

Курсы назначают сами себе сотрудники из доступных для них курсов, доступные курсы формируются исходя из квалификации и специальности сотрудника. На рисунке 4.5 изображен формат отображения доступных курсов.

После просмотра доступных курсов сотрудник может добавить себе этот курс для этого на экране, изображенном на рисунке 4.6, требуется нажать кнопку «Выбрать курс».

Далее появится окно добавления курса для изучения (Рисунок 4.7), где требуется указать курс, который сотрудник выбирает и дату, до которой сотрудник обязуется изучить данных курс и пройти по нему тест.

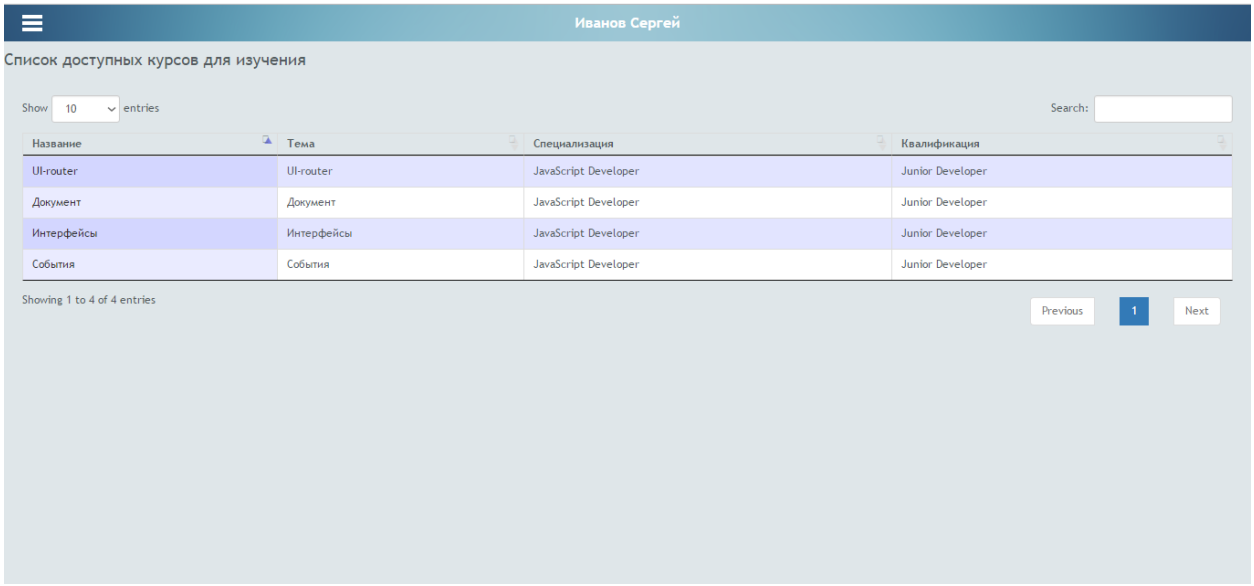

Рисунок 4.5 – Список доступных курсов для сотрудника

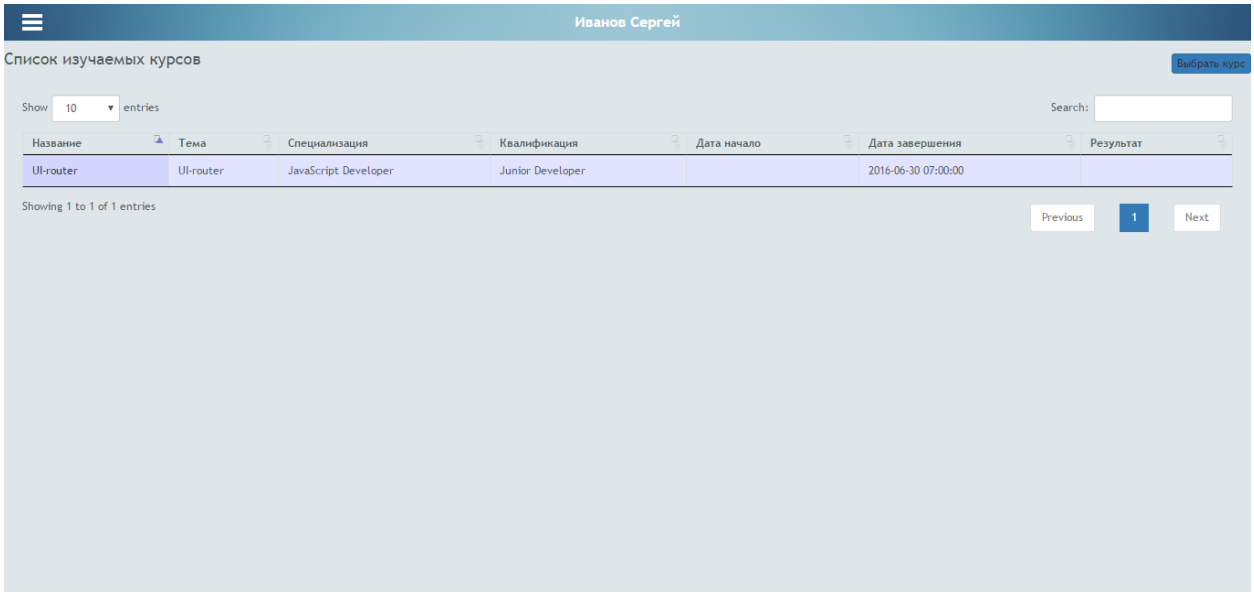

Рисунок 4.6 – Список изучаемых курсов

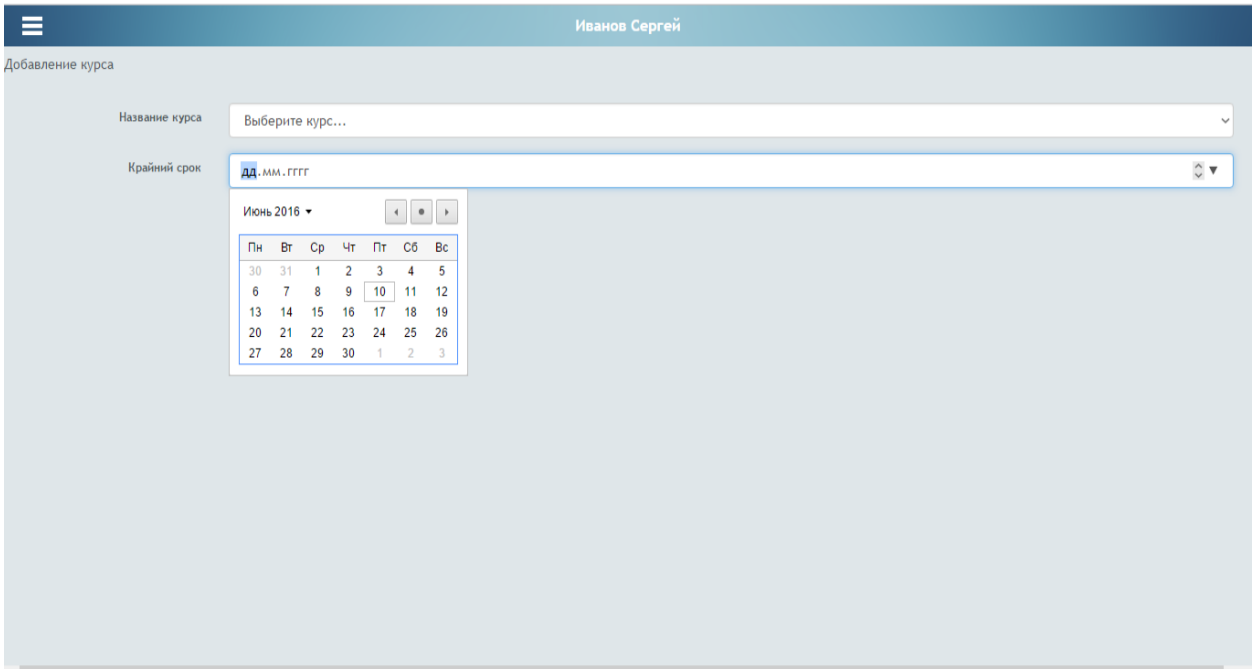

Рисунок 4.7 – Окно добавления курса для изучения

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы над проектом, были выполнены следующие задачи. Произведен анализ особенностей обучения и контроля управления персоналом. Так же в рамках анализа были изучены существующие системы, используемые для решения аналогичных задач. На основе технического задания, была выбрана клиент – серверная архитектура приложения с использованием фреймворка Express JS. Была разработана база данных, которая позволяет хранить данные пользователей, управлять учебными курсами, и все что связано с курсами, т.е. материалами и тестами, также база данных позволяет управлять учебными планами сотрудников.

При разработке системы большое внимание уделяется пользовательскому интерфейсу, при проектировании используется адаптивная верстка с применением фреймворка Bootstrap, позволяющая удобно пользоваться системой на различных устройствах.

Разработанная система позволяет регистрироваться администраторам, а также осуществлять администраторами регистрацию сотрудников. Позволяет отображать список сотрудников, доступных специальностей и квалификаций. Так же была реализована функция повышения квалификации сотрудника.

Для контроля знаний были разработаны функции, позволяющие сотруднику ответственному за обучение создавать и редактировать курсы, а обучающимся сотрудникам добавлять эти курсы себя для обучения и проходить по ним тесты.

Для просмотра карьерного роста были разработаны функции и скрипты, которые позволяют отслеживать карьерный рост сотрудников в виде диаграмм. Просматривать карьерный рост могут как обучающиеся сотрудники, так и сотрудники ответственные за контроль обучения.

Данную систему в будущем планируется внедрить в компанию Aspirity.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Система управления персоналом в организации [Электронный ресурс]: Энциклопедия экономиста - Режим доступа: http://www.grandars.ru/college/biznes/sistema-upravleniya-personalom.html
2. Обучение персонала. [Электронный ресурс] - Режим доступа: http://1-ye.ru/info/258-obuchenie-personala.html
3. Обучение персонала. [Электронный ресурс] - Режим доступа: http://www.grandars.ru/college/biznes/obuchenie-personala.html
4. Развитие персонала. [Электронный ресурс] - Режим доступа: http://www.grandars.ru/college/biznes/razvitie-personala.html
5. Управление персоналом: Учебник для вузов /Под ред. Т.Ю. Базарова, Б.Л. Еремина. — 2-е изд., перераб. и доп. — Москва: ЮНИТИ, 2002. — 560 с.
6. Планирование карьеры. [Электронный ресурс] - Режим доступа: http://www.grandars.ru/college/biznes/planirovanie-karery.html
7. Система WebTutor. [Электронный ресурс] - Режим доступа: http://www.websoft.ru/db/wb/root_id/webtutor/doc.html
8. Moorepay: Payroll Services & HR Support. [Электронный ресурс] - Режим доступа: https://www.moorepay.co.uk
9. NodeJS. [Электронный ресурс] - Режим доступа: http://nodejs.ru
10. MongoDB [Электронный ресурс] - Режим доступа: https://ru.wikipedia.org/wiki/MongoDB
11. Mongoose [Электронный ресурс] – Режим доступа: http://mongoosejs.com
12. Сухов, К. К. Node.js Путеводитель по технологии - Москва:«ДМК – Пресс», 2015. – 416с.
13. bCrypt [Электронный ресурс]: Криптографическая хеш-функция bCrypt – Режим доступа: https://ru.wikipedia.org/wiki/Bcrypt

14. Actor (UML) [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Actor_(UML)
15. Основы маршрутизации [Электронный ресурс]: Express JS – Режим доступа: http://expressjs.com/ru/starter/basic-routing.html
16. Официальный сайт библиотеки Google Charts [Электронный ресурс] : документация по библиотеке Google Charts. – Режим доступа: https://developers.google.com/chart/

ПРИЛОЖЕНИЕ А

Диаграмма деятельности HR – менеджера

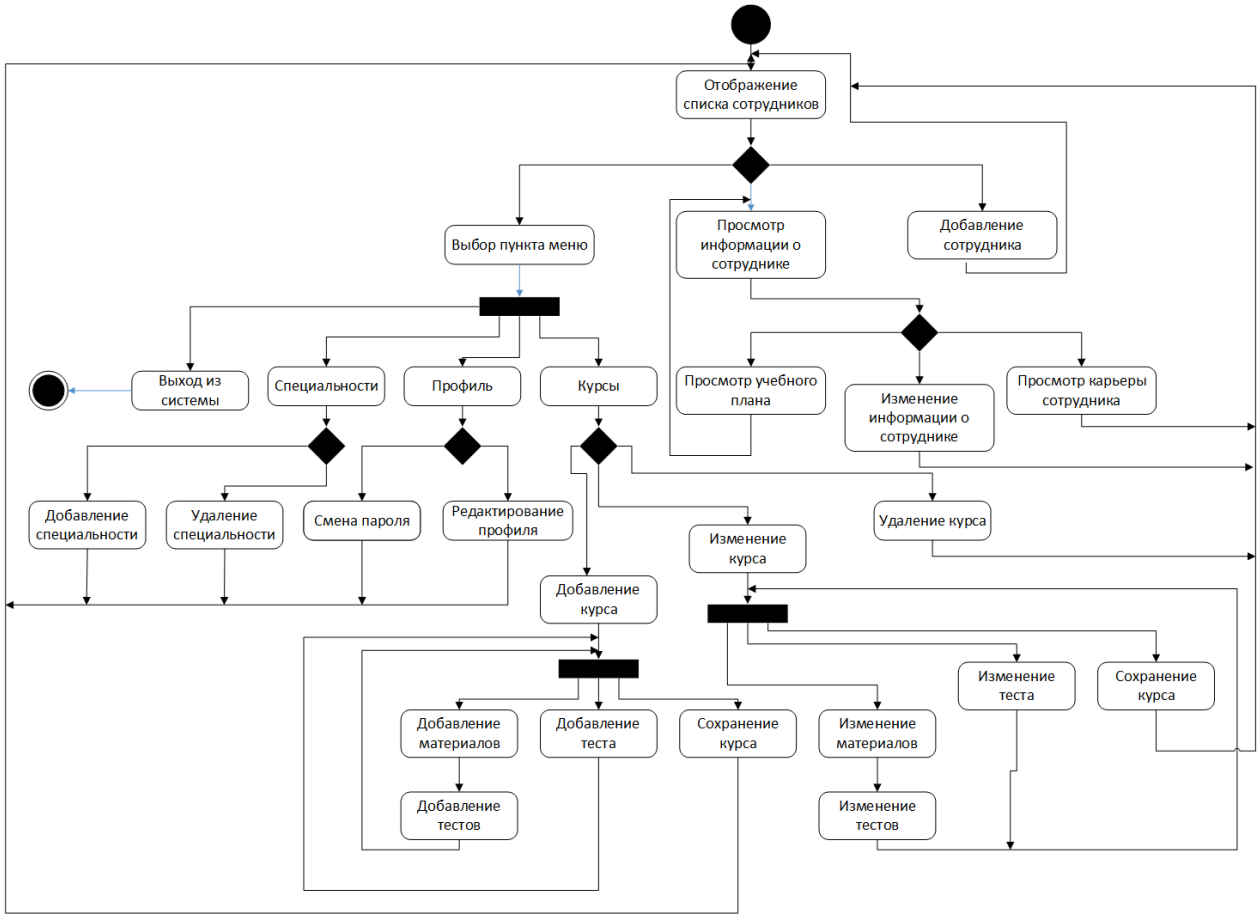

Рисунок А.1 – Диаграмма деятельности для HR – менеджера

ПРИЛОЖЕНИЕ Б

Клиентская часть регистрации сотрудников

```
 <form id="usercreateform" action="/users/create" method="post" class="form-horizontal">
                <div class="form-group">
                    <label class="col-sm-3 control-label" for="lastName">Фамилия</label>
                    <div class="col-sm-9">
                        <input id="lastName" type="text" class="form-control" name="lastName">
                    </div>
                </div>
                <div class="form-group">
                    <label class="col-sm-3 control-label" for="firstName">Имя</label>
                    <div class="col-sm-9">
                        <input id="firstName" type="text" class="form-control" name="firstName">
                    </div>
                </div>
                <div class="form-group">
                    <label class="col-sm-3 control-label" for="fiddleName">Отчество</label>
                    <div class="col-sm-9">
                        <input id="middleName" type="text" class="form-control" name="middleName">
```

```
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3 control-label"
for="post">Должность</label>
                <div class="col-sm-9">
                    <select size="1" id="post" class="form-control"
name="post">
                        <option disabled>Укажите должность
сотрудника</option>
                        <option value="Front-end developer">Front-end
developer</option>
                        <option value="Back-end developer">Back-end
developer</option>
                    </select>
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3 control-label"
for="speciality">Специализация</label>
                <div class="col-sm-9">
                    <select size="1" id="speciality" class="form-control"
name="speciality">
                        <option disabled>Укажите специализацию
сотрудника</option>
                        <% for(var i = 0; i<specialities.length; i++) { %>
                        <option value="<%= specialities[i]._id %>" ><%=
specialities[i].name %></option>
                        <%}%>
```

```
                </select>
            </div>
        </div>
        <div class="form-group">
            <label class="col-sm-3 control-label" for="qualification">Квалификация</label>
            <div class="col-sm-9">
                <select size="1" id="qualification" class="form-control" name="qualification">
                    <option disabled>Укажите квалификацию сотрудника</option>
                    <% for(var i = 0; i<qualifications.length; i++) { %>
                    <option value="<%= qualifications[i]._id %>" ><%= qualifications[i].name %></option>
                    <%}%>
                </select>
            </div>
        </div>
        <div class="form-group">
            <label class="col-sm-3 control-label" for="username">Email</label>
            <div class="col-sm-9">
                <input id="username" type="email" class="form-control" name="username">
            </div>
        </div>
        <div class="form-group">
```

```
                <label class="col-sm-3 control-label"
for="password">Пароль</label>

                <div class="col-sm-9">

                    <input id="password" type="password" class="form-control"
name="password">

                </div>

            </div>

            <div class="hr-line-dashed"></div>

            <input class="btn btn-lg btn-primary btn-block" type="submit"
value="Зарегистрировать">

        </form>
```

ПРИЛОЖЕНИЕ В

Регистрация пользователя

```
var User = require('../../models/user');
var bCrypt = require('bcrypt-nodejs');

module.exports = function(req, res){
   var params = {
      username: req.param('username'),
      lastName: req.param('lastName'),
      firstName: req.param('firstName'),
      middleName:req.param('middleName'),
      post: req.param('post'),
      speciality: req.param('speciality'),
      qualification:  req.param('qualification'),
      password: createHash(req.param('password'))
   };
   User.create(params, function(err, user){
      if (err){
         console.log('Error in SignUp: '+err);
         return res.redirect('users/create');
      }
      // already exists
      if (user) {
```

```
            console.log('User already exists with username: '+username);
            return req.flash('message','User Already Exists');
        }
    });
};
```

ПРИЛОЖЕНИЕ Г

Валидация данных

```
jQuery.extend(jQuery.validator.messages, {
   required: "Это поле должно быть заполнено.",
   email: "Введите правильный адрес электронной почты.",
   minlength: jQuery.validator.format("Минимальное количестово символов {0}.")
});
jQuery.validator.addMethod("lettersonly", function(value, element) {
   return this.optional(element) || /^[a-яA-ЯёЁa-zA-Z-]+$/i.test(value);
}, "Это поле дожно содержать только буквы.");
$(document).ready(function() {
   $("#registerForm").validate({
      rules: {
         firstName: {
            lettersonly: true,
            required: true
         },
         lastName: {
            lettersonly: true,
            required: true
         },
         middleName: {
            lettersonly: true,
```

```
            required: true
        },
        Post: {
            required: true
        },
        username: {
            required: true,
            email: true
        },
        password: {
            required: true,
            minlength: 8
        }
    },
    highlight: function(element) {
        $(element).closest('.form-group').addClass('has-error');
    },
    unhighlight: function(element) {
        $(element).closest('.form-group').removeClass('has-error');
    },
    errorElement: 'span',
    errorClass: 'help-block',
    errorPlacement: function(error, element) {
        if(element.parent('.input-group').length) {
            error.insertAfter(element.parent());
```

```
        } else {
            error.insertAfter(element);
        }
    }
});
```

ПРИЛОЖЕНИЕ Д

Авторизация пользователя

```
module.exports = function(passport){
    passport.use('loginA', new LocalStrategy({
            passReqToCallback : true
        },
        function(req, username, password, done) {
            // check in mongo if a user with username exists or not
            Admin.findOne({'username': username},
                function (err, admin) {
                    // In case of any error, return using the done method
                    if (err)
                        return done(err);
                    // Username does not exist, log the error and redirect back
                    if (!admin) {
                        console.log('User Not Found with username '+username);
                        return done(null, false, req.flash('message', 'User Not
found.'));
                    }
                    // User exists but wrong password, log the error
                    if (!isValidPasswordAdmin(admin, password)) {
                        console.log('Invalid Password');
                        return done(null, false, req.flash('message', 'Invalid
Password')); // redirect back to login page
```

```
                    }
                    // User and password both match, return user from done method
                    // which will be treated like success
                    return done(null, admin);
                }
            );
        })
    );
var isValidPasswordAdmin = function(admin, password){
    return bCrypt.compareSync(password, admin.password);
};
```

ПРИЛОЖЕНИЕ Е

Функция добавления курса в план сотрудника

```
var Plan = require('../../models/plan');
var Course = require('../../models/course');

module.exports = function(req, res){
    var IdUser = req.user._id;
    var titleCourse = req.param('course');
    var deadline = req.param('deadline');

    Course
        .findOne({'title': titleCourse})
        .exec(function(err, course){
            if (err) {
                return res.notFound();
            }
            if(!course) {
                return res.notFound();
            }

            var params  = {
                'deadline': deadline,
                'course': course.id,
```

```
            'user': IdUser
        };

        Plan
            .findOne({'course': course.id, 'user': IdUser})
            .exec(function (err, course) {
                if (course) {
                    req.flash('error', 'Ошибка добавления. Материал уже находится на изучении.');
                    return res.redirect('/user/plan/create/');
                } else {
                    Plan
                        .create(params,function (err, plan) {
                            if (err){

                            }
                            return res.redirect('/user');
                        });
                }
            });
    })
}
```