

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

«Разработка веб-приложения профилирования компаний - резидентов
Технопарка Университета ИТМО»

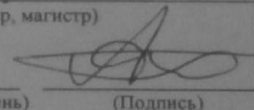
Автор Гордеев Борис Романович
(Фамилия, Имя, Отчество)


(Подпись)

Направление подготовки 09.03.02 Информационные системы и
технологии

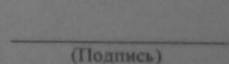
Квалификация Бакалавр
(бакалавр, инженер, магистр)

Руководитель к.т.н. Кашевник А.М.
(Фамилия, И. О., ученое звание, степень)


(Подпись)

К защите допустить

Зав. кафедрой ИС Парфенов В.Г., проф., д.т.н.
(Фамилия, И. О., ученое звание, степень)


(Подпись)

“ ” _____ 2016 г.

Санкт-Петербург, 2016 г.

Студент Гордеев Борис Романович Группа М3407 Кафедра ИС Факультет ИТиП
(ФИО)

Направленность (профиль), специализация 09.03.02 "Информационные системы и технологии"

Консультант(ы):

а) _____ (Фамилия, И., О., ученое звание, степень) _____ (Подпись)

б) _____ (Фамилия, И., О., ученое звание, степень) _____ (Подпись)

Квалификационная работа выполнена с оценкой _____

Дата защиты "____" июня 2016 г.

Секретарь ГЭК Маятин А.В. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”

УТВЕРЖДАЮ

Зав. кафедрой ИС

проф. Парфенов В.Г.

(ФИО)

(подпись)

«29» «декабря» 2015 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студента Гордеева Бориса Романовича Группа М3407 Кафедра ИС Факультет ИТиП

Руководитель к.т.н. Кашевник Алексей Михайлович

(ФИО, ученое звание, степень, место работы, должность)

1 Наименование темы: Разработка веб-приложения профилирования компаний - резидентов Технопарка Университета ИТМО

Направление подготовки (специальность) 09.03.02

Направленность (профиль) 09.03.02 "Информационные системы и технологии"

Квалификация Бакалавр

(бакалавр, магистр, специалист)

2 Срок сдачи студентом законченной работы

«17» «мая» 2016г.

3 Техническое задание и исходные данные к работе

Задачей разработки является система в виде программного продукта, взаимодействующая с базой данных, а также имеющая пользовательский интерфейс в виде Web – приложения. Web – приложение позволит обеспечить выполнение предусмотренных системой операций пользователем. Исполнителем совместно с руководителем составлено подробное техническое задание, регламентирующее требования к данному приложению, а также раскрывающее задачи и цели разработки. Исполнителю также был предоставлен набор научных работ в области и возможность регулярно контактировать с руководителем.

4 Содержание выпускной работы (перечень подлежащих разработке вопросов)

Объектом разработки выпускной квалификационной работы является веб-приложение профилирования компаний-резидентов технопарка ИТМО. В содержании работы будет описан процесс разработки в виде разбора каждого из этапов разработки – анализа, проектирования, реализации и тестирования.

5 Перечень графического материала (с указанием обязательного материала)

Обязательным материалом, предоставленным в работе, являются диаграмма программной архитектуры, разработанная в нотации UML Class, и диаграмма системной архитектуры, разработанная в нотации UML Deployment. Остальные изображения, включенные в работу: Диаграмма основных сценариев использования (UML Use Case), 5 диаграмм UML Activity, 1 диаграмма UML Sequence, 2 диаграмма UML Class, 3 изображения Reverse Engineer базы данных MySQL, 1 изображение интерфейса системы.

6 Исходные материалы и пособия

Для выполнения задания исполнителю был предоставлен набор научных работ. Полный список научных работ, использованных исполнителем, приведён в конце работы.

КАЛЕНДАРНЫЙ ПЛАН

№№ п/п	Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Отметка о выполнении, подпись руков.
1	Анализ предметной области	1 февраля 2016	
2	Разработка системной архитектуры системы управления компетенциями	1 февраля 2016	
3	Разработка программной архитектуры системы управления компетенциями	1 февраля 2016	
4	Разработка модели данных системы управления компетенциями	1 марта 2016	
5	Реализация системы управления компетенциями	15 апреля 2016	
6	Написание пояснительной записки и отчётности	15 апреля 2016	

8. Дата выдачи задания «22» «декабря» 2015г.

Руководитель

(подпись)

Задание принял к исполнению

(подпись)

«22» «декабря» 2015г.

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”

АННОТАЦИЯ

ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студент Гордеев Борис Романович
(ФИО)

Наименование темы ВКР: Разработка веб-приложения профилирования компаний - резидентов Технопарка Университета ИТМО

Наименование организации, где выполнена ВКР Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования Разработка веб-приложения профилирования компаний-резидентов Технопарка Университета ИТМО

2 Задачи, решаемые в ВКР Профилирование компаний-резидентов Технопарка Университета ИТМО

3 Число источников, использованных при составлении обзора 16

4 Полное число источников, использованных в работе 16

5 В том числе источников по годам

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	6	5	5

6 Использование информационных ресурсов Internet 1
(Да, нет, число ссылок в списке литературы)

7 Использование современных пакетов компьютерных программ и технологий
(Указать, какие именно, и в каком разделе работы)

Visual Paradigm CE 12.2 (Анализ, Проектирование, Реализация), Spring Tool Suite 3.72
(вместе с Java 8) (Реализация), MySQL Workbench 6.3.6 (вместе с MySQL Server 5.7)
(Анализ, Проектирование, Реализация), Microsoft Office 2010 (Документирование).

8 Краткая характеристика полученных результатов В ходе работы была реализована
требуемая система, процесс её реализации описан в виде ВКР. Система протестирована, что
может говорить о её работоспособности и пригодности к введению в эксплуатацию.

9 Полученные гранты, при выполнении работы При выполнении работы грантов не

(Название гранта)

Получено

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы Да

(Да, нет)

a) 1 Gordeev B., Baranits O., KASHCHENIK A. Web-Based Competency Management System for ITMO University

(Библиографическое описание публикаций)

Technopark, 12-th Conf. of Open Innovation Association FRUCT, St. Petersburg, 2016, P. 463-466

X

б) 1 Выступление с презентацией на демо секции 18-ой конференции FRUCT

(Библиографическое описание выступлений на конференциях)

2

3

Выпускник

(ФИО)

(подпись)

Руководитель

(ФИО)

(подпись)

“ ” 2016 г.

Реферат

Отчет 45 страниц, 1 часть, 14 изображений, 1 приложение, 16 источников
ИНФОРМАЦИОННЫЕ СИСТЕМЫ, СИСТЕМЫ ПРОФИЛИРОВАНИЯ, СИСТЕМЫ
УПРАВЛЕНИЯ КОМПЕТЕНЦИЯМИ, КОМПЕТЕНЦИИ, ВЕБ-ПРИЛОЖЕНИЯ

Объектом исследования являются методы и системы профилирования и управления компетенциями, а также их разработка.

Цель работы – разработка веб-приложения для хранения и обработки профилей компаний-резидентов Технопарка Университета ИТМО.

Методы исследования – разработка и анализ ПО.

Результаты выполнения – в ходе выполнения работы была реализована система профилирования на основе компетентностного подхода для хранения профилей компаний. В работе описаны все этапы разработки данной системы – анализ, проектирование, реализация на различных уровнях и тестирование. Также указаны некоторые результаты выполнения работы – введение системы в эксплуатацию и написание научных работ по полученному в ходе выполнения опыту.

Оглавление

Реферат.....	4
Оглавление	5
Введение	6
1 Анализ научных работ, аналогов и формирование требований к системе.....	8
1.1 Терминология и научные основы в области управления компетенциями	8
1.2 Современные работы в области управления компетенциями и профилирования	10
1.3 Функциональные требования	12
1.4 Результаты анализа	13
2 Проектирование системы	14
2.1 Выбор технологий и инструментов разработки.....	14
2.2 Системная архитектуры	17
2.3 Программная архитектура.....	18
2.4 Основные сценарии использования	20
2.5 Результаты проектирования системы	22
3 Реализация системы	23
3.1 Порядок проверок и приёма.....	23
3.2 Реализация модели данных и взаимодействие с базой данных	23
3.3 Реализация полнотекстового поиска.....	26
3.4 Безопасность, права доступа и аккаунты пользователей.....	27
3.5 Реализация операций поиска и сравнения.....	29
3.6 Реализация компонента заявок.....	31
3.7 Реализация веб-приложения	34
3.8 Результаты реализации системы	36
4 Тестирование системы	37
4.1 Формирование требований к тестированию и определение используемых для тестирования технологий.....	37
4.2 Тестирование модели данных.....	38
4.3 Тестирование алгоритмов поиска и сравнения.....	40
4.4 Другие интеграционные тесты	40
4.5 Ручное тестирование на примере профиля компании “Rehabot”	41
4.6 Результаты выполнения тестирования	42
Заключение	43
Список научных источников и упоминаемых материалов	44
Приложения.....	46
Приложение А – диаграмма программной архитектуры	46

Введение

Технопарк Университета ИТМО представляет собой совокупность небольших компаний, называемых компаниями – резидентами. Зачастую, для нахождения компании, которая могла бы справиться с определённой задачей, клиент обращается в Технопарк. Таким образом, актуальной является задача профилировать компании – резиденты Технопарка ИТМО так, чтобы пользователь такой системы мог определить, подойдёт ли та или иная компания конкретно под его задачу, и если подойдёт, то насколько хорошо. Для решения такой задачи было решено использовать подход управления компетенциями. Подобный подход позволяет решать поставленную задачу с использованием несложных математических алгоритмов, описанных далее. Целью данной работы является анализ предметной области данной задачи, проектирование и реализация решающей эту задачу системы, реализующей компетентностный подход.

Управление человеческими ресурсами, основанное на компетенциях, (Competency-based management, CBM) стало очень важным элементом в эффективной работе любого предприятия или бизнес – организации в связи с необходимостью последних быть достаточно гибкими чтобы адаптироваться к быстрым переменам на рынке и переориентациям бизнес планов. В таких ситуациях, CBM будет центральным инструментом управления человеческими ресурсами, который позволит предприятию управлять и развивать умения его сотрудников, нанимать самых подходящих кандидатов и выполнять эффективное планирование как результатов своей деятельности, так и развития сотрудников. [1]

Использование методов подхода компетенций для управления человеческими ресурсами уже широко распространено в США и активно развивается в международной практике. [2]

Использование данного подхода не только позволяет лучше визуализировать возможности сотрудников, но и даёт толчок в сторону новых стратегий их обучения, управляемых не HR, но менеджерами и сотрудниками. В данном контексте роль HR будет развиваться в сторону «помощника» и консультанта в корректном использовании подхода компетенций для ускорения индивидуального, командного и организационного роста [2].

Задачей разработки является система в виде программного продукта, взаимодействующая с базой данных, а также имеющая пользовательский интерфейс в виде Web – приложения. Web – приложение позволит обеспечить выполнение предусмотренных системой операций пользователем.

Целями создания системы являются:

1. Дать возможность клиенту Технопарка Университета ИТМО провести анализ предложенных компаний и выбрать ту, к которой он смог бы обратиться. Клиенту будет дана возможность сделать это посредством web – приложения.

2. Дать возможность компании – резиденту Технопарка Университета ИТМО предоставить максимально подробно информацию о своих возможностях потенциальному клиенту.

1 Анализ научных работ, аналогов и формирование требований к системе

1.1 Терминология и научные основы в области управления компетенциями

Компетенция - конкретное, идентифицируемое, определяемое и измеримое знание, умение, способность и/или другая относящаяся к контексту характеристика (как то отношение, поведение, физическая способность), которой может обладать человеческий ресурс, и которая необходима или существенна для качественного выполнения действия в контексте конкретного бизнес-процесса [3]. Способность или умение, находящееся на некотором уровне и требуемое для выполнения задачи [4].

Профиль компетенций - Множество компетенций с соответствующими им уровнями, требуемыми для участия в назначенной роли (ролях), которая заключается в выполнении всех подразумеваемых этой ролью задач [4].

С профилями компетенций можно выполнять следующие действия [4]:

1. Создание профиля

Описание компетенций с соответствующими им уровнями, которые требуются для участия в назначенной роли (ролях), которая заключается в выполнении всех подразумеваемых этой ролью задач.

2. Удаление профиля

3. Добавление компетенции к профилю

Добавление компетенций с соответствующими им уровнями к профилю, необходимых для участия в новой роли, которая заключается в выполнении всех задач подразумеваемых этой ролью.

4. Удаление компетенций из профиля

Добавление компетенций с соответствующими им уровнями из профиля, необходимых для участия в уже нерелевантной роли.

5. Извлечение подпрофиля компетенций

Ограничение профиля только тем набором компетенций с соответствующими уровнями, который необходим для выполнения данного набора задач.

6. Сравнение профиля с ресурсом

Сравнить задачи, для выполнения которых ресурс нужен, с задачами, для выполнения которых компетенции с соответствующими уровнями включены в профиль. Данная функция подразумевает сравнение двух ресурсов по множеству решаемых ими задач, а результатом данного сравнения должно было бы стать число от 0 до 1. В рамках данного проекта такое сравнение принято считать нерелевантным к задаче, сравнение в системе будет происходить по множеству компетенций.

7. Поиск профилей

Нахождение всех профилей, которые включают компетенции с соответствующими уровнями, необходимые для выполнения задач, указанных в запросе, иначе говоря нахождение всех профилей, поддерживающих выполнение обозначенных задач.

8. Ранжирование профилей

Упорядочивание набора профилей компетенций на основании критерия. Под критерием подразумевается результат сравнения профилей с ресурсом.

Стоит отметить, как компетенции используются на уровне взаимодействия между компаниями.

Компетенции активно изучались на индивидуальном и командном уровнях, и не так давно стали изучаться на уровне фирм, но редко изучались на уровне взаимодействия организаций [5]. Самой главной проблемой при определении компетенций организаций стала проблема баланса, а именно попытка включения в определение одновременно идей о знании (know-how) и действии (приложении умения) [6]. При рассмотрении большого числа литературы можно выделить четыре подхода:

Первый подход говорит, что *organizational competence* включает в себя понимание конкретного феномена и связанных дисциплин, как то фармацевтика или электроника.

Второй подход говорит, что *organizational competence* включается в себя технологию, как то вычисления или печать, и связанные продукты.

Третий подход предполагает, что *organizational competence* включает в себя функциональное умение, как то маркетинг, производство, распределение или планирование.

Четвёртый подход гласит, что *organizational competence* включает в себя какую-то интеграцию, как правило технологии и способности. Примером является способность Honda Corporation интегрировать технологию внутреннего сгорания вместе с функциональной способностью инжиниринга и производства для изготовления высококачественных двигателей малого размера.

1.2 Современные работы в области управления компетенциями и профилирования

В ходе анализа требований к приложению был произведён анализ приложений-аналогов. Из научных статей можно выделить следующие аналоги:

Fotis Fraganidis и Greogoris Mentzas [7] реализовали систему менеджмента компетенций для корпоративного онлайн-обучения. Система опиралась на стандарт хранения компетенций HR-XML. В основе системы лежала онтология компетенций. Были сделаны выводы в виде общих рекомендаций по разработке подобных систем. Данные рекомендации были приняты к сведению при реализации данной работы.

Jürgen Dorn и Markus Pichlmair [8] разработали систему менеджмента компетенций для учёта успеваемости студентов. Был также использован стандарт HR-XML. Понятие компетенций было рассмотрено не только в отношении практических умений, но и личных характеристик. Был изучен формат основных сущностей.

Pooja Tripathi, Jayanthi Ranjan и Tarun Pandeya [1] разработали фреймворк для *competency management* в институтской среде, опирающийся на модель PAKS (Personality, Ability, Knowledge, Skills).

Gilbert Paquette [9] представил онтологию для разработки основанных на компетенциях приложений для обучения и управления знаниями. На основании этой онтологии был предоставлен фреймворк для систем электронного обучения.

Одним из аналогов разрабатываемой системы можно считать систему SOBOLEO [10]. SOBOLEO представляет собой веб-приложение, где уровень хранения данных работает за счёт хранения RDF триплетов в файлах, а логика реализована за счёт языка программирования Java.

V. Iordan и A. Cicortas [11] на основании онтологии разработали мультиагентную систему, реализующую сравнение множеств компетенций друг с другом. Компетенции рассматривались в контексте взаимоотношения студента со своим университетом и потенциальным работодателем.

C. Niculescu и S. Trausan-Matu [12] разработали онтологию для системы управления и сформулировали некоторые свойства системы, реализующей данную систему. В той частности, они говорили и необходимости использования компетентностей для обеспечения роста и приложения их к задачам.

Różewski, B. Małachowski и J. Jankowski [13] рассмотрели концепцию DCMS динамических компетенций. Авторы утверждают, что реализация подобной концепции способна помочь компании учитывать будущие и текущие кадровые нужды.

L. Razmerita [14] предложил концепцию OntobUMf, делающую сильный контекст на поведении пользователя и соответствующего взаимодействия с ним.

K. Rezgui, H. Mhiri и K. Ghédira [15][16] проводят общий анализ компетентностных подходов в литературе и предлагают для этого решения для основанного на компетенциях электронного обучения. В той частности, авторы написали профилирующий компонент для системы электронного обучения Moodle, реализующий компетентностный подход.

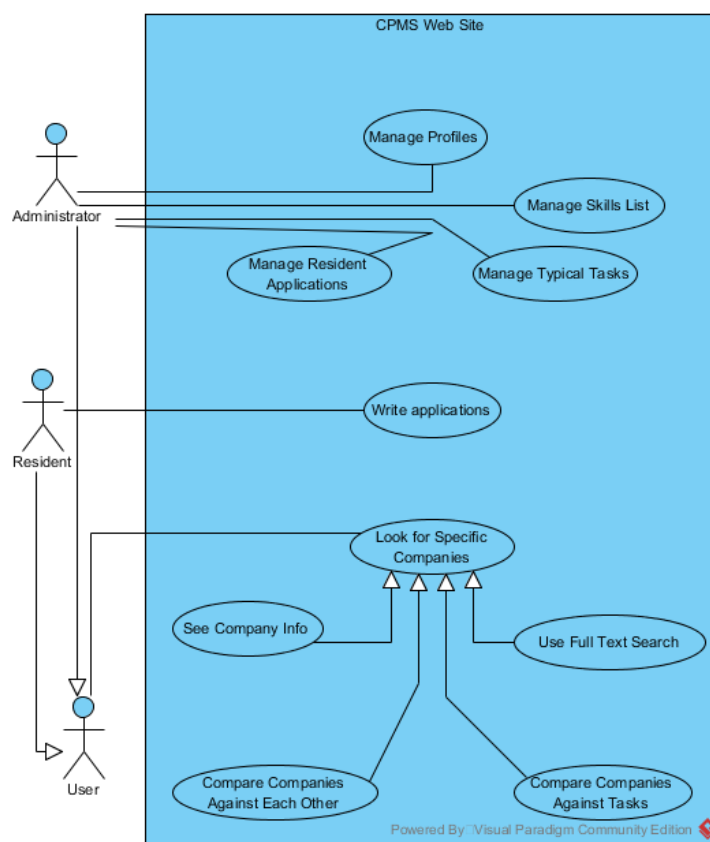
В общем доступе сложно найти открытые системы управления компетенциями. Мотивируется это, в основном, приложением данного типа ИС в корпоративных и академических системах, которые являются закрытыми. В той частности, в «Системе управления обучением» Университета ИТМО, находящейся по адресу «de.ifmo.ru», система сопоставления определённого количества баллов определённому предмету в профиле учащегося является вариантом реализации компетентностного подхода.

В общем доступе можно найти системы, в которых в профиль пользователя входит набор умений. Зачастую умения в этом наборе не имеют оценки, потому данные системы нельзя назвать системами управления компетенциями. В основном, это HR-системы, но не только. Среди них можно выделить популярный ресурс вопросов и ответов для разработчиков “Stack Overflow”. В данном случае пользователь может указать у себя в профиле технологии, которыми он владеет. Используя данные теги, для пользователя могут быть подобраны карьера из ресурса “Stack Overflow Careers”, а также отдельно подобраны вопросы, по которым данный пользователь мог бы дать ответы.

1.3 Функциональные требования

Исходя из основных сценариев использования системы (изображение 1), к разрабатываемой системе можно сформировать следующие функциональные требования:

- Хранение на английском и на русском языке в системе профилей компаний-резидентов, а также решаемых ими задач и умений, которыми они обладают.
- Разделение прав пользователей, выделяющее пользователей, резидентов и администраторов.
- Возможность для администраторов управления информацией через веб-приложение.
- Возможность для резидентов создания в системе заявок на внесение своей информации в систему.
- Возможность полнотекстового поиска по информации.
- Возможность сравнения профилей между собой.
- Возможность по задаче найти все профили, которые могут решить эту задачу.
- Возможность по профилю найти все задачи, которые он может решить.
- Возможность поиска профилей по их компетенциям. После выполнения операций сайт должен в читабельном виде предоставлять результаты выполнения пользователю.
- Поддержка полнотекстового (Full text) поиска.



Изображение 1 – Use Case диаграмма, демонстрирующая основные сценарии использования веб-приложения

1.4 Результаты анализа

В результате выполнения анализа были получены требования к приложению и основные сведения о предметной области. Были определены решаемые приложением задачи и варианты взаимодействия с ним. В приложении должно поддерживаться разграничение ролей, хранилище данных должно быть быстрым и масштабируемым, а в качестве тонкого клиента должен быть использован web – браузер.

2 Проектирование системы

2.1 Выбор технологий и инструментов разработки

Ниже указаны основные технологии, используемые при разработке системы, а также процесс их выбора.

- Язык программирования

Для выполнения задачи разработки потребуется язык, удовлетворяющий следующим требованиям:

- Это должен быть высокоуровневый язык программирования, чтобы обеспечить комфортную работу над предметной областью с минимальным написанием дублирующегося кода.
- Этот язык должен иметь поддерживаемые фреймворки для web – разработки чтобы использовать уже готовые компоненты вместо написания большинства стандартных компонентов с нуля.
- Т.к. над этим проектом потом, возможно, потребуется взять управление другим разработчикам, язык программирования должен быть легко читаемым и иметь встроенные возможности для документирования кода.

Для разработки web – приложений наиболее часто используются следующие высокоуровневые языки программирования:

- Java
- C#
- Ruby
- Python

Все эти языки программирования в результате в той или иной степени удовлетворяют поставленным требованиям. Ввиду этого было решено выбрать язык с точки зрения выбора дальнейших технологий для веб-разработки, простоты реализации веб-приложения с их использованием и осведомлённости с ними исполнителя. Таким образом для дальнейшей разработки был выбран язык программирования Java.

- Фреймворк

Экосреда Java имеет обильный выбор фреймворков, наиболее популярными среди которых являются:

- Spring
- JSF
- Struts
- GWT
- Play!
- Wicket
- Grails
- Vaadin

Все эти фреймворки имеют свои положительные и отрицательные стороны, поэтому в большинстве случаев сочетаются - на большой проект их может использоваться 2 или 3. Большинство из них равно удовлетворяет поставленным требованиям.

Предпочтение в данном случае было отдано Spring. Этот фреймворк активно поддерживается разработчиками, поэтому не смотря на свой возраст не является сильно устаревшим. Разработчики web – приложений, как правило, избегают Spring из-за слабой поддержки графических технологий, помогающих в разработке web – страниц. Для этого, как правило, Spring используется в сочетании с другим фреймворком. Тем не менее, требования к дизайну в данной работе отсутствуют, поэтому данный недостаток не имеет значения.

- Среда разработки

Разработчики Spring, Pivotal, выпускают свою собственную среду разработки на основе open-source среды разработки Eclipse. Т.к. разработчиками рекомендуется использовать именно данную среду для работы со Spring, для дальнейшей разработки была выбрана она.

- Система контроля версий

Существует ряд систем контроля версий, наиболее популярными среди которых являются SVN и Git. В данном случае был выбран Git. В качестве «сервера» использовался BitBucket. GUI приложение не устанавливалось, а управление системой контроля версий осуществлялось при помощи консольных команд.

- СУБД

Сформируем требования к СУБД.

- Эта СУБД должна быть реляционной.

- Эта СУБД должна иметь непроприетарную (бесплатную) лицензию.
- Эта СУБД должна иметь поддержку SQL – запросов из программного кода Java («драйвер», «коннектор»).

Есть две основные СУБД, удовлетворяющие поставленным требованиям:

- MySQL
- PostgreSQL

Выбор был сделан в пользу MySQL, так как с точки зрения поставленных задач эти продукты идентичны.

- Frontend – Фреймворк

Для простоты разработки удобного в использовании интерфейса веб-страницы, как правило, используется функционал фреймворков. В данном случае для разработки был выбран фреймворк Bootstrap, а также JQuery, на котором он основан. Bootstrap позволяет при помощи средств CSS, не работая напрямую с JavaScript, создавать масштабируемый дизайн с динамическими элементами. JQuery же даёт ряд инструментов, упрощающих написание скриптов JavaScript.

- Сервер приложений

После того, как приложение написано (а также в процессе разработки для демонстрации возможностей и отладки) является необходимым сервер приложений. Сервер приложений как обеспечивает работу самого веб-приложения, так и взаимодействует с пользователями этого веб-приложения, давая каждому из них виртуальный ресурс.

В экосреде Java существует огромное количество серверов приложений. Страница на сайте Wikipedia выделяет их несколько десятков. Наиболее популярными можно выделить:

- Jetty
- JBoss (WildFly)
- Tomcat

Инструментарий разработки Spring Boot также имеет встроенный сервер приложений Tomcat Embedded, призванный облегчить разработку веб-приложений за счёт автоматизации конфигурации. На время разработки будет использоваться именно данный сервер приложений, но ближе к запуску приложения будет задействован сервер Tomcat.

2.2 Системная архитектуры

Для реализации системы была выбрана классическая трёхуровневая архитектура приложения. Таким образом, система будет состоять из следующих компонентов:

1. Уровень данных (Persistency Layer)

На данном уровне должно быть реализовано хранение данных, соответствующее целям разработки. Оно должно допускать хранение данных в виде модели предметной области, аналогичной и схожей с представленной на Изображении 1, оно должно давать функционал полнотекстового поиска по данным для обеспечения удобства поиска информации в системе, также быть достаточно масштабируемым для веб-приложения. В ходе определения стека технологий была выбрана конкретная база данных, реализующая данный уровень.

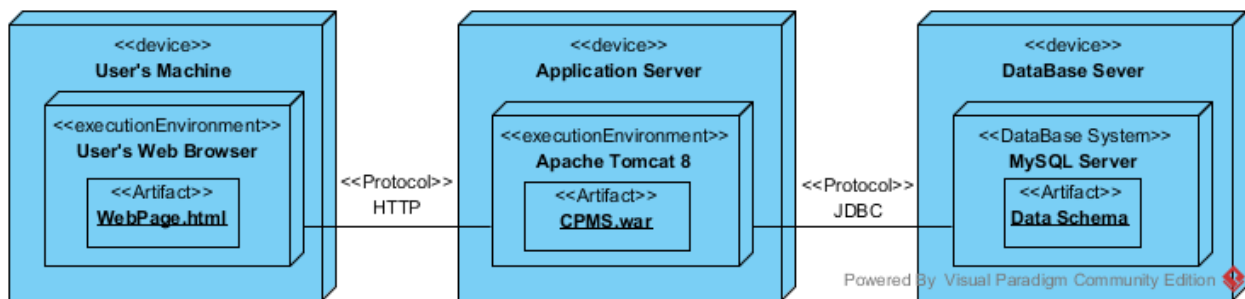
2. Уровень приложения (Application Layer)

На данном уровне должен быть реализован удобный доступ к данным, управление ими, а также презентация их пользователю. Помимо управления данными, на данном уровне происходит обеспечение взаимодействия с пользователем. В веб-приложении данное взаимодействие происходит за счёт элементов модели MVC – модели данных, (Model), представления (View) и контроллера (Controller).

Так как разрабатываемый программный код будет находится на данном уровне, то именно здесь должны быть реализованы требуемые операции взаимодействия с профилями компетенций и поиска по ним, а также политика безопасности и ограничение прав пользователей. Также, необходимо помнить, что представления (Views) – динамически генерируемые документы, генерация которых зависит от контекста взаимодействия с пользователем. Реализовывать данный функционал с нуля является тяжёлой задачей, и для её упрощения является целесообразным использовать фреймворк для веб-разработки. Подобные фреймворки дают удобную платформу для реализации и настройки функционала для разработки почти любого веб-приложения.

3. Уровень представления (Presentation Layer)

На данном уровне реализуется пользовательский интерфейс. Так как было сделано решение выполнить систему как web – приложение, интерфейс будет сделан в виде web – сайта. Данный пользовательский интерфейс должен быть выполнен исключительно для демонстраций возможностей системы т.к. последняя, скорее всего, будет интегрирована в web – приложение Технопарка Университета ИТМО. Архитектура приложения была изображена в нотации UML Deployment Diagram (Изображение 2).



Изображение 2 – Диаграмма UML Deployment, изображающая архитектуру приложения

2.3 Программная архитектура

Программная архитектура приложения, работающего на сервере приложений, должна будет состоять из следующих компонентов:

1. Информационные объекты (Сущности, Entities)

Удобное для использования в коде представление информации из базы данных.

2. Объекты доступа к данным (Data Access Objects, DAOs)

Для обеспечения корректной работы бизнес-логики объектов и правильной обработки ошибок базы данных, а также агрегирования и реализация функций доступа к объектам необходимы DAO. DAO будут реализованы для каждого необходимого информационного объекта и будут использоваться через интерфейсы.

3. Операции (Operations)

На каждую из указанных операций поиска по профилям компетенций целесообразно выделить отдельный интерфейс, от которого могли бы наследоваться различные реализации этих операций.

4. Фасад (Façade)

Реализация паттерна проектирования (Façade), используется с целью агрегировать все необходимые операции и DAO в одном месте, допустив не только возможность удобно ими пользоваться другим классам, но и удобно перенастраивать (переопределять).

5. Интерфейс службы безопасности (Security)

Для правильной работы с пользователями и их правами необходим интерфейс, который предоставлял бы все необходимые для этого функции.

6. Контекст пользователя

Для выполнения операций между конкретными действиями, требуемыми от пользователя, необходимо будет содержать временную информацию. С этой целью необходимо завести специальный объект, где можно было бы хранить всю временную информацию, необходимую пользователю.

7. Контроллеры (Controlllers)

Объекты без состояния, используемые для взаимодействия с пользователями системы. Получают от последних сигналы посредством протокола http, в ответ на что возвращают ему представления.

8. Представления (View)

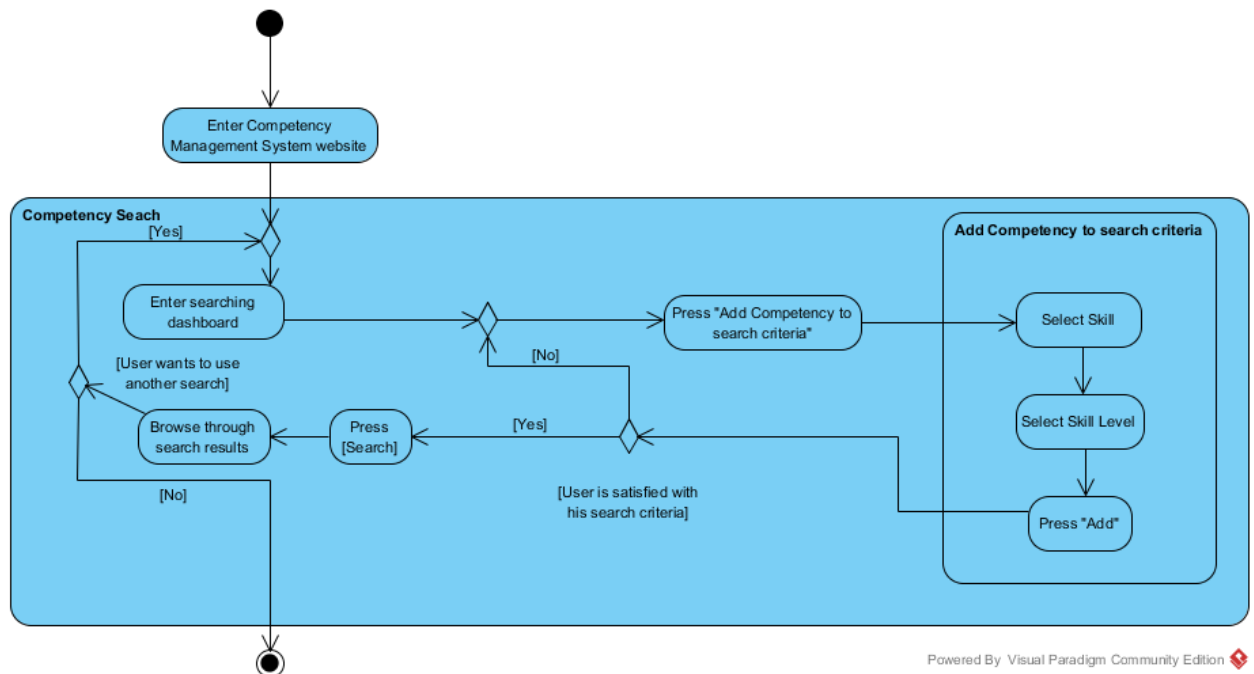
Документы в разметке HTML, динамически генерируемые в зависимости от контекста взаимодействия с пользователем и используемые для графического взаимодействия с ним.

Полученная программная архитектура изображена в приложении А.

2.4 Основные сценарии использования

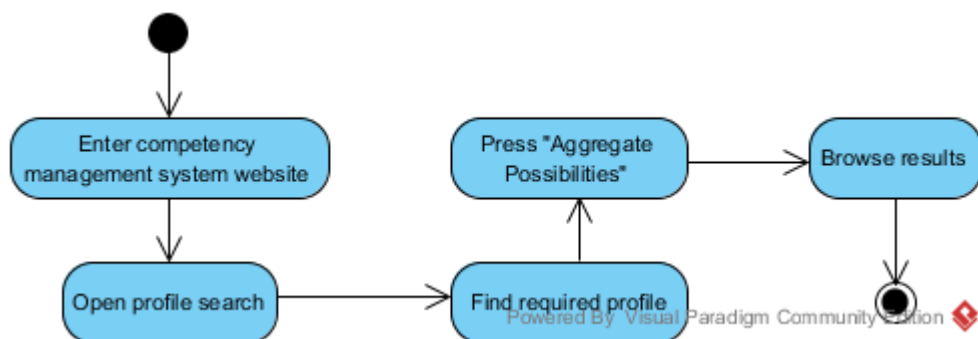
Рассмотрим основные сценарии использования системы.

1. Предположим случай, в котором пользователь знает, какие компетенции ему потребуются от субъекта, которого он ищет. В таком случае пользователь может воспользоваться поиском по компетенциям. Данный поиск найдёт все профили, метрика сравнения которых с критерием, заданным пользователем, будет отличаться от идеальной не больше, чем на допустимый интервал. Также существует альтернативная версия данного поиска, которая не будет отсеивать профили, но отсортирует все профили по степени их схожести с критерием. Сценарий использования в нотации UML Activity Diagram можно увидеть на Изображении 3.



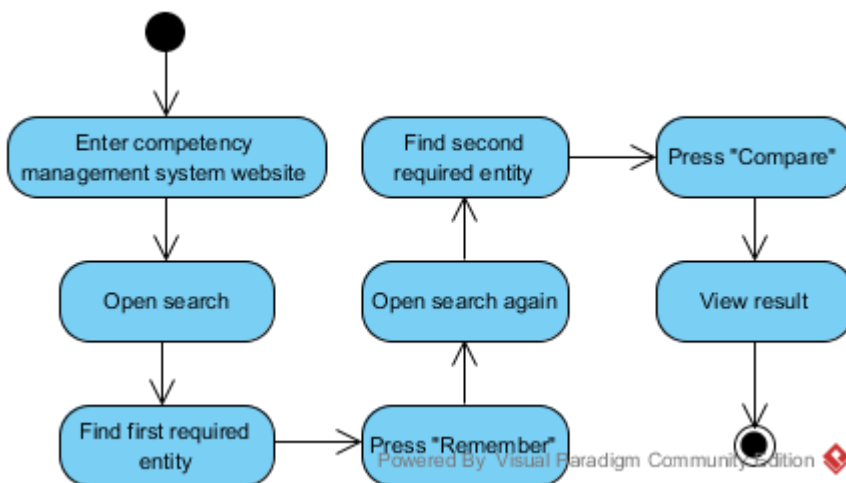
Изображение 3 – Сценарий использования поиска по компетенциям

2. Предположим обратный случай, когда пользователь знает субъекта, но хочет узнать его возможности. В данном случае пользователь может воспользоваться такой функцией, как «агрегация возможностей». Данная функция позволяет узнать все задачи (Task), которые может выполнить субъект. Сценарий в нотации UML Activity Diagram можно увидеть на Изображении 4.



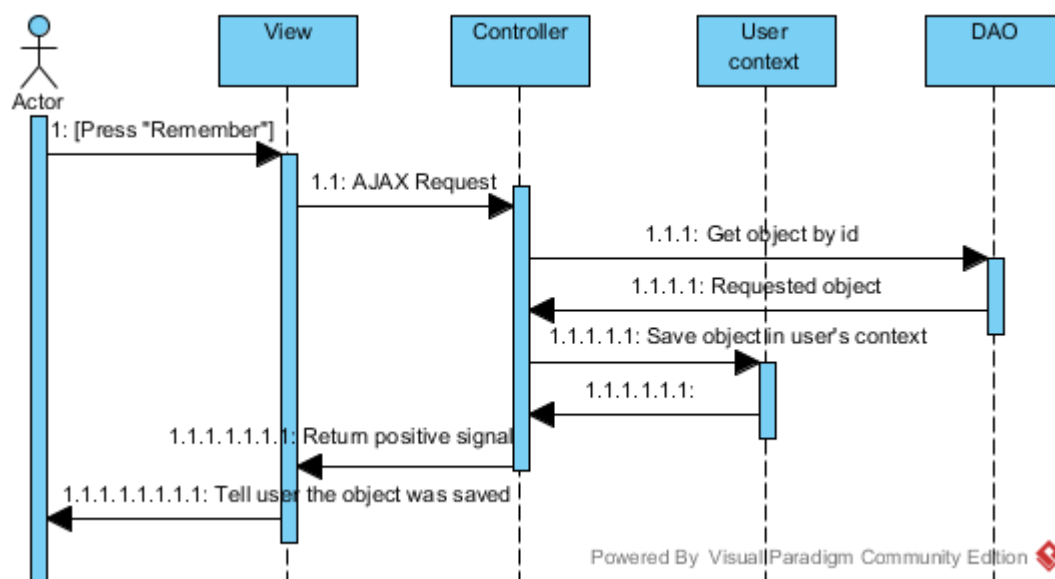
Изображение 4 – Сценарий использования агрегирования возможностей профиля

3. Также, возможен случай, когда пользователь знает либо два профиля, которые он хочет сравнить, либо профиль и задачу. В данном случае его действия сводятся к тому, чтобы сохранить в системе один из сравниваемых объектов, после чего найти второй, нажать на кнопку «сравнить» и получить результаты. Диаграмму в нотации UML Activity Diagram, иллюстрирующую выполнения данного сценария, можно увидеть на Изображении 5.



Изображение 5 – Сценарий сравнения одного из информационных объектов с другим, сохранённым в системе

Чтобы продемонстрировать внутреннюю работу процесса сохранения объекта в контексте пользователя, была выполнена диаграмма в нотации UML Sequence Diagram, увидеть её можно на Изображении 6.



Изображение 6 – Диаграмма Sequence, раскрывающая процесс сохранения объекта в контексте пользователя для дальнейшего использования

2.5 Результаты проектирования системы

В результате проектирования был получен исчерпывающий набор диаграмм UML, описывающих структуру и функциональную реализацию разрабатываемого приложения. Данные диаграммы будут использоваться при реализации архитектуры системы и написании программного кода.

3 Реализация системы

3.1 Порядок проверок и приёма

При разработке подобной системы является важным регулярное общение с заказчиком с целью своевременного обнаружения ошибок, несоответствий требованиям и своевременного внесения правок. На протяжении выполнения работы у исполнителя будет возможность регулярно общаться с заказчиком по электронной почте. Также, заказчик и исполнитель условились каждую пятницу устраивать встречи для демонстрации результатов недели и их обсуждения.

Для формализации требований и их закрепления на начальном этапе выполнения работы было составлено и согласовано ТЗ.

Приём приложения будет осуществляться поэтапно совместно с его разработкой. Окончательный приём приложения планируется осуществить не позднее защиты дипломной работы.

3.2 Реализация модели данных и взаимодействие с базой данных

Драйвер базы данных даёт возможность выполнять SQL – запросы напрямую в среде Java, также предоставляет специальные контейнеры для работы с результатами выполнения этих запросов. Тем не менее, напрямую SQL используется только в том случае, если надо написать высоконагруженную систему, в которой важна скорость работы обмена данными. В большинстве остальных случаев применяются реализации ORM.

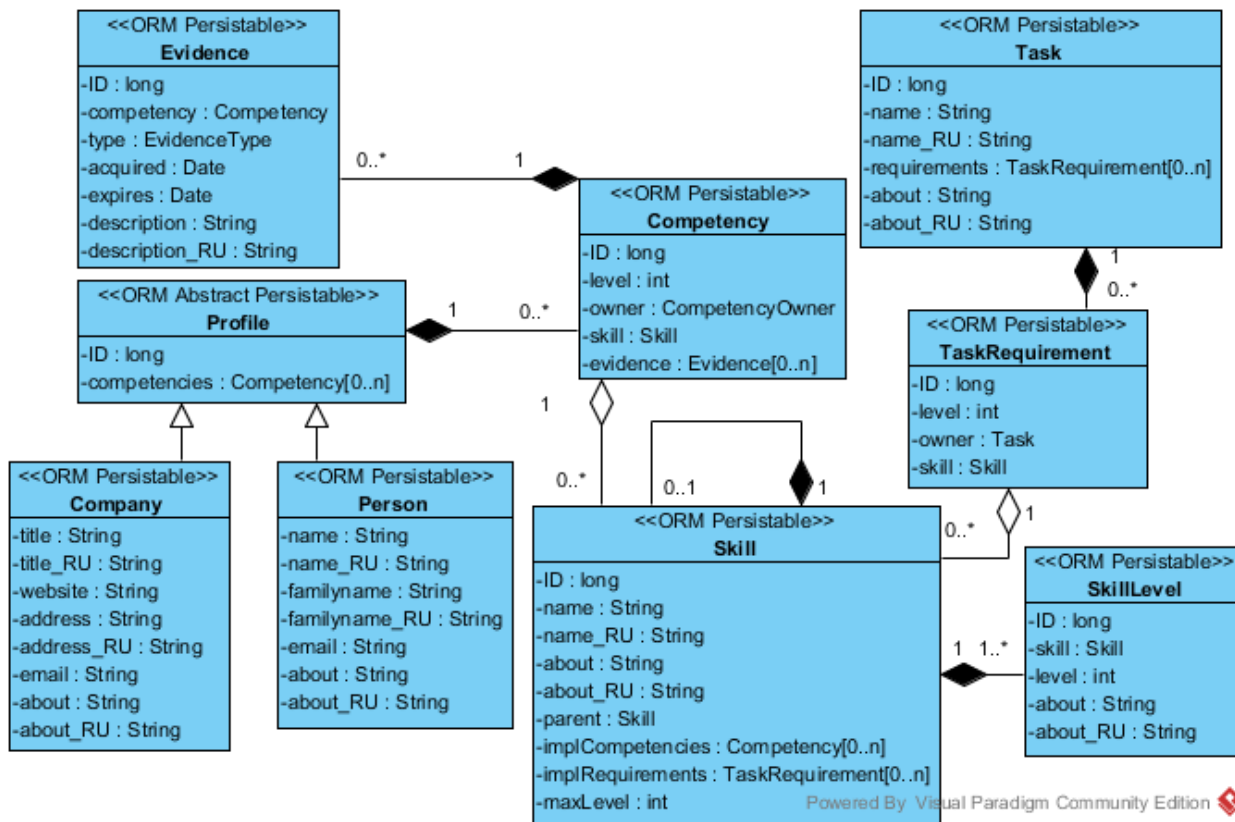
ORM (Object-Relational Mapping) позволяет абстрагироваться от задачи написания модели данных (Schema) и задачи написания запросов к данным и работать напрямую с информационными объектами (Entity), что убирает значительный объём работ и большое количество дублирующегося кода. Как уже ранее упоминалось, использование подобных технологий ресурсозатратно и зачастую недостаточно гибко, однако в рамках данного проекта оно оправдано.

Spring Framework предоставляет свою реализацию Java Persistence API (JPA), называемую Spring Data JPA. Также, набор зависимостей Spring Boot предоставляет Hibernate, являющийся одним из самых популярных ORM в среде Java. Эти две технологии дают достаточный набор инструментов для реализации необходимого функционала.

Модель данных, реализуемая при помощи ORM, показана на изображении 7.

Следует упомянуть, что одним из преимуществ ORM является автоматизация каскадных операций – то есть операций, которые должны быть применены к зависимым сущностям, если они применены к владельцам. Какие в данном случае операции были автоматизированы при помощи средств ORM между сущностями – «целыми» и сущностями – «частями»:

- Композиция
 - Удаление «сирот» (Orphan Removal) - в случае, если сущность – «часть» была отсоединена от своего «целого», она будет автоматически удалена из базы данных.

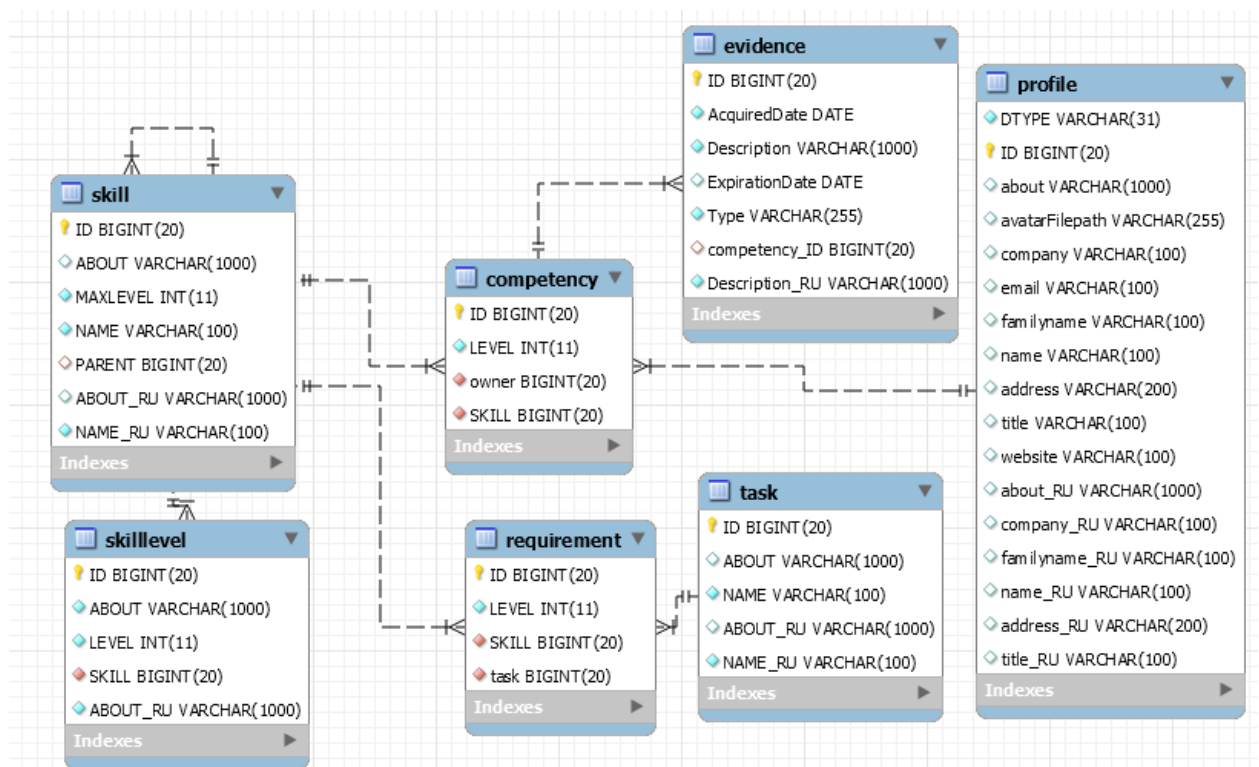


Изображение 7 – Диаграмма Class, раскрывающая реализуемую модель данных

- Сохранение и обновление (SaveUpdate) – в случае, если сущность будет сохранена или обновлена в базе данных, то её зависимые сущности также будут сохранены или обновлены.
- Удаление (Delete) – если сущность будет удалена, то её зависимые сущности будут удалены.
- Сохранение состояния (Persist) – в случае, если созданная, но ещё не сохранённая в базе данных, сущность будет сохранена в базе данных, то её зависимые сущности также будут сохранены в базе данных.

- Соединение (Merge) – операция копирует состояние несохранённого в базе данных объекта в объект, сохранённый в базе данных с таким же идентификатором, и делает то же самое с его зависимыми сущностями.
- Отсоединение (Detach) – в случае, если сущность будет «отсоединена» от базы данных (её состояние не будет синхронизироваться с её состоянием в базе данных), то её зависимые сущности также будут «отсоединены».
- Агрегация – используются отношения отсоединения (Detach) и удаления (Delete), но распространяются не от родительской сущности к зависимым, а наоборот.

В результате выполнения данной модели была сгенерирована SQL – схема. В MySQL Workbench была вызвана функция “Reverse Engineer”, которая составила диаграмму получившейся схемы данных (Изображение 8).



Изображение 8 – Схема базы данных, сгенерированной средствами ORM, восстановленная Reverse Engineer

Следующим шагом в написании уровня доступа к данным является написание сервисов (Service), или DAO (Data Access Object) объектов. Данный шаг, как правило, делается в связи с необходимостью обеспечить проверку целостности данных при их вводе / выводе, а также чтобы удобно агрегировать функционал ORM. Помимо этого, считается плохим решением получать данные в веб-приложении напрямую из хранилищ (Repository) ORM. В связи с этим,

после написания модели данных был создан общий DAO интерфейс. Он содержал следующие методы:

- `long count()` – возвращает суммарное количество хранимых объектов.
- `List<T> getRange(long from, long to)` – возвращает объекты в порядке их хранения начиная от `from` и заканчивая `to`.
- `List<T> getAll()` – возвращает все хранимые объекты.
- `List<T> search(String request)` – возвращает все объекты, удовлетворяющие полнотекстовому запросу `request`.
- `List<T> searchRange(String request, int from, int to)` – возвращает `to-from` объектов, пропуская первая `to` объектов из всех, что были найдены по полнотекстовому запросу `request`.
- `long searchCount(String request)` – возвращает количество объектов, найденных по полнотекстовому поиску `request`.
- `T getOne(long id)` – возвращает объект с идентификатором `id`, или `null`, если такой не был найден.
- `T insert(T newInstance)` – сохраняет новый объект `newInstance` в базе данных, возвращая его сохранённую версию.
- `T update(T oldInstance)` – обновляет хранимый в базе данных объект `oldInstance`, возвращая его обновлённую версию.
- `void delete(T instance)` – удаляет из базы данных объект `instance`.
- `void rebuildIndex()` – перестраивает индекс полнотекстового поиска.

От данного интерфейса были унаследованы и созданы 3 DAO сервиса для сущностей `Profile`, `Skill` и `Task` соответственно, а остальные сущности извлекались из коллекций возвращённых объектов как зависимые.

3.3 Реализация полнотекстового поиска

Пользователю понадобится как-то находить ключевые объекты системы – `Profile`, `Skill` и `Task`. Специально с такой целью целесообразно реализовывать полнотекстовый поиск. В `Hibernate` уже встроен `Apache Lucene`, необходимо только его настроить.

`Lucene` хранит необходимую ему для поиска информацию в виде индексов, которые в свою очередь хранятся в файловой системе. Для этой цели `Lucene` была выделена отдельная папка на физическом хранилище.

Важной частью в настройке Lucene является настройка анализатора – инструмента, который определит, какая информация и как будет храниться в индексе. Встроенный анализатор предназначен для тех случаев, когда информация ищется программно, а не человеком. Поэтому, был сделан новый анализатор исходя из индексируемых полей.

Так как в данном случае в качестве индексируемых полей используются только названия (компаний-резидентов, умений, задач), то все обнаруженные в тексте специальные символы будут частями названий (“C++”, “C#”), поэтому все специальные символы при анализе вносились в индекс, а фразы разбивались на слова только по пробелам. Также, при анализе весь текст был переведён в нижний регистр для создания поиска, нечувствительного к регистру.

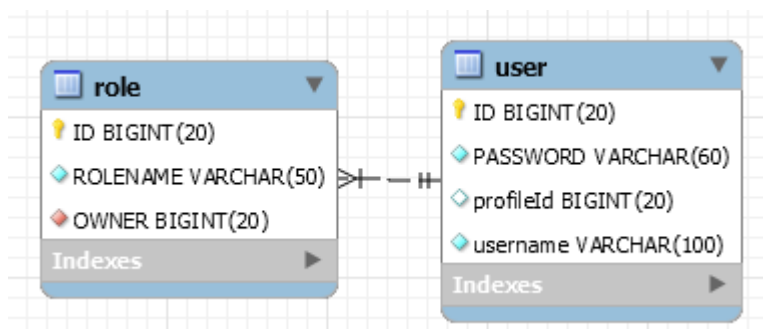
Когда от пользователя происходил поисковый запрос, текст запроса разбивался на слова, приводился в нижний регистр, после чего каждое слово оборачивалось специальным символом «*» в начале и в конце, что позволяло искать не только само слово, но и вхождения данного слова в другие слова.

После того, как поля Java классов, по которым необходимо вести поиск, будут помечены соответствующей аннотацией, Lucene начнёт записывать информацию для дальнейшего использования при поиске, но только в случае если объект был внесён в базу данных через интерфейс Hibernate. В некоторых случаях может создать рассинхронизацию индексов Lucene с актуальной информацией, записанной в базе данных. Специально для этого у каждого из имеющихся DAO был реализован метод, заставляющий Lucene стирать свои индексы и создавать их с нуля опираясь на информацию в базе данных.

3.4 Безопасность, права доступа и аккаунты пользователей

Профиль компании не был запланирован как аккаунт пользователя т.к. жизненные циклы этих двух объектов могут быть друг с другом не связаны. Так, например, у обладателей аккаунта администратора не может быть своего профиля. Поэтому, для реализации пользователей и разграничения их прав необходимо создать соответствующие сущности, после чего задействовать сервис авторизация.

Spring MVC имеет модуль Spring Security, который позволяет настраивать готовое решение сервиса авторизации и управлять правами доступа для пользователей. Причём в качестве пользователя может выступать любой объект, реализующий соответствующий интерфейс. В рамках данного проекта был реализован интерфейс сервиса пользователей, позволяющий Spring Security взаимодействовать с пользователями из базы данных. В базе данных была создана специальная схема для пользователей и ролей. Reverse engineer этой схемы, выполненной также при помощи средств ORM, можно увидеть на изображении 9.



Изображение 9 – Схема базы данных пользователей системы, сгенерированной средствами ORM, восстановленная Reverse Engineer

Среди пользователей системы можно выделить следующие группы и соответствующие им роли:

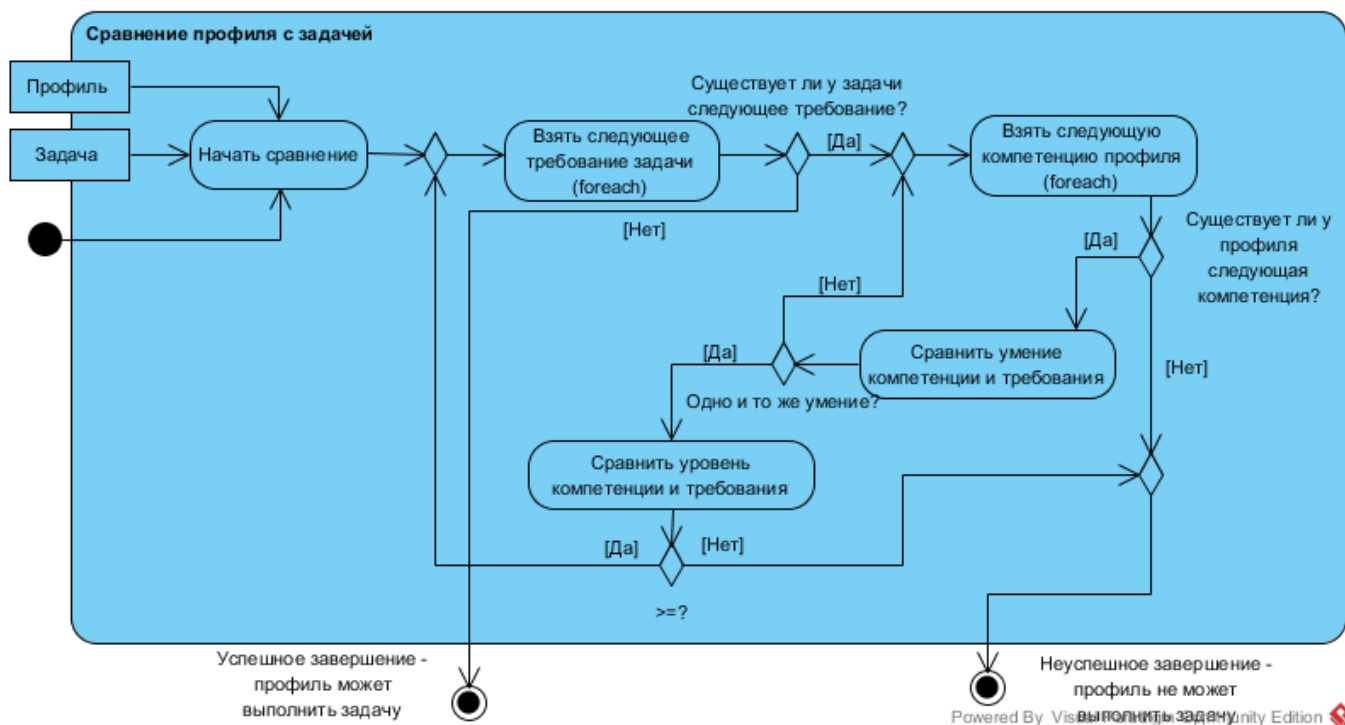
1. Пользователь системы – любой человек, использующий общедоступный функционал системы. Т.к. регистрироваться ради использования общедоступного функционала не является целесообразным, было решено не создавать для пользователей системы отдельной роли.
 2. Резидент системы – человек, имеющий отношение к зарегистрированной в системе компании – резиденту. У данного пользователя должна быть возможность вносить в систему информацию о своих компетенциях. Для таких пользователей предусмотрена роль «Resident».
 3. Администраторы системы – человек, который может использовать администраторский функционал системы, в том числе свободно вносить и удалять информацию. Для таких пользователей предусмотрена роль «Admin».
- Данные роли были использованы при настройке авторизации веб-приложения.

3.5 Реализация операций поиска и сравнения

- Сравнение профиля с задачей и поиск профилей по задаче

Сравнение профиля с задачей необходимо, чтобы понять, может ли профиль выполнить поставленную задачу.

Реализация алгоритма: Для входных параметров – профиля Р и задачи Т, каждое требование Т(і) задачи Т сравнивается с компетенциями профиля Р. Если есть хотя бы одна компетенция Р(і) профиля Р, умение S которой равно умению требования задачи Т(і), а уровень Р(і).level этой компетенции больше или равен уровню Т(і).level соответствующей задачи, то профиль соответствует этому требованию. В противном случае, если уровень Р(і).level этой компетенции меньше уровня Т(і).level или равен нулю, то данный профиль не удовлетворяет требованию задачи. Если профиль Р удовлетворяет всем требованиям задачи Т, то этот профиль может выполнить эту задачу. Если профиль Р не удовлетворяет одному или более требованиям задачи Т, то этот профиль не может выполнить эту задачу. Диаграмма UML, демонстрирующая данный алгоритм, показана на изображении 10.



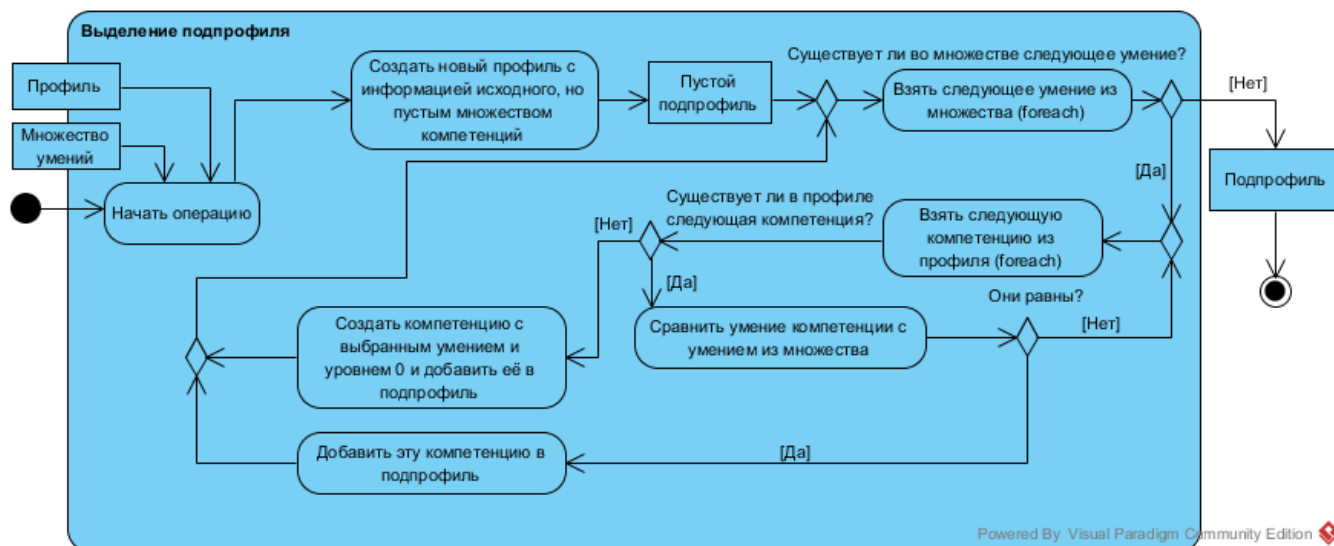
Изображение 10 – Диаграмма UML Activity, изображающее сравнение профиля и задачи по предоставленному алгоритму

Данный алгоритм сравнения может использоваться для поиска всех профилей, которые могут выполнить определённую задачу, или наоборот – всех задач, которые может выполнять профиль.

- Выделение подпрофиля из профиля

Выделение подпрофиля – операция, которая создаёт из множества компетенций профиля новое множество, ограниченное каким-то набором умений. Данная операция, в основном, используется другими операциями для упрощения их использования, но также может быть полезна пользователю. Например, если пользователю необходимо свести все компетенции какого-то профиля или профилей к множеству из только тех, которые ему нужны.

Реализация алгоритма: в качестве входных параметров используется профиль P и множество умений $Set<S>$. Для каждого умения $S(i)$ из множества $Set<S>$ перебирается каждая компетенция $C(i)$ из множества компетенций профиля P . Если была найдена такая компетенция $C(i)$, умение которой равно умению $S(i)$, то эта компетенция добавляется ко множеству компетенций подпрофиля P_s . Если такой компетенции найдено небыло, создаётся новая компетенция, уровень которой устанавливается равным нулю, после чего эта компетенция добавляется к подпрофилю P_s . Диаграмма UML, демонстрирующая данный алгоритм, показана на изображении 11.



Изображение 11 – Диаграмма UML Activity, изображающее выделение подпрофиля по предоставленному алгоритму

- Сравнение одного множества компетенций с другим

Сравнение одного множества компетенций с другим применяется если необходимо сравнить друг с другом два профиля или профиль с созданным пользователем множеством компетенций. Результатом такого сравнения является число, показывающее степень соответствия одного множества другому. В ходе данной работы было решено в качестве результата сравнения использовать дробные числа от 0 до 1, чтобы их было удобно переводить

в проценты. Соответственно, число 0 означает 0% соответствия, или полное различие между двумя множествами компетенций. Число 1 означает 100% соответствия, когда два множества равны друг другу. Для сравнения двух множеств компетенций применялся следующий алгоритм:

Сначала, выделялось общее для этих двух множеств компетенций множество умений. Таким образом, в получившемся множестве умений каждое умение можно было найти либо в одном из двух множеств компетенций, либо в обоих сразу. После этого, при помощи операции выделения подпрофиля оба профиля относительно получившегося множества умений приводились к общему виду. После этого, они сравнивались по следующей формуле (1):

$$R = 1 - \frac{\sum_{S \in (P1.C, P2.C)} \frac{|Level_{P1.C(S)} - Level_{P2.C(S)}|}{Max(Level_{P1.C(S)} - Level_{P2.C(S)})}}{Summ(S)} \quad (1)$$

Где:

«R» – результат, число от 0 до 1;

«S ∈ (P1.C, P2.C)» - каждое умение из общего множества умений для множеств компетенций двух сравниваемых профилей;

Level_{Px.C(S)} – уровень владения умением S для профиля Px;

Summ(S) – общая сумма умений в множестве общих умений.

Операция сравнения одного множества компетенций с другим применяется не только для сравнения двух профилей друг с другом, но также и для поиска по компетенциям (ранжирования профилей). В данном случае пользователь указывает предикат в виде сформированного им набора компетенций а система возвращает ему результат сравнения всех профилей компетенций в системе, отсортированный по степени соответствия. Пользователь также может использовать такой поиск, если ему понадобится найти все компании, хорошо владеющие каким-то конкретным умением или набором каких-то умений.

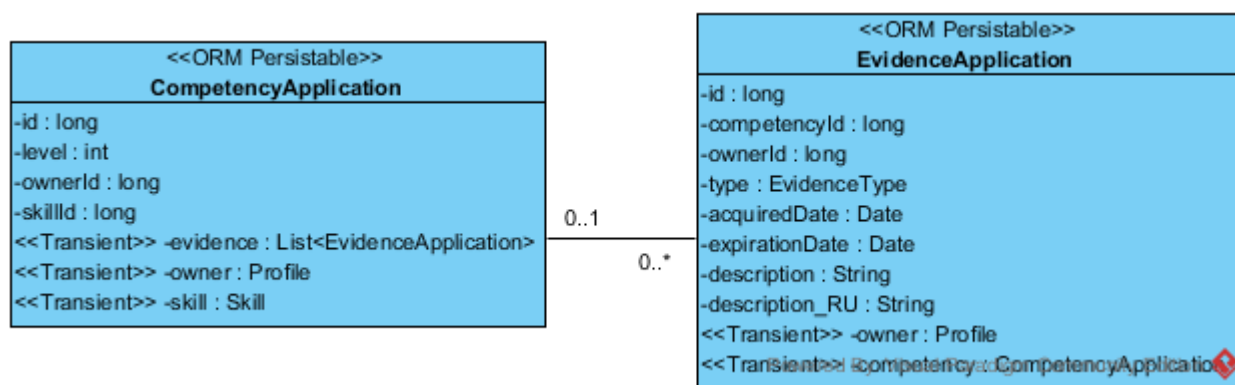
3.6 Реализация компонента заявок

Компетенции документируют состояние возможностей субъекта только на определённый момент времени. Таким образом, можно говорить о том, что со временем компетенции могут устаревать, в связи с чем резидент может пожелать их обновить. Также, у резидента системы со временем могут появляться новые компетенции в связи с расширением его деятельности. Уведомлять каждый раз администрацию напрямую будет резиденту тяжело, в связи с чем он может отказаться от пользования системой. Также, является нецелесообразным давать

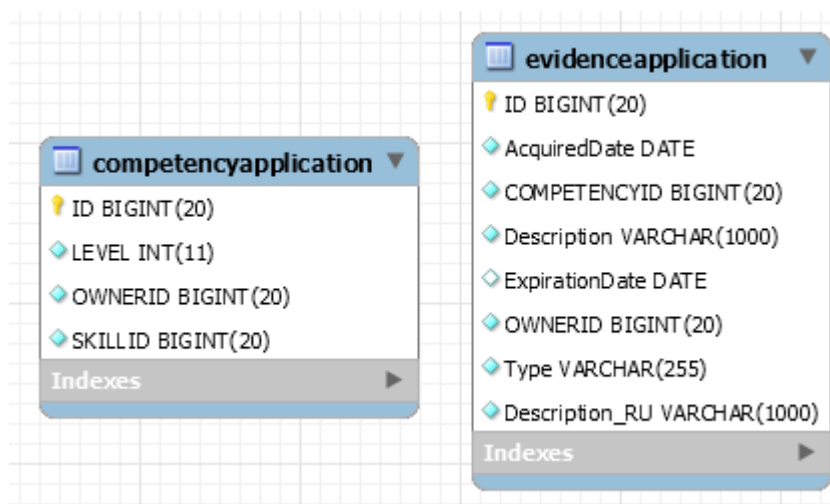
пользователю редактировать информацию в системе напрямую. Поэтому, было решено пойти на компромисс и создать компонент заявок.

Компонент заявок позволяет пользователю вносить информацию о своих компетенциях на рассмотрение администратора напрямую через систему. Администратор по результатам просмотра может принять заявку (отразить в системе изменения), или отклонить её.

Для хранения заявок было решено также использовать Hibernate. Формат ORM – сущностей, хранимых в системе, можно увидеть на изображении 12, а схему получившейся на основании этой схемы базы данных – на изображении 13.



Изображение 12 – Модель данных компонента заявок



Изображение 13 – Схема базы данных заявок, сгенерированной средствами ORM, восстановленная Reverse Engineer

Стоит заметить, как в данном случае реализованы зависимости между сущностями – система не даёт Hibernate управлять зависимостями, а делает это сама. Во время реализации на это были две причины. Во-первых, тогда было неизвестно, как реализуемая система будет взаимодействовать с официальным сайтом Технопарка ИТМО. Рассматривался вариант, что в

результате система была бы RESTful – сервисом, а заявки хранились бы в отдельной базе от основной схемы. Во-вторых, причина в том, что было решено дать пользователю возможность подавать заявки как на создание отдельных подтверждений компетенций, так и компетенций как с этими подтверждениями, так и без. Таким образом, их жизненные циклы почти никак не связаны.

Основные методы интерфейса сервиса заявок приведены на схеме программной архитектуры, полученной на этапе проектирования. Ниже детально приведены все его методы после реализации.

- `void suggestCompetency(CompetencyApplication application)` – сохраняет в базу данных заявку на создание или обновление компетенции обладателя заявки.
- `List<CompetencyApplication> retrieveCompetencies()` – возвращает все сохранённые в системе заявки компетенций для рассмотрения администратором.
- `List<CompetencyApplication> retrieveCompetencies(long ownerId)` – возвращает все заявки компетенций, внесённые в систему обладателем профиля с PK ownerId.
- `CompetencyApplication retrieveCompetency(long applicationId)` – возвращает конкретную заявку по её PK.
- `void deleteCompetency(long applicationId)` – удаляет сохранённую заявку по её PK.
- `void approveCompetency(long applicationId)` – извлекает из системы заявку по её PK. Если такая заявка существует, она будет принята. Если у обладателя этой заявки есть компетенция с таким же умением, как и эта заявка, то эта компетенция будет обновлена – её уровень изменён, а подтверждения заявки станут новыми подтверждениями компетенции. Если компетенции с таким умением нет, то эта заявка станет новой компетенцией.
- `void suggestEvidence(EvidenceApplication application)` – сохраняет в базу данных заявку на создание нового подтверждения компетенции.
- `List<EvidenceApplication> retrieveEvidence()` – возвращает все сохранённые в системе заявки подтверждений компетенций на рассмотрение администратору.
- `List<EvidenceApplication> retrieveEvidence(long ownerId)` – возвращает все сохранённые в системе заявки подтверждений компетенций, внесённые обладателем профиля с PK ownerId.
- `EvidenceApplication retrieveEvidenceById(long applicationId)` – возвращает заявку подтверждения компетенции с PK applicationId.

- `void deleteEvidence(long applicationId)` – удаляет заявку подтверждения компетенции с PK `applicationId`.

3.7 Реализация веб-приложения

После того, как модель данных и сервисы доступа к ней реализованы, остаётся только написать взаимодействие с пользователем. В парадигме MVC, это уровни V и C – контроллеры и представления. Функционал контроллеров и представлений в Spring MVC реализуется легко. Для того, чтобы реализовать контроллер, достаточно создать Java класс, а затем пометить его соответствующими аннотациями. Таким образом также и настраиваются пути в веб-приложении.

Методы контроллера в Spring MVC возвращают `String`, который указывает имя шаблона представления (view template). Этот шаблон представления необходимо найти обработчику представлений (view resolver), сгенерировать из этого шаблона полноценную веб-страницу и передать её клиенту.

В эко среде Java два наиболее популярных обработчика представлений – Thymeleaf и Velocity. Они в равной степени удовлетворяют требованиям, но выбран был Thymeleaf. Он является наиболее современным и рекомендуется командой разработки Spring.

Ниже приводятся основные пути реализованного веб-приложения и даётся краткое описание реализуемого ими функционала:

1. “/” (корень приложения)

Страница, куда пользователь будет попадать по приходу на сайт. Объясняет ему основы работы с системой и показывает, куда далее он может пойти, а также как пользоваться навигационной полосой (navbar).

2. “/viewer/”

Содержит списки корневых сущностей – профилей, задач и умений, и разбивает их на страницы, по 10 на штуку. Также даёт возможность администратору создавать новые сущности. Здесь также находится поля для полнотекстового поиска.

Индивидуальные пути для каждой сущности с «личной» страницей также расположены здесь. Таким образом, сущности можно просматривать, а также «запоминать» в контексте пользователя для каких-то дальнейших действий. Администратор может здесь их изменять.

Попасть на списки можно через навигационную полосу, на страницы сущностей через ссылки в списках.

3. “/editor/”

Содержит формы для создания и изменения сущностей, а также POST – адреса для них же. Доступ ко всем страницам по этому адресу доступен только пользователю, находящемуся в группе администраторов.

4. “/security/”

Содержит набор инструментов по работе с пользователем. Позволяет аутентифицироваться и выходить из системы любому пользователю, а администратору – создавать новых пользователей. На страницу аутентификации возможно попасть, нажав на ссылку на профиль пользователя, не войдя в систему. В профиле пользователя находится ссылка на выход из аккаунта.

5. “/skillTree”

Первоначально, планировалось использовать списки для просмотра умений в веб-приложении. Так как умения хранятся в виде дерева (у каждого умения есть либо 1, либо 0 умений-родителей), такой подход оказался сложным для администраторов системы незнакомых со внутренним устройством модели данных. Поэтому было решено создать отдельную страницу, на которой умения были бы расположены в виде дерева. Для этого использовался открытый код dtree.js. Впоследствии оказалось, что подобный интерфейс был удобен не только для администраторов, но и для пользователей. Поэтому было решено полностью перенести весь функционал по взаимодействию с умениями на эту страницу и сделать её управляемой через AJAX.

Попасть на данную страницу возможно с любой другой через ссылку на навигационной полосе.

6. “/dashboard”

Для всего функционала, для которого было нецелесообразно создавать отдельные страницы, была создана страница “dashboard”. Здесь пользователи могут просмотреть сущности, сохранённые в их контекстах пользователя, выполнить с ними некоторые операции поиска. Для резидентов именно здесь находится возможность подать заявки на добавлении к их профилю новых компетенций, а для администратора здесь доступна возможность эти заявки просматривать и принимать/отклонять их.

Попасть на данную страницу возможно с любой другой через ссылку на навигационной полоске.

3.8 Результаты реализации системы

В процессе реализации, по предоставленной в предыдущем разделе системной и программной архитектуре было получено работоспособное веб-приложение, готовое к запуску на сервере приложений. Функционал данного приложения отвечает всем поставленным требованиям и способен решать поставленные перед ним задачи.

4 Тестирование системы

4.1 Формирование требований к тестированию и определение используемых для тестирования технологий

На предыдущем шаге было разработано удовлетворяющее функциональным требованиям приложение. Чтобы убедиться в том, что оно способно в работающем состоянии выполнять как функциональные, так и остальные требования, необходимо его протестировать.

Т.к. к приложению небыло поставлено требований по скорости взаимодействия с базой данных и отклика интерфейса (в разумных пределах), является необходимым только убедиться в его полной работоспособности.

Можно выделить следующие основные компоненты приложения, тестирование которых необходимо провести:

1. Модель данных и сервисы, предоставляющие интерфейс взаимодействия с ней. Необходимо протестировать корректность основных операций взаимодействия с хранилищем данных – сохранение, извлечение, поиск, изменение, удаление. Это охватывает все метода интерфейса IDAO.
2. Реализации алгоритмов поиска и сравнения. Для каждой реализации каждого алгоритма необходимо привести несколько случаев его работы и убедиться в их правильном выполнении.
3. Работа полнотекстового поиска. Необходимо убедиться, что для конкретного запроса поиск вернёт ожидаемые результаты.
4. Работа компонента заявок. Необходимо убедиться как в корректности работы хранения заявок, так и корректности приёма/отказа заявок.
5. Работа веб-приложения. Необходимо проверить корректность взаимодействия front-end и back-end частей для каждой веб-страницы. Данные тесты также должны покрыть работу разграничения прав доступа.
6. Корректность вёрстки и другое ручное тестирование. Необходимо открыть каждую веб-страницу в окне веб-браузера разных размеров и убедиться, что это не мешает работе с веб-приложением.

Пятый пункт будет необходимо выполнить при помощи ручного тестирования, остальные пункты тестирования возможно автоматизировать при помощи специальных наборов инструментов тестирования.

Spring Boot предоставляет некоторые технологии для тестирования, в той частности JUnit покрывает все unit-тесты и большинство интеграционных, за исключением тестов веб-приложения. Для тестирования веб-приложения команда разработки Spring советует использовать инструмент HtmlUnit совместно с Selenium WebDriver. Данные технологии дают возможность эмулировать работу веб-браузера и взаимодействовать с полученными веб-страницами при помощи нативного языка программирования, в данном случае Java.

4.2 Тестирование модели данных

Работа с моделью данных обеспечивается через классы-сервисы, реализующие описанный ранее интерфейс IDAO. Поэтому, тестироваться будут именно реализации данного интерфейса для соответствующих сущностей – для Skill (и вложенной сущности SkillLevel), для Task (и вложенной сущности TaskRequirement) и для абстрактной сущности Profile (реализованная сущностями Company и Person и имеющая вложенные сущности Competency и Evidence). Ниже приводится краткое описание реализованных тестов. В процессе работы тестирование иногда проводилось на встроенной базе данных H2 для ускорения работы, а иногда на базе данных MySQL.

1. Тестирование реализации интерфейса IDAO для сущности Skill (Умение)
 - 1.1. Умение возможно сохранить, а потом извлечь.
 - 1.2. Умение возможно сохранить, а потом изменить (два теста).
 - 1.3. Умение возможно сохранить, а потом удалить.
 - 1.4. Извлечение несуществующего умения вернёт null.
 - 1.5. Умение невозможно сохранить, если его Parent – несуществующее умение.
 - 1.6. В случае удаления умения его Children – умения не будут удалены.
 - 1.7. Возможность создать у умения уровень, а потом извлечь его через извлечение умения.
 - 1.8. Возможность создать у умения уровень, а потом изменить этот уровень.
 - 1.9. Возможность создать у умения уровень, а потом удалить этот уровень.
 - 1.10. Невозможно создать уровень больше выше максимального уровня умения.
 - 1.11. Невозможно обновить уровень так, чтобы он стал выше максимального уровня умения.
 - 1.12. Невозможно создать уровень умения, когда у этого умения такой уровень уже существует.

1.13. Невозможно обновить один уровень умения так, чтобы у умения оказалось два одинаковых уровня.

1.14. В случае изменения максимального уровня умения те его уровни, которые окажутся меньше нового максимального значения, будут удалены.

2. Тестирование реализации интерфейса IDAO для сущности Task (Задача)

2.1. Задачу возможно создать, а потом извлечь.

2.2. Задачу возможно создать, а потом изменить.

2.3. Задачу возможно создать, а потом удалить.

2.4. Невозможно создать требование какой-то задачи, которое имело бы такое же умение, как уже существующая задача.

2.5. Невозможно обновить требование какой-то задачи, чтобы оно имело такое же умение, как и другое требование этой же задачи.

3. Тестирование реализации интерфейса IDAO для сущности Profile (Профиль)

3.1. Возможно создать несколько профилей компаний, а потом извлечь их одной страницей.

3.2. Возможно создать несколько профилей компаний, а потом извлечь каждый из них по уникальному идентификатору.

3.3. Возможно создать несколько профилей компаний, а потом извлечь их всех.

3.4. Возможно создать профиль компании, а потом удалить его.

3.5. Возможно создать профиль компании, а потом изменить его.

3.6. Невозможно извлечь неправильное количество профилей компаний.

3.7. Возможно создать профиль компании, а потом создать его компетенцию.

3.8. Возможно создать профиль компании с компетенцией, а потом удалить эту компетенцию.

3.9. Возможно создать профиль компании с компетенцией, а потом создать подтверждение этой компетенции.

3.10. Возможно создать профиль компании с компетенцией и подтверждением, а потом удалить это подтверждение.

3.11. Возможно создать профиль компании с компетенцией и подтверждением, а потом изменить это подтверждение.

3.12. Невозможно создать компетенцию профиля компании так, чтобы у этой компетенции было такое же умение, как и у другой компетенции этой самой компании.

3.13. Невозможно обновить компетенцию профиля компании так, чтобы у этой компетенции было такое же умение, как и у другой компетенции этой самой компании.

4.3 Тестирование алгоритмов поиска и сравнения

Алгоритмы поиска и сравнения были реализованы каждый через свой интерфейс. Для каждой такой реализации единственный её метод является необходимым протестировать – совпадает ли результат в конкретном случае с ожидаемым.

1. Выделение подпрофиля. Были созданы три умения. По первому и второму созданы компетенции тестового профиля, второе и третье добавлены ко множеству, по которому строился подпрофиль. Таким образом, у результирующего профиля не должно быть компетенции с первым умением, компетенция со вторым должна остаться такой же, но должна появиться компетенция с третьим умением и нулевым уровнем владения.

2. Сравнение профиля с задачей. Были проанализированы три случая, в двоих из которых профиль должен справиться с задачей, в другом – нет.

3. Сравнение профилей. Были проанализированы три случая – когда результат должен быть 1 (равны), 0 (различны) и в интервале от 0 до 1 (имеют сходство).

4. Поиск с использованием компетенций. Был создан предикат из компетенций и множество профилей с различной степенью схожести с этим множеством. Было проанализировано, что по результату выполнения поиска будут найдены только необходимые профили.

5. Расположение профилей по схожести со множеством компетенций. Аналогично с поиском, но было проверено, чтобы результат выполнения был упорядочен верно.

6. Поиск всех задач, которые может выполнить профиль. Было создано три задачи, две из которых профиль мог выполнить, а одну – нет. Был произведён поиск, после чего результат был проверен.

4.4 Другие интеграционные тесты

Полнотекстовый поиск применяется через интерфейс IDAO, то есть к трём сущностям – Skill (Умение), Profile (Профиль) и Task (Задача). Для каждой сущности из данных было проведено несколько тестов на вхождение заданного слова или фразы в название объекта, а результаты сопоставлены с ожидаемыми.

Для компонента заявок было описано несколько случаев, в которых создавалась заявка, а потом либо отменялась, либо принималась.

В случае с тестированием веб-приложения были взяты некоторые отдельные страницы, протестирована корректная работа на этих страницах JavaScript и AJAX, правильность передачи и получения информации через формы, а также работа генерируемых ссылок. В той частности, были протестированы:

1. Работа навигационной полосы в левом верхнем углу экрана.
2. Работа ссылок при перелистывании страниц.
3. Работа полнотекстового поиска.
4. Работа формы создания профиля компания.

4.5 Ручное тестирование на примере профиля компании “Rehabot”

Были взяты данные одной из компаний-резидентов технопарка “Rehabot”.

- Название компании на английском языке: Rehabot
- Название компании на русском языке: Рехабот
- Вебсайт: <http://rehabot.me/>
- Адрес компании на английском языке: Birzhevaya line, 14, Saint Petersburg, Russia
- Адрес компании на русском языке: Россия, гор. Санкт-Петербург, Биржевая линия

14.

- Адрес электронной почты: support@rehabot.me
- Описание компании на английском языке: In our work, we combine professionals from different industries: engineers, programmers, designers, doctors. One of the developments Reebot — mechanical glove for the recovery of fine motor skills of hands. The device is easy to use both in the clinic and at home without direct supervision by a specialist. The method is based on mechanotherapy. The glove provides the ability of passive (early recovery), active dynamic exercises, exercises with weights, resistance.

- Описание компании на русском языке: В нашей работе мы объединяем профессионалов из разных сфер деятельности: инженеров, программистов, дизайнеров и врачей. Одна из разработок – Reebot – механическая перчатка для лечения моторики руки. Устройство легко в использовании как в клинике, так и в домашних условиях без непосредственного наблюдения специалистом. Данный метод основан на механотерапии. Перчатка предоставляет возможность пассивного (раннего) восстановления, активных динамических упражнений, упражнений с весами, сопротивления.

Данные компании были внесены в систему. Часть страницы компании, содержащая эту информацию о ней, показана на изображении 14.



Изображение 14 – Полученная в ходе ручного тестирования часть страницы компании Rehabot , содержащая информацию о ней

4.6 Результаты выполнения тестирования

В ходе выполнения тестирования были протестированы основные аспекты работы приложения. Тестирование было произведено на различных уровнях от управления данным вплоть до взаимодействия с GUI. Таким образом, если конкретная сборка приложения полностью прошла тестирование, её можно считать работоспособной.

Заключение

В ходе выполнения разработки системы была проанализирована предметная область, научная основа и аналоги системы. На основании анализа были получены функциональные требования, определены решаемые системой задачи, основные сценарии использования системы. Полученные данные были использованы для проектирования системы, в ходе которого были получены программная и системная архитектуры системы, определены используемые при реализации технологии, раскрыты некоторые сценарии использования, построены и предоставлены соответствующие диаграммы. По результатам проектирования приложение было реализовано и протестировано. Таким образом, в ходе выполнения данной работы были пройдены все этапы разработки ПО и получена работоспособная система, удовлетворяющая поставленным требованиям и способная решать поставленные задачи.

Реализованная система была успешно запущена на сервере Tomcat 8 в Технопарке Университета ИТМО и доступна по локальной сети.

По результатам выполнения работы была сделана публикация на конференции FRUCT, после чего прототип системы продемонстрирован на демо секции конференции.

Список научных источников и упоминаемых материалов

- 1 PAKS: A Competency based model for an Academic Institutions [Текст] / Prof. Pooja Tripathi; Dr. Jayanthi Ranjan; Dr Tarun Pandeya . - [Б. м.] International Journal of Innovation, Management and Technology, Vol. 1, No. 2 (2010) . - ISSN 2010-0248
- 2 Emerging competency methods for the future [Текст] / Timothy R. Athey; Michael S. Orth . – [Б. м.] Human Resource Management, Vol. 38, No. 3 (1999) . – ISSN 1748-8583
- 3 Competencies 1.0 (Measurable Characteristics), Recommendation [Текст] / HR – XML Consortium (2001-Oct-16)
- 4 Ontology-based Approach to Competence Profile Management [Текст] / Vladimir Tarasov; School of Engineering at Jönköping University, Sweden . – [Б. м.] Journal of Universal Computer Science; vol. 18, no. 20 (2012) . - ISSN 0948-6968
- 5 The effect of individual, network, and collaborative competencies on the supply chain management system [Текст] / Jane Barnes; Ying Liao; School of Business, Meredith College, Raleigh, NC 27607, USA . – [Б. м.] International Journal of Production Economics, Volume 140, Issue 2 (2012) . - ISSN: 0925-5273
- 6 Organizational Competencies: Clarifying the Construct [Текст] / William B. Edgar; Chris A. Lockwood; University of Arizona; Northern Arizona University . - [Б. м.] The Journal of Business Inquiry . - ISSN 2155-4072
- 7 Ontology-Based Competency Management for Corporate E-Learning [Текст] / Fotis Fraganidis; Greogoris Mentzas; University of Athens, Greece / Competencies in Organizational E-Learning: Concepts and Tools, 2007
- 8 A competence management system for universities [Текст] / Jürgen Dorn; Markus Pichlmair; Vienna University of Technology, Austria; Institute of Software Technology and Interactive Systems, Austria . – (AIS Electronic Library, ECIS 2007)
- 9 An Ontology and a Software Framework for Competency Modeling and Management [Текст] / Paquette, G . – [Б. м.] Educational Technology & Society, Vol. 10 No. 3 (2007) . - ISSN 1436-4522
- 10 Система SOBOLEO . – URL: <http://knowledge-maturing.com/services-and-tools/tools/people-tagging> (дата обращения: 14.05.2016)

- 11 Ontologies used for Competence Management [Текст] / V. Iordan; A. Cicortas . – [Б. м.] Acta Polytechnica Hungarica, vol. 5 no. 2 (2008) . - ISSN 1785-8860
- 12 An Ontology-centered Approach for Designing an Interactive Competence Management System for IT Companies [Текст] / C. Niculescu; S. Trausan-Matu . - [Б. м.] Informatica Economică vol. 13 no. 4 (2009) . - ISSN 1453-1305
- 13 Preliminaries for Dynamic Competence Management System building [Текст] / P. Rózewski; B. Małachowski; J. Jankowski . – [Б. м.] Proceedings of the 2013 Federated Conference on Computer Science and Information Systems (2013)
- 14 An Ontology-based Framework for Modeling User Behavior - A Case Study in Knowledge Management Systems [Текст] / L. Razmerita . – [Б. м.] Man and Cybernetics, Part A: IEEE Transactions on Systems and Humans vol. 41 no. 4, (2011)
- 15 Extending Moodle Functionalities with Ontology-based Competency Management [Текст] / K. Rezgui; H. Mhiri; K. Ghédira . – [Б. м.] Procedia Computer Science vol. 35 (2014) . - ISSN 1877-0509
- 16 An Ontology-Based Approach to Competency Modeling and Management in Learning Networks [Текст] / K. Rezgui; H. Mhiri; K. Ghédira . - Agent and Multi-Agent Systems: Technologies and Applications (2014)

Приложения

Приложение А – диаграмма программной архитектуры

