МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СИСТЕМЫ ФОРМИРОВАНИЯ МАТРИЦЫ КОМПЕТЕНЦИЙ ПЕРСОНАЛА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Борзовой Александры Евгеньевны

Научный руководитель
зав.каф.,к. ф.-м. н., доцент _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время существует такой эффективный инструмент управления рисками как матрица компетенций. Он представляет собой набор моделей компетенций, для сотрудников компании, в которой отображается набор навыков, которыми должны владеть сотрудники определённых должностей, а также уровень владения этими навыками отдельных сотрудников.

Актуальность данной работы состоит в том, что матрица компетенций играет важную роль в организации работы компании. Основываясь на собранной информации о навыках сотрудников, она позволяет управлять многими процессами в работе с персоналом: подбор, адаптация, аттестация, обучение и развитие.

Объектом исследования является система формирования матрицы компетенций персонала.

Предметом исследования являются инструменты и способы реализации системы формирования матрицы компетенций персонала.

Цель бакалаврской работы — разработка системы формирования матрицы компетенций персонала средствами языка программирования Java.

Поставленная цель определила **следующие задачи**:

- изучить учебную и научную литературу по теме: матрица и модели компетенций;
- изучить существующие варианты формирования матрицы компетенций;
- применить в разработке системы возможности языка программирования Java;
- разработать веб-приложение для работы с матрицей компетенций.

Методологические основы разработки системы формирования матрицы компетенций персонала представлены в работе Олянич Д.В.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и двух приложений. Общий объём работы — 55 страниц, из них 42 страницы — основное содержание, включая 28 рисунков, цифровой носитель в качестве приложения, список использованных источников информации — 20 наименований.

1 Анализ предметной области

В данном разделе рассматриваются теоретические основы формирования матрицы компетенций, а также описываются используемые в разработке технологии.

1.1 Определения

Модель компетенций — это набор навыков сотрудников с указанием уровня владения ими, объединённых по принадлежности к конкретной должности. [1]

Матрица компетенций персонала — это набор моделей компетенций, в которых представлен набор навыков для конкретных должностей, а также уровень владения этими навыками различных сотрудников.

1.2 Применение

Матрица компетенций является чрезвычайно полезным инструментом. Пользуясь ей, можно:

1. Отслеживать уровень компетентности сотрудников в команде, отделе или бизнес-направлении;
2. Эффективно распределять сотрудников на проекты;
3. Планировать развитие сотрудников (составлять планы обучения, проводить аттестации и.т д.);

1.3 Оценка компетенций

Для правильной оценки компетенций необходимо определить количество уровней развития навыков, желательно не более 5-6 и подробно описать перечень знаний, характерный для каждого уровня. Между уровнями должна быть чёткая качественная разница, а то что свойственно для низкого уровня по умолчанию подразумевается для более высоких.

Оценивать компетенции можно по любой шкале, это могут быть цифры в определённом диапазоне, знаки или буквы.

1.4 Формирование

Формирование матрицы компетенций подразумевает составление для каждой должности модели компетенций. Для этого необходимо определить перечень навыков, которыми должен обладать сотрудник на определённой должно-

сти и выделить самые значимые (те, которые применяются чаще остальных). Затем для заполнение матрицы необходимо провести оценку каждого сотрудника. [2]

Зачастую формирование матрицы компетенций в компаниях происходит путём анкетирования сотрудников с использованием различных инструментов, например Microsoft Forms или Google Forms.

В свою очередь веб-приложение, совмещающее в себе функционал анкетирования сотрудников, сохранения и анализа результатов, является более удобным в применении.

1.5 Используемые технологии

Для достижения поставленной цели в процессе разработки системы были использованы различные технологии.

1.5.1 СУБД

Так как формирование матрицы компетенций предполагает хранение информации об уровне освоения работниками различных навыков, в системе необходимо использование БД, а для её создания и управления данными в ней нужна СУБД.

При разработке системы использовалась СУБД PostgreSQL.

1.5.2 Backend

Backend — внутренняя часть приложения, которая находится на сервере и скрыта от пользователей.

В разработанной системе Backend был написан с помощью следующих технологий: Java 11, Maven, фреймворк Spring (Boot, Data, MVC, Security), библиотека Lombok.

1.5.3 Frontend

Frontend — это клиентская часть приложения (интерфейс, с которым взаимодействует пользователь).

В разработанной системе Frontend был написан с помощью следующих технологий: HTML5, шаблонизатор Thymeleaf, шаблоны и стили Bootstrap.

2 Программная реализация системы формирования матрицы компетенций персонала

Данный раздел содержит в себе описание программной реализации системы формирования матрицы компетенций персонала средствами языка программирования Java с использованием перечисленных в первом разделе технологий.

2.1 Общее описание системы

Система представляет собой сайт, на котором сотрудники компании могут пройти тестирование по различным специальностям. Результаты тестирования сохраняются в БД и представляют собой сформированную матрицу компетенций, которая содержит модули, количество которых равно количеству тестов, представленных в системе. Также реализована возможность сортировки сотрудников по уровням их владения различными навыками, а также возможность просмотра результата всех тестов, пройденных сотрудником. Эти действия можно осуществить, если войти в систему под специальной учётной записью администратора.

Приложение имеет клиент-сервисную архитектуру и реализует архитектурный паттерн MVC (Model View Controller) — данные приложения и управляющей логики разделены на три отдельных компонента: модель, представление и контроллер. [3]

Реализовано тестирование по специальностям: Java Developer, Kotlin Developer и Data Integration. Список вопросов к тестам небольшой и может быть дополнен при необходимости.

2.2 Приготовления

2.2.1 IDE

Для разработки была использована **IntelliJ IDEA** - это функциональная и эргономичная IDE для профессиональной разработки на Java, Scala, Kotlin и других ЯП.

2.2.2 Создание проекта

Так как в основе приложения лежит Spring Boot для создания основы проекта и подключения необходимых библиотек можно воспользоваться инструментом Spring Initializr.

При таком способе создания проекта, автоматически будет создан класс разрабатываемого приложения, не содержащий ничего кроме метода main для запуска и аннотации @SpringBootApplication.

2.2.3 Подключение зависимостей

Частично нужные зависимости можно подключить в Spring Initializr.

Так как в проекте используется Maven, остальные зависимости нужно подключать в файле pom.xml: [4]

- **Hibernate Validator** - позволяет валидировать поля объектов с помощью аннотаций;
- **Json Simple** - позволяет работать с json объектами;
- **Spring Security Core** - многофункциональная библиотека, но в разработанной системе использован только BCryptPasswordEncoder - объект, который позволяет зашифровывать пароли пользователей для хранения в БД; [5]
- **Hibernate Types 52** - позволяет создавать модели объектов БД с json полями.

2.3 Подготовка БД

При разработке системы был использован подход Database-First, таким образом перед написанием самого приложения была создана БД, состоящая из 5 таблиц:

- **users** - данные сотрудников (имя, фамилия, email, пароль, должность и результаты прохождения тестов);
- **roles** - роли (все доступные должности и специальная роль - администратор);
- **tests_result** - результаты прохождения тестов (один JSON-объект на каждое направление тестирования);
- **questions** - таблица с вопросами для тестов (название вопроса, его текст, тип и 4 варианта ответа);
- **q_types** - таблица с типами вопросов (каждый тип - одно направление тестирования).

Таблицы questions, q_types и roles заполнены заранее и их содержимое никак не меняется при работе приложения.

2.4 Создание приложения

2.4.1 Модели

В самом начале нужно создать классы-сущности, описывающие объекты таблиц БД. Каждой таблице должен соответствовать один класс-сущность.

Для их создания также очень полезен инструмент Database, нажав правой кнопкой по любой таблице и выбрав пункт "Entity from DB" откроется окно добавления сущностей, в котором можно настроить поля генерируемого объекта.

При нажатии кнопки "Create" автоматически будет создан класс-сущность, поля которого будут замаплены на поля выбранной таблицы. Но для того чтобы класс работал правильно и код был читаемым нужно предпринять следующие шаги:

- Удалить геттеры и сеттеры, добавить на класс аннотацию @Data;
- Добавить аннотацию @Type(type = "org.hibernate.type.TextType") на все поля с аннотацией @Lob;
- Добавить валидацию к полям.

Классы-сущности можно создать и без использования инструмента Database, в таком случае нужно самостоятельно написать эти классы по тем же правилам.

Также были созданы вспомогательные классы: Data_Integration_Result, Java_Result и Kotlin_Result, WrapperForm. И два класса-исключения: EmailOrPasswordIncorrectException и UserAlreadyExistException

2.4.2 Репозитории

В Spring Data JPA взаимодействие с БД происходит с помощью репозиториев. Эти интерфейсы должны расширять интерфейс JpaRepository.

Создать их автоматически нельзя, но это достаточно просто сделать самому. Имя интерфейса должно содержать название класса-сущности для которого он предназначен и слово "Repository". В угловых скобках нужно указать тот же класс и тип поля, которое является его первичным ключом.

Интерфейс не содержащий никаких других определённых методов предоставляет базовые методы взаимодействия с сущностью:

- **findById(. . .)** - поиск по id, возвращает объект класса-сущности;
- **save(. . .)** - сохранение сущности в БД;
- **delete(. . .)** - удаление сущности из БД;
- **и другие.**

Для того чтобы осуществлять с сущностями взаимодействия, не являющиеся базовыми, например, выполнять поиск не по id, а по значению одного из полей или же осуществлять сложные SELECT запросы, необходимо определять методы для таких взаимодействий в репозитории. [6]

2.4.3 Сервисы

Сервисы - это классы содержащие в себе большую часть логики приложения. У каждого класса-сущности есть один класс-сервис для него. Название сервиса должно состоять из имени сущности и слова "Service". Также все их нужно помечать аннотацией @Service и @Transactional для корректной работы.

В каждый класс-сервис нужно подставить бин соответствующего репозитория с помощью аннотации @Autowired и реализовать все его нужные методы.

Сервисы для сущностей Role, Q_Type и TestsResult не содержат ничего кроме бина репозитория и реализации его методов.

В классах UserService и QuestionService реализовано несколько методов для работы с сущностями User и Question.

2.4.4 Контроллеры

REST-контроллер — класс, который обрабатывает html запросы, каждый метод класса определяет действия программы, а также возвращаемый html-шаблон при переходе пользователя по определённому URL. [7]

Для правильной работы на каждом контроллере должна стоять аннотация @Controller. Метод запроса и URL указываются в аннотациях над методами класса.

Программа содержит 6 контроллеров:

- **MainController** — переход на главную страницу и личный кабинет пользователя;
- **SecurityController** — процесс регистрации и авторизации пользователя;
- **JavaTestController** — прохождение теста по компетенциям Java Developer;
- **KotlinTestController** — прохождение теста по компетенциям Kotlin Developer;
- **DataIntegrationController** — прохождение теста по компетенциям Data Integration.
- **AdminController** — просмотр модулей компетенций, сортировка сотрудников по уровням владения определёнными навыками, поиск результатов тестирования сотрудника по его почте.

2.5 Результат

Воспользоваться разработанной системой можно в любом браузере, запустив приложение и перейдя по URL: http://localhost:8080. Откроется страница, на которой можно авторизоваться или зарегистрироваться.

2.5.1 Интерфейс сотрудника

После успешной регистрации или авторизации пользователь переходит на страницу личного кабинета, в котором может выбрать раздел с нужным ему тестом.

При переходе на страницу любого теста, по нему отображается краткая информация.

При нажатии на кнопку "Пройти тест" пользователь переходит к странице с вопросами по выбранной специальности.

При нажатии кнопки завершить, пользователь возвращается в личный кабинет и больше не может проходить данный тест.

2.5.2 Интерфейс администратора

Если пользователь войдёт в систему, используя учётную запись администратора, он получит доступ к функционалу для работы с данными, содержащимися в матрице компетенций. Сразу после входа откроется страница с небольшой навигационной панелью.

При нажатии на ссылку "Модули компетенций" пользователь попадает на страницу выбора направления.

При нажатии кнопки "Поиск" вверху страницы появится возможность выбора навыков для сортировки, а внизу отобразится список всех пользователей, прошедших соответствующий тест. Пользователи будут выведены в случайном порядке.

При выборе навыков, пользователи будут отсортированы, чем больше уровень владения пользователем соответствующих навыков, тем выше он будет в итоговой таблице.

При нажатии кнопки "В меню" пользователь в любой момент может вернуться к главной панели навигации.

Ссылка панели навигации с надписью "Поиск по email" ведёт на страницу, где можно по email узнать, какие тесты пользователь проходил.

ЗАКЛЮЧЕНИЕ

При написании настоящей работы поставленная цель была достигнута.

Разработана система формирования матрицы компетенций персонала в форме веб-приложения. Реализован следующий функционал:

- регистрация и авторизация пользователя;
- прохождение тестов по трём различным направлениям: Java Developer, Kotlin Developer, Data Integration;
- просмотр модулей компетенций;
- сортировка содержимого модулей;
- поиск информации о сотруднике в матрице компетенций по email.

При разработке использовался ЯП Java. В качестве СУБД использовалась PostgreSQL. Веб-приложение написано с использованием фреймворка Spring, точнее его компонентов: Boot, Data, MVC, а также шаблонизатора Thymeleaf. Верстка веб страниц осуществлялась с помощью HTML5, JS, CSS, а также шаблонов и стилей Bootstrap.

Поставленные задачи были полностью выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Матрица компетенции [Электронный ресурс]. — URL: https://cyberleninka.ru/article/n/matritsa-kompetentsii/viewer (Дата обращения 21.05.2022). Загл. с экр. Яз. рус.

2 *Олянич, Д.* Теория организации: учебник / Д. Олянич. — Ростов н/Д: Феникс, 2008.

3 *Хеклер, М.* Spring Boot по-быстрому / М. Хеклер. — Санкт-Петербург: Питер, 2022.

4 *Штукатуров, С.* Краткое знакомство с Maven [Электронный ресурс] / С. Штукатуров. — URL: https://tproger.ru/articles/maven-short-intro/ (Дата обращения 04.07.2021). Загл. с экр. Яз. рус.

5 Spring Boot Security - Password Encoding Using BCrypt [Электронный ресурс]. — URL: https://www.javainuse.com/spring/boot_security_jdbc_authentication_bcrypt (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

6 Spring Data JPA: Волшебные методы [Электронный ресурс]. — URL: https://alexkosarev.name/2017/02/09/spring-data-jpa-magic-methods/ (Дата обращения 23.05.2022). Загл. с экр. Яз. рус.

7 Построение REST сервисов с помощью Spring [Электронный ресурс]. — URL: http://spring-projects.ru/guides/tutorials-bookmarks/ (Дата обращения 26.05.2022). Загл. с экр. Яз. рус.