ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники

Образовательная программа: Системное и прикладное программное обеспечение

Направление подготовки (специальность): 09.03.04 Программная инженерия

Отчет об учебной, ознакомительной практике

Тема задания: Исследование и разработка оптимальной структуры для внедрения нового функционала: чек-листов

Обучающийся: Яковлев Григорий Алексеевич, P34111

Руководитель практики от профильной организации: Вахитов Линар Маратович, Владелец Продукта (Product Manager) ООО «Ланит-Терком»

Руководитель практики от университета: Маркина Татьяна Анатольевна, Доцент факультета программной инженерии и компьютерной техники

Санкт-Петербург

2024 г

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тема практики: Исследование и разработка оптимальной структуры для внедрения нового функционала: чек-листов.

Цель работы: Разработать и интегрировать систему чек-листов в существующую информационную систему для контроля качества работы сотрудников.

Задачи: По требованиям к новой системе чек-листов определить структуру базы данных для хранения и обработки данных чек-листов. Создать интерфейс для взаимодействия с чек-листами. Разработать и обеспечить интеграцию системы чек-листов с существующей инфраструктурой организации. Протестировать систему на предмет корректности работы функций.

Этапы выполнения:

1. Анализ требований и постановки задачи
2. Разработка схемы базы данных
3. Разработка контроллера для работы с чек-листами
4. Проверка и устранение ошибок

1 Анализ требований и постановки задачи

Когда я приступил к работе, мой руководитель предоставил мне документацию по сервису, которая включала описание бизнес-процессов приложения. Документация содержала как общее видение функционала системы чек-листов, так и конкретные технические требования к её реализации. Изучение предоставленных материалов позволило мне погрузиться в контекст задачи и понять ключевые цели проекта.

После тщательного анализа требований и постановки задачи на следующем этапе моей работы стояла задача выявления необходимых таблиц и структуры базы данных для хранения и эффективной работы с чек-листами. Мое внимание сосредоточилось на выявлении ключевых аспектов, которым должна была удовлетворять новая система: удобство использования, эффективное хранение и обработка данных.

Я начал с изучения функциональных требований к системе чек-листов, чтобы определить, какие данные необходимо хранить, как они будут использоваться, и какие отношения между ними необходимо установить.

На основе этого анализа были определены основные сущности системы, такие как:

- Чек лист
- Сотрудники группы разбора
- Ежедневные задания
- Критерии для формирования ежедневных заданий

Для каждой сущности был разработан набор атрибутов, необходимых для хранения всей релевантной информации. Например, для чек-листов были выделены атрибуты, такие как идентификатор Чек-Листа, дата изменения, вопросы чек-листа, статус, оценка по чек листу.

Далее я приступил к определению отношения между сущностями. Были установлены как один-ко-многим, так и многие-ко-многим отношения, чтобы

обеспечить максимальную гибкость и возможность многократного использования элементов в различных чек-листах.

Процесс проектирования базы данных сопровождался сопоставлением разрабатываемой структуры с функциональными требованиями, что позволило мне гарантировать, что разработанная система будет полностью соответствовать бизнес-требованиям.

Единственная проблема, с которой я столкнулся – это проблема точного перевода функциональных требований в техническую спецификацию базы данных. Не всегда было очевидно, какие структуры данных лучше всего подойдут для конкретных задач, особенно когда речь шла о нестандартных или сложных функциях системы.

Результатом моей работы стала документация по структуре базы данных, которая включала описание таблиц, их атрибутов, отношений между ними. Эту документацию я выложил на внутренней wiki-странице компании. Публикация документации в централизованном репозитории также подразумевала возможность её постоянного обновления и дополнения по мере развития и адаптации системы под новые требования и задачи.

2 Разработка схемы базы данных

На этапе разработки схемы базы данных моя задача заключалась в том, чтобы превратить теоретическую модель, разработанную на предыдущих этапах, в физическую модель. Этот процесс включал разработку таблиц, определение необходимых индексов и внешних ключей, а также анализ возможности кэширования данных для оптимизации производительности системы.

Процесс переноса модели базы данных из теоретической документации в практическую реализацию прошел удивительно гладко и без серьезных проблем. Это стало возможным благодаря тщательному предварительному планированию и четкой структуре документации, которую я разработал на предыдущих этапах. Каждая таблица, её атрибуты, типы данных и связи были

подробно описаны, что облегчило их реализацию в системе управления базами данных.

После успешного создания структуры таблиц я перешел к следующему критически важному этапу – определению индексов и стратегии кэширования. Индексация была ключевым аспектом для оптимизации производительности системы, поэтому я выявил запросы к базе данных, которые будут наиболее частыми и добавил индексы на поля, которые чаще всего используются для поиска данных. Одним из таких полей стало поле «Дата проверки сотрудником», потому что будет применяться ежедневный подсчет количества проверок.

Затем я сосредоточился на кэшировании данных, чтобы дополнительно улучшить производительность системы. Особое внимание было уделено таблицам справочникам, которые содержат неизменяемую или редко изменяемую информацию, такой таблицей стала например «Статусы чек-листов». Кэш снизил нагрузку на базу данных и ускорил доступ к данным.

В целом, этот этап разработки показал важность предварительной работы над документацией для обеспечения эффективной реализации теоретической модели в практическую базу данных. В итоге физическая модель данных была успешно реализована без особых проблем. Код-пример сущности представлен в Приложении А.

3 Разработка контроллера для работы с чек-листами

На этапе разработки контроллера для работы с чек-листами была поставлена задача создать компонент системы, который бы обеспечивал сохранение чек-листа после заполнения сотрудниками.

В первую очередь был спроектирован интерфейс контроллера, содержащий метод сохранения чек-листа после его заполнения сотрудниками.

Сам контроллер необходимо разработать с использованием принципов REST API для обеспечения простоты интеграции с клиентскими приложениями.

Метод сохранения был реализован таким образом, чтобы обработать данные, введенные сотрудником, включая проверку на валидность и полноту заполнения. При разработке метода была учтена, возможность сохранения частично заполненных чек-листов для последующего их заполнения или изменения.

Далее следовало тестирование контроллера, которое включало в себя разработку юнит-тестов с использованием JUnit и Mockito для проверки логики обработки запросов без необходимости взаимодействия с реальной базой данных.

В результате, разработанный контроллер обеспечил надежное и эффективное взаимодействие между пользовательским интерфейсом и серверной частью. Код сервисного класса представлен в Приложении Б.

4 Проверка и устранение ошибок

Теперь, когда код написан, необходимо проверить, что разработанная система функционирует корректно и эффективно, соответствует всем техническим требованиям и бизнес-логике проекта.

В ходе модульного тестирования значительных ошибок выявлено не было. Последующее интеграционное тестирование подтвердило корректное взаимодействие всех компонентов системы.

Однако, важной частью проверки системы являлась проверка на соответствие всем заранее согласованным требованиям. Для этого я внимательно изучил постановку задачи и техническую документацию, чтобы убедиться, что реализованный функционал соответствует всем указанным требованиям.

Я сверил каждый аспект системы с определенными требованиями, проверил, что все функции работают корректно и в полной мере удовлетворяют запросы заказчика.

ЗАКЛЮЧЕНИЕ

В рамках учебной практики мною была успешно разработана оптимальная структура для нового функционала: чек-листов. Задача включала анализ требований, разработку физической модели данных, создание контроллера для работы с чек-листами и тестирование системы. Так же в ходе работы было уделено внимание на повышение производительности системы, чего удалось достичь путем внедрения индексации и стратегий кэширования.

Разработка дала мне ценный опыт в областях проектирования программного обеспечения, разработки и оптимизации работы с базами данных. Результаты проверки подтвердили, что система чек-листов соответствует всем техническим и бизнес-требованиям.

СПИСОК ИСПОЛЬЗУЮЕМЫХ ИСТОЧНИКОВ

1. Документация для разработки Spring Boot приложений [Электронный ресурс]. – Режим доступа. – https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/, свободный (дата обращения 06.03.2024)
2. Документация PostgreSQL [Электронный ресурс]. – Режим доступа. – https://www.postgresql.org/docs/, свободный (дата обращения 06.03.2024)
3. Требования по Чек-Листам [Электронный ресурс]. – Режим доступа. – https://wiki.mos.social/pages/viewpage.action?pageId=568801086, закрытый (дата обращения 06.03.2024)

ПРИЛОЖЕНИЕ А

Код-пример сущности Чек-Листа

```
@Entity
@Getter
@Setter
@ToString
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "check_list")
public class CheckList {

    @Id
    @Column(name = "id", nullable = false)
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "CheckListIdSeq")
    @SequenceGenerator(sequenceName = "check_list_id_seq", name = "CheckListIdSeq", allocationSize = 1)
    private Long id;

//...

    @NonNull
    @ToString.Exclude
    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "check_list_status_id", nullable = false)
    private CheckListStatus checkListStatus;

    @ToString.Exclude
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "responsible_id")
    private Responsible responsible;

    @Column(name = "date_change")
    private LocalDateTime dateChange;

//...

    @Column(name = "checklist_question_1")
    private Boolean checklistQuestion1;

    @Column(name = "checklist_question_1_1")
    private Boolean checklistQuestion1q1;

//...
}
```

ПРИЛОЖЕНИЕ Б

Код сервисного класса для работы с Чек-Листами

```
@NonNull
@Override
@Transactional
public ResponseEntity<Long> saveCheckList(@NonNull Long checkListId,
                                          @NonNull SaveCheckListServiceRequestDTO requestDTO) {
    //1. Система находит запись по checkListId
    CheckList checkList = checkListService.existingAndLockById(checkListId);

    //2. Сохраняет поля, вычисляет score_sostav, grade_sostav. score_cp, grade_cp
    checkList = checkListService.saveCheckListByResponsible(checkList, requestDTO);

    //3. Система возвращает в качестве результата check_list_id и завершает сценарий
    return ResponseEntity.ok(checkList.getId());
}
```