

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)
Институт математики и информационных технологий
Кафедра алгебраических и информационных систем

КУРСОВАЯ РАБОТА
по предмету
«Проектирование информационных систем»

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ
«ИНСТРУМЕНТАЛЬНАЯ РАЗДАТОЧНАЯ КЛАДОВАЯ»

Студент 3 курса очного отделения
Группа 02371–ДБ
Федоров Никита Олегович

Руководитель:
к.ф.-м.н., доцент Рябец Л.В.

Иркутск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
Глава 1. Описание предметной области	4
1.1. Анализ предметной области	4
1.2. Требования к разрабатываемому приложению	5
Глава 2. Обзор технологий разработки	7
2.1. Spring Framework	7
2.2. Thymeleaf	7
2.3. Hibernate	7
2.4. SQL	8
Глава 3. Описание реализации приложения	9
3.1. Хранимые сущности и проектирование структуры классов	9
3.2. Разработка backend части веб-приложения	10
3.3. Разработка frontend части веб-приложения	12
3.4. Реализованная функциональность	14
ЗАКЛЮЧЕНИЕ	18
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	19
ПРИЛОЖЕНИЕ 1. UML-диаграмма классов приложения	23

ВВЕДЕНИЕ

Разработка современных веб-приложений требует надлежащего понимания принципов работы с базами данных и управления данными на стороне сервера. В данной курсовой работе представлена разработка веб-приложения «Инструментальная раздаточная кладовая» с использованием фреймворка Spring и библиотеки Hibernate. [20; 25]

В современном мире, где всё больше компаний автоматизируют производство, необходимо иметь эффективные инструменты для управления запасами инструментов и их мониторинга. Веб-приложение «Инструментальная раздаточная кладовая» решает эту проблему, позволяя управлять каталогом инструментов, отслеживать их запасы и быстро передавать информацию работникам ИРК. [8; 18; 22]

Цель данной работы - продемонстрировать возможности использования Spring и Hibernate для разработки полнофункционального веб-приложения, которое позволяет управлять каталогом инструментов, отслеживать их запасы и быстро передавать необходимую информацию. [21; 23; 24; 30]

В ходе выполнения работы были применены принципы объектно-ориентированного программирования, разработки приложений с использованием фреймворка Spring и работы с базами данных с помощью Hibernate.

Глава 1. Описание предметной области

1.1. Анализ предметной области

Одним из популярных приложений для 3D (2D) моделирования с разработкой УП для станков с ЧПУ является программное обеспечение MasterCAM. Эта программа предоставляет возможность разрабатывать модели для станков с ЧПУ. [2; 5; 6]

Во-вторых, существуют приложения, специально разработанные для станков с ЧПУ и обширного ряда ЧПУ-контроллеров. Эти приложения предназначены для бесконтактной наладки инструмента и других задач. [1; 12; 13]

Для наладки инструмента на станках с ЧПУ также используются карты наладки станка с ЧПУ. Это специальные чертежи, в которых фиксируются все изменения в производственном процессе на определенном станке. [25—29]

Кроме того, существуют и другие приложения, которые могут облегчить операции на станках с ЧПУ и управление ими. [3; 10; 11; 17]

В рамках работы мы затронем обязанности работника инструментальной раздаточной кладовой, а именно наладку им инструмента согласно поступаемым запросам и отслеживание состояния содержимого кладовой. Приложение должно быстро и понятно доставлять информацию работникам ИРК и иметь возможность частично заполнять информацию для отчётов, связанных с учётом или заказом инструмента. Помимо этого, доступ к приложению должен быть у технологов на производстве для составления технологических карт внутри приложения их отправки работникам ИРК или внесения внутренних изменений.

1.2. Требования к разрабатываемому приложению

Во время разработки были определены следующие задачи:

Функциональные требования:

- 1) Разработка backend части приложения (логики), включая:
 - Обеспечение корректной работы бизнес-логики приложения;
 - Реализация логики взаимодействия с базой данных;
 - Реализация системы авторизации и аутентификации пользователей;
 - Реализация системы различных ролей для пользователей;
 - Реализация функционала заполнения данных для отчётов на дальнейшую печать.
- 2) Разработка frontend части приложения (клиентская часть), включая:
 - Обеспечение удобного и интуитивно понятного пользовательского интерфейса;
 - Реализация возможности просмотра и редактирования данных в приложении;
 - Реализация возможности просмотра отчётов и печати их в нужном формате.

Нефункциональные требования:

- 1) Надёжность: приложение должно обеспечивать надёжную работу в течение длительного времени без сбоев и с минимальным количеством ошибок.
- 2) Безопасность: приложение должно обеспечивать безопасность данных и защиту от несанкционированного доступа.
- 3) Производительность: приложение должно обеспечивать быструю и эффективную работу, даже при большом объёме данных.

- 4) Поддержка совместимости: приложение должно поддерживать совместимость с различными операционными системами и браузерами.
- 5) Пользовательский опыт: приложение должно обеспечивать удобный и интуитивно понятный пользовательский интерфейс, который будет удовлетворять потребностям конечных пользователей.

Глава 2. Обзор технологий разработки

2.1. Spring Framework

Spring — это один из самых популярных Java-фреймворков, который предоставляет обширный набор инструментов для разработки Java-приложений. Spring обеспечивает инверсию управления (IoC) и управление жизненным циклом бинов, что упрощает разработку и обеспечивает легкую интеграцию с другими технологиями. Также Spring содержит модули для работы с веб-приложениями (Spring MVC) и доступа к данным (Spring Data) [32].

2.2. Thymeleaf

Thymeleaf — это мощный шаблонизатор, который используется для создания веб-интерфейсов в Java-приложениях. Он позволяет разработчикам создавать динамические HTML-страницы с использованием шаблонов и встроенных выражений. Thymeleaf хорошо интегрируется с фреймворком Spring, что обеспечивает удобство разработки веб-приложений [16].

2.3. Hibernate

Hibernate — это ORM (Object-Relational Mapping) фреймворк для работы с базами данных в Java-приложениях. Hibernate упрощает взаимодействие с базами данных путем предоставления объектно-ориентированного интерфейса для работы с таблицами и данными. Он позволяет разработчикам работать с объектами Java, не задумываясь о деталях SQL запросов, что упрощает процесс разработки и поддержки приложений [4].

2.4. SQL

SQL (Structured Query Language) - это язык структурированных запросов, который используется для работы с реляционными базами данных. С помощью SQL разработчики могут создавать, изменять и управлять данными в базах данных. SQL является стандартным языком для работы с реляционными базами данных и позволяет выполнять разнообразные операции, такие как выборка данных, добавление записей, обновление и удаление данных [14; 19; 31].

Глава 3. Описание реализации приложения

Приложение состоит из серверной, клиентской части и базы данных. В первой содержится логика по которой работает приложение от взаимодействия пользователя с веб страницами до изменения данных в базе данных.

3.1. Хранимые сущности и проектирование структуры классов

- 1) Instrument - сущность, хранящая информацию об инструментах;
- 2) Instrument Parts - сущность, описывающая связь между инструментами и их деталями;
- 3) Irk Worker - сущность, содержащая информацию о сотрудниках ИРК в системе;
- 4) Part - сущность, содержащая информацию о деталях инструментов;
- 5) Tech Card - сущность, описывающая технологические карты;
- 6) Techcard Instruments - сущность, устанавливающая связь между инструментами и техническими картами;
- 7) Technitian - сущность, содержащая информацию о технологах в системе;

В прил. 4 на странице 23 продемонстрирована диаграмма классов веб-приложения.

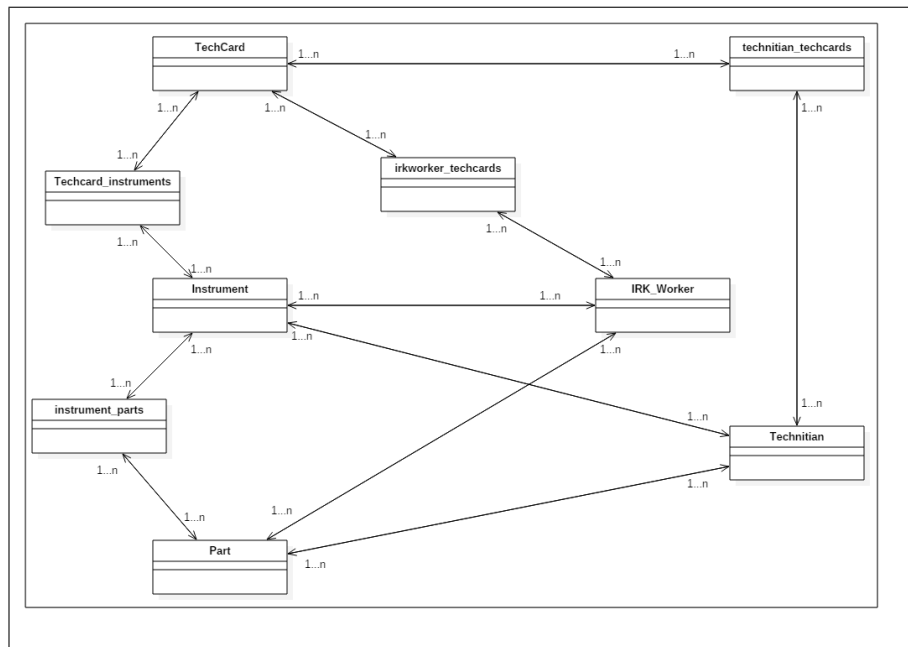


Рисунок 1. ER-диаграмма базы данных

3.2. Разработка backend части веб-приложения

Backend часть веб-приложения состоит из трёх типов функциональных классов: контроллеры, сервисы и репозитории [9].

Контроллеры отвечают за обработку HTTP-запросов, взаимодействие с клиентом и передачу данных между клиентом и бизнес-логикой приложения. В Spring контроллеры обычно аннотируются с помощью `@Controller`. Они содержат методы, которые обрабатывают запросы на определённые URL-адреса и возвращают данные клиенту. [15]

Сервисы содержат бизнес-логику приложения и выполняют операции с данными, полученными от репозитория. В Spring сервисы обычно аннотируются с помощью `@Service`. Они инкапсулируют логику работы с данными и предоставляют методы для выполнения операций над данными.

Репозитории предоставляют абстракцию для работы с базой данных. Они используются для выполнения операций CRUD над сущностями. В Spring репозитории обычно аннотируются с помощью `@Repository`. Они предоставляют методы для поиска, сохранения, обновления и удаления данных.

Поскольку реализованы сущности Part, Instrument и TechCard, то дальнейшее пояснение логики приложения будет на примере одной из этих сущностей. При попытке перехода на страницу listParts или аналогичные в контроллере с аннотацией @RequestMapping("/parts") срабатывает метод:

Листинг 1. метод listParts

```
@GetMapping
public String listParts(Model model) {
    List<Part> parts = partService.getAllParts();
    model.addAttribute("parts", parts);
    return "listParts";
}
```

После перехода перед нами появляется список из объектов внутри сущности с возможностью выполнения CRUD операций, причём при скажем удалении одного из объектов сработает цепочка методов:

Листинг 2. метод deletePart в контроллере

```
@GetMapping("/delete/{partId}")
public String deletePart(@PathVariable("partId") Long partId) {
    Part part = partService.getPartById(partId);
    partService.deletePart(part);
    return "redirect:/parts";
}
```

Листинг 3. метод deletePart в сервисе

```
public void deletePart(Part part) {
    partRepository.delete(part);
}
```

Данная цепочка методов приведёт к удалению выбранного объекта в сущности Part и всех его связей. Аналогичным образом настроены цепочки методов для CRUD операций во всех остальных сущностях.

3.3. Разработка frontend части веб-приложения

Разработка frontend части приложения (клиентская часть) включает в себя создание пользовательского интерфейса и взаимодействие с пользователем. Для этой цели использовался фреймворк Thymeleaf. Thymeleaf - это мощный шаблонизатор, который позволяет разработчикам создавать динамические HTML-страницы с использованием шаблонов и встроенных выражений. Он хорошо интегрируется с фреймворком Spring и обеспечивает удобство разработки веб-приложений.[7]

В веб-приложении реализованы по 3 представления для каждой сущности с целью поддержки следующего функционала:

- 1) Отображение всех существующих объектов для сущности с возможностью удаления отдельных объектов;
- 2) Добавление нового объекта;
- 3) Обновление параметров объекта;

Представления берут данные напрямую из базы данных используя код с атрибутами Thymeleaf аналогичный следующему:

Листинг 4. Код таблицы для представления listParts

```
<table>
    <thead>
        <tr>
            <th>ID</th>
            <th>Type</th>
            <th>Specification</th>
            <th>Quantity</th>
            <th>Reference</th>
            <th></th>
        </tr>
    </thead>
```

```

<tbody>
<th:block th:each="part : ${parts}">
<tr>
    <td th:text="${part.id}"></td>
    <td th:text="${part.type}"></td>
    <td th:text="${part.specification}"></td>
    <td th:text="${part.quantity}"></td>
    <td th:text="${part.reference}"></td>
<td>
    <a th:href="@{/parts/edit/{partId}(partId=${part.id})}">Edit</a>
    <a th:href="@{/parts/delete/{partId}(partId=${part.id})}">Delete</a>
</td>
</tr>
</th:block>
</tbody>
</table>

```

Представления отображающие существующие объекты некоторых сущностей веб-приложения представлены в конце пункта 3.4 на рис. 2 и рис. 3.

3.4. Реализованная функциональность

В рамках текущей работы реализованы следующие варианты использования разработанного приложения:

1) Создание и управление сущностями

Описание: Создание, редактирование и удаление сущностей, необходимых для реализации проекта. Это включает в себя создание моделей для каждой сущности, определение полей и связей между ними.

Основной актёр: Разработчик

Цель: Обеспечить создание и управление сущностями для реализации проекта.

Предусловия: Разработчик имеет доступ к системе и необходимым инструментам для создания объектов.

Текущий поток событий:

- Разработчик создает модель для новой сущности.
- Разработчик определяет поля и связи для модели.
- Разработчик сохраняет модель.
- Система создает сущность на основе модели.

Альтернативный поток событий:

- Разработчик редактирует существующую модель.
- Разработчик изменяет поля и связи для модели.
- Разработчик сохраняет изменения.
- Система обновляет сущность на основе измененной модели.

2) Взаимодействие с сущностями

Описание: Взаимодействие с сущностями через классы Controller, Service и Repository. Это включает в себя создание логики для создания, чтения, обновления и удаления сущностей.

Основной актёр: Система

Цель: Обеспечить логику взаимодействия с сущностями для реализации проекта.

Предусловия: Классы Controller, Service и Repository созданы и настроены.

Текущий поток событий:

- Система получает запрос на создание сущности.
- Controller обрабатывает запрос и вызывает соответствующий метод Service.
- Service обрабатывает запрос и вызывает соответствующий метод Repository.
- Repository создает сущность и возвращает результат в Service.
- Service возвращает результат в Controller.
- Controller возвращает результат в систему.

Альтернативный поток событий:

- Система получает запрос на чтение сущности.
- Controller обрабатывает запрос и вызывает соответствующий метод Service.
- Service обрабатывает запрос и вызывает соответствующий метод Repository.
- Repository читает сущность и возвращает результат в Service.
- Service возвращает результат в Controller.

- Controller возвращает результат в систему.

3) Клиентское взаимодействие с проектом

Описание: Взаимодействие с проектом через клиентские страницы. Это включает в себя создание пользовательского интерфейса для доступа к функциональности проекта.

Основной актёр: Пользователь

Цель: Обеспечить доступ к функциональности проекта через пользовательский интерфейс.

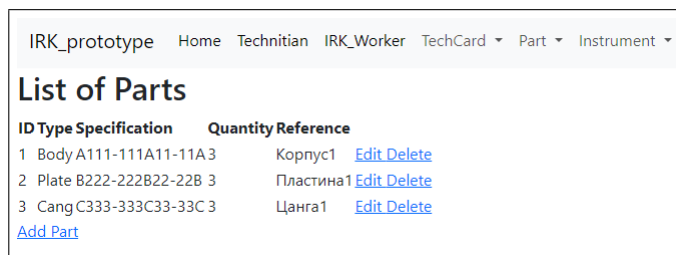
Предусловия: Клиентские страницы созданы и настроены.

Текущий поток событий:

- Пользователь открывает клиентскую страницу.
- Пользователь вводит данные и отправляет запрос.
- Система обрабатывает запрос и вызывает соответствующий метод Controller.
- Controller обрабатывает запрос и вызывает соответствующий метод Service.
- Service обрабатывает запрос и вызывает соответствующий метод Repository.
- Repository обрабатывает запрос и возвращает результат в Service.
- Service возвращает результат в Controller.
- Controller возвращает результат в систему.
- Система отображает результат на клиентской странице.

Альтернативный поток событий:

- Пользователь редактирует данные на клиентской странице.
- Пользователь отправляет запрос на обновление.
- Система обрабатывает запрос и вызывает соответствующий метод Controller.
- Controller обрабатывает запрос и вызывает соответствующий метод Service.
- Service обрабатывает запрос и вызывает соответствующий метод Repository.
- Repository обновляет сущность и возвращает результат в Service.
- Service возвращает результат в Controller.
- Controller возвращает результат в систему.



IRK_prototype Home Technician IRK_Worker TechCard ▾ Part ▾ Instrument ▾				
List of Parts				
ID	Type	Specification	Quantity	Reference
1	Body	A111-111A11-11A3		Корпус1 Edit Delete
2	Plate	B222-222B22-22B 3		Пластина1 Edit Delete
3	Cang	C333-333C33-33C 3		Цанга1 Edit Delete
Add Part				

Рисунок 2. Страница с информацией о всех деталях

IRK_prototype

All Instrument Details

Instrument Number	Parts	
123	<ul style="list-style-type: none">Type: Body, Specification: A111-111A11-11A, Quantity: 3, Reference: Корпус1Type: Plate, Specification: B222-222B22-22B, Quantity: 3, Reference: Пластина1Type: Cang, Specification: C333-333C33-33C, Quantity: 3, Reference: Цанга1	Edit Delete
12	<ul style="list-style-type: none">Type: Body, Specification: A111-111A11-11A, Quantity: 3, Reference: Корпус1Type: Plate, Specification: B222-222B22-22B, Quantity: 3, Reference: Пластина1	Edit Delete
13	<ul style="list-style-type: none">Type: Body, Specification: A111-111A11-11A, Quantity: 3, Reference: Корпус1Type: Cang, Specification: C333-333C33-33C, Quantity: 3, Reference: Цанга1	Edit Delete
23	<ul style="list-style-type: none">Type: Plate, Specification: B222-222B22-22B, Quantity: 3, Reference: Пластина1Type: Cang, Specification: C333-333C33-33C, Quantity: 3, Reference: Цанга1	Edit Delete

Add Instrument

Рисунок 3. Страница с информацией о всех инструментах

ЗАКЛЮЧЕНИЕ

Реализована часть задач, поставленных при начале работы над проектом. Текущий код проекта можно описать как действующий скелет для программы что будет отвечать требованиям для разработки, описаным ранее. В результате выполнения курсовой работы были получены следующие результаты:

- 1) Реализована логика взаимодействия с базой данных;
- 2) Обеспечен удобный и интуитивно понятный пользовательский интерфейс;
- 3) Реализована возможность просмотра и редактирования данных в приложении;
- 4) Остальные задачи, такие как:
 - Реализована система авторизации и аутентификации пользователей ;
 - Реализация функционала заполнения данных для отчётов на дальнейшую печать;
 - Реализация возможности просмотра отчётов и печати их в нужном формате;

не были реализованны в рамках данного проекта по различным причинам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ahmed K., Hussain A. A Cloud-Based Approach to Inventory and Asset Tracking Systems // Proceedings of the International Conference on Cloud Computing and Security. — Berlin : Springer, 2020. — С. 174—182.
2. Bernstein R. Practical Guide to Web Application Development. — San Francisco : O'Reilly Media, 2018. — 504 с.
3. Bhuiyan M. A., Hossain M. S. Designing and Implementing a Web-Based Warehouse Management System // International Journal of Information Technology and Computer Science. — 2018. — Т. 10, № 3. — С. 12—20.
4. Christian Bauer G. K. Hibernate in Action. — Mel. : Manning Publications, 2004. — 408 с.
5. Doma E. E. Inventory Management: Principles and Practices. — New York : Springer, 2015. — 321 с.
6. Fowler M. Patterns of Enterprise Application Architecture. — Boston : Addison-Wesley Professional, 2019. — 533 с.
7. Garrett J. J. The Elements of User Experience: User-Centered Design for the Web and Beyond. — 2nd. — Berkeley : New Riders Publishing, 2011. — 272 с.
8. Hoffman A. Web Application Security: A Guide for Developers. — NY : Apress, 2019. — 416 с.
9. Hunt A., Thomas D. The Pragmatic Programmer. — Boston : Addison-Wesley Professional, 2000. — 352 с.
10. Lee J., Kim S. Web-Based Inventory Management System for Small and Medium-Sized Enterprises // Journal of Industrial Engineering and Management. — 2020. — Т. 13, № 1. — С. 34—45.

11. Lee J. Kim S. Instrumental Distribution Warehouse: A Case Study on Web-Based Inventory Management // Journal of Industrial Engineering and Management. — 2019. — Т. 12, № 1. — С. 56—67.
12. Lee S., Kim J. Implementing RFID for Real-Time Inventory Monitoring in Warehouses // Proceedings of the International Conference on Embedded Systems and Applications. — Seoul : ACM, 2018. — С. 199—207.
13. Liu Z., Zhang X. Machine Learning Algorithms for Demand Forecasting in Warehouse Management // Proceedings of the International Conference on Artificial Intelligence and Data Science. — Shanghai : Springer, 2020. — С. 89—98.
14. Martinez J., Torres C. Big Data Analytics for Efficient Warehouse Management // Proceedings of the International Conference on Big Data and Analytics. — Madrid : IEEE, 2019. — С. 110—119.
15. Singh M. N., Kaur A. Cloud Computing with Security: A Practical Approach. — Boca Raton : CRC Press, 2018. — 298 с.
16. Thymeleaf vs JSP. — 2024. — URL: <https://mindmajix.com/thymeleaf-vs-jsp> (дата обр. 21.05.2024).
17. Балюк А. С. О верхней оценке сложности трехзначных функций в классе поляризованных полиномов // Синтаксис и семантика логических систем. Материалы 6-й Международной школы-семинара. — Иркутск : Иркутский государственный университет, 2019. — С. 21—22.
18. Белова Н. А. Информационные технологии в логистике и управлении складом. — Москва : Юрайт, 2021. — 365 с.
19. Белова Н. М. Моделирование процессов управления запасами на складах с использованием систем автоматизации // Современные информационные технологии и системы управления. — 2020. — Т. 14, № 3. — С. 29—37.

20. Волков А. П. Разработка системы контроля складских операций на основе веб-технологий // Сборник докладов конференции по автоматизации производственных процессов. — Москва : Издательство МГУ, 2020. — С. 76—80.
21. Воробьев А. И. Разработка веб-приложений с использованием Spring и Hibernate. — Санкт-Петербург : Питер, 2020. — 480 с.
22. Грас В. Web-приложения с нуля: Полный курс. — М. : Альпина Паблишер, 2020. — 368 с.
23. Иванов В. Н. Spring Framework: разработка корпоративных приложений на Java. — Санкт-Петербург : БХВ-Петербург, 2021. — 512 с.
24. Киселев М. П. Особенности интеграции Spring и Hibernate в корпоративных приложениях // Информационные технологии и системы. — 2020. — Т. 15, № 3. — С. 28—35.
25. Коновалов И. В. Особенности разработки веб-приложений для малых предприятий // Информационные системы и технологии. — 2020. — № 4. — С. 47—53.
26. Никитин А. В. Разработка веб-приложений на основе современных технологий. — Санкт-Петербург : Издательство СПбГУ, 2019. — 412 с.
27. Организация инструментального хозяйства. — 2024. — URL: https://studref.com/614569/ekonomika/organizatsiya_instrumentalnogo_hozyaystva (дата обр. 20.05.2024).
28. Организация управления инструментальным хозяйством. — 2016. — URL: <https://studfile.net/preview/5597762/page:20/> (дата обр. 21.05.2024).
29. Петров И. В. Системы управления складом: от теории к практике. — Москва : Инфра-М, 2018. — 288 с.

30. Сергеев П. Н. Применение Hibernate для автоматизации работы с базами данных в веб-приложениях на Java // Материалы Всероссийской конференции по программированию и информационным системам. — Казань : Казанский федеральный университет, 2021. — С. 67—72.
31. Танимура К. SQL для анализа данных. — Спб. : БХВ, 2023. — 384 с.
32. Уоллс К. Spring in Action. Шестое издание. — М. : ДМК Пресс, 2022. — 544 с.

ПРИЛОЖЕНИЕ 1.

UML-диаграмма классов приложения

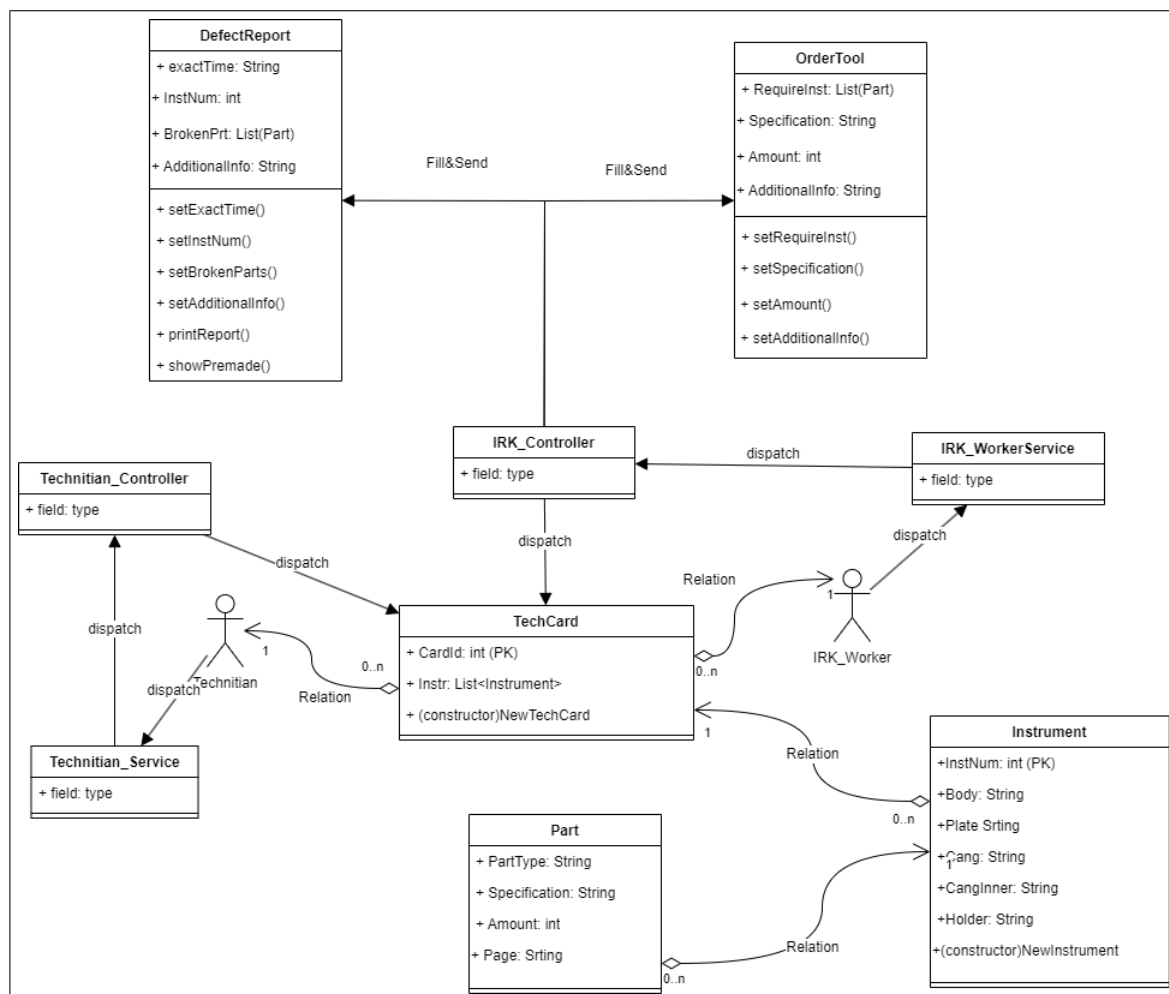


Рисунок 4. UML-диаграмма классов основного модуля приложения