|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **РОСЖЕЛДОР**  **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**  **ВЫСШЕГО ОБРАЗОВАНИЯ**  **СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**  **ПУТЕЙ СООБЩЕНИЯ (СГУПС)**   |  |  |  |  | | --- | --- | --- | --- | | **К защите:** |  |  | | | **Заведующий кафедрой** | **Информационные** | |  | | **технологии транспорта** | | |  | |  | д-р техн. наук, профессор | |  | |  |  | В.И. Хабаров | | | *подпись* |  | *инициалы, фамилия* | | |  |  |  | | | *дата* |  |  | |   **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**  **(БАКАЛАВРСКАЯ РАБОТА)**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Тема:** | Создание прототипа иммунной интеллектуальной системы для обработки данных на сортировочной станции | | | | |  | | | | | |  | | | | | |  |  | БР.БПИ.09.2022 |  |  | |  |  | *шифр документа* |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  | **Выполнил** |  |  |  | **Руководитель** | |  |  | М.П. Лыкова |  |  |  | канд. техн. наук, доц.  А.А. Уланов | | *подпись* |  | *инициалы, фамилия* |  | *подпись* |  | *инициалы, фамилия* | |  |  |  |  |  |  |  | | *дата* |  |  |  | *дата* |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Консультанты по разделам** |  |  |  |  | | 1 |  |  |  |  | |  |  |  |  |  | | 2 |  |  |  |  | |  |  |  |  |  | | 3 |  |  |  |  | |  |  |  |  |  | | Нормоконтролер работы |  |  |  | ст. преп.  Т. А. Распопина | |  |  | *подпись* |  |  | |  |  |  |  |  | |  |  | *дата* |  |  |   **2022 г.** |

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

Факультет: Бизнес-информатики

Кафедра: Информационные технологии транспорта

Направление: 09.03.03 «Прикладная информатика»

Профиль: Интеллектуальные транспортные системы на транспорте

|  |  |
| --- | --- |
|  | ***УТВЕРЖДАЮ****: зав. кафедрой «Информационные технологии транспорта»*  д-р техн. наук, проф.  В. И. Хабаров |
|  | *«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2022 г.* |

**З А Д А Н И Е**

**на выполнение выпускной квалификационной работы**

|  |  |  |  |
| --- | --- | --- | --- |
| студенту | Лыковой Марии Павловне | | |
|  |  | | |
| 1. Тема «Создание прототипа иммунной интеллектуальной системы для обработки данных на сортировочной станции » утверждена приказом № 203/c от «30» мая 2022 г. | | | |
| 2. Задание выдано «12» мая 2022 г. | | | |
| 3. Срок сдачи законченной работы на кафедру «20» июня 2022 г. | | | |
| 4. Исходные данные: данные, полученные в ходе прохождения преддипломной практики | | | |
| 5. Содержание расчетно-пояснительной записки | | | |
| Наименование разделов и вопросов | | Примерное количество страниц | График (сроки) выполнения |
| Введение | | 2 | 17.04.2022 |
| Описание предметной области | | 12 | 01.05.2022 |
| Моделирование бизнес-процессов | | 3 | 05.05.2022 |
| Обоснование среды разработки | | 17 | 11.05.2022 |
| Руководство пользователя | | 7 | 19.05.2022 |
| Заключение | | 1 | 20.05.2022 |

6. Содержание и объемы графической части

|  |  |  |
| --- | --- | --- |
| Наименование графического документа (чертежа, схемы, графика) | Количество  листов  формата А1 | График  (сроки)  выполнения |
| Презентация PowerPoint | 28 | 05.05.2022 |

7. Консультанты по разделам

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Наименование  раздела | | Фамилия, И. О.  консультанта | Подпись консультанта,  дата выдачи задания | |
|  |  | |  |  | |
| Руководитель | |  | | | А.А. Уланов |
|  | | *(подпись, фамилия, И.О.)* | | |  |
| Задание к использованию принял | |  | | | М.П. Лыкова |
|  | | *(подпись студента)* | | |  |

УДК 004.896

**АННОТАЦИЯ**

В работе 49 страниц, 27 рисунков, 2 таблицы, 18 источников, 1 приложение.

Ключевые слова: *иммунитет, иммунные интеллектуальные системы, машинное обучение, информационные системы, сортировочная станция.*

Предметная область – прототип иммунной интеллектуальной системы для обработки данных на сортировочной станции. Проведено исследование иммунных интеллектуальных систем и алгоритмов, на которых они работают. Изучены ошибки, возникающие в данных при прохождении вагонами технологических операций на сортировочной станции. Описаны бизнес-процессы.

Результат: реализован прототип самообучающейся иммунной интеллектуальной системы в среде разработки Microsoft Visual Studio 2019 с использованием языка программирования C#.

**ABSTRACT**

The work contains 49 pages, 27 figures, 2 tables, 18 sources, 1 application.

Keywords: *immunity, immune intelligent systems, machine learning, information systems, marshalling yard.*

The subject area is a prototype of an immune intelligent system for processing data at a marshalling yard. A study of immune intelligent systems and the algorithms on which they work has been conducted. The errors that occur in the data during the passage of technological operations by wagons at the marshalling yard are studied. Business processes are described.

Result: a prototype of a self-learning immune intelligent system was implemented in the Microsoft Visual Studio 2019 development environment using the C# programming language.

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

АСУ СТ - автоматизированная система управления станциями.

БД – база данных.

ДНК – дезоксирибонуклеиновая кислота.

ИИС – иммунная интеллектуальная система.

ИС – информационная система.

ПС – подвижной состав.

СУБД – система управления базами данных.

BPM (business process management) – управление бизнес – процессами.

C# – объектно-ориентированный язык программирования высокого уровня.

ER (Entity Relationship) – модель «сущность-связь».

IDE (Integrated development environment) – интегрированная среда разработки.

IDS (Intrusion Detection System) – система обнаружения вторжений.

ML (Machine Learning) – машинное обучение.

WPF (Windows Presentation Foundation) - система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем.

Microsoft SQL Server – система управления реляционными базами данных компании Microsoft.

MVVM (Model View ViewModel) - паттерн, который позволяет отделить логику приложения от визуальной части (представления).

SQL (Structured Query Language) – декларативный язык программирования, применяемый для управления данными в реляционной базе данных.

UML (Unified Modeling Language) – унифицированный язык моделирования.

Visual Studio – среда разработки от компании Microsoft.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc105115160)

[1 Описание предметной области 8](#_Toc105115161)

[1.1 Основные термины и определения 8](#_Toc105115162)

[1.2 Общие сведения об иммунной системе 9](#_Toc105115163)

[1.3 Связь с предметной областью 11](#_Toc105115164)

[1.4 Мультиагентные информационные системы 12](#_Toc105115165)

[1.5 Описание работы сортировочной станции 15](#_Toc105115166)

[1.6 История изучения иммунных интеллектуальных систем 17](#_Toc105115167)

[1.7 Машинное обучение 18](#_Toc105115168)

[2 Моделирование бизнес – процессов 20](#_Toc105115169)

[3 Обоснование среды разработки 23](#_Toc105115170)

[4 Руководство пользователя 40](#_Toc105115171)

[Заключение 47](#_Toc105115172)

[Список использованных источников 48](#_Toc105115173)

[Приложение А Техническое задание 50](#_Toc105115174)

# **ВВЕДЕНИЕ**

Сегодня высокоразвитые и современные транспортные и логистические системы являются ключевыми факторами в экономической конкурентоспособности страны. Это важное направление деятельности и нужно развивать его, предлагая новые технологии. На данный момент иммунные информационные системы успешно применяются для решения задач оптимизации и классификации данных, для сжатия информации, кластеризации, поиска аномалий, машинного обучения, обработки неструктурированных данных и извлечения информации, компьютерной безопасности и адаптивного контроля. Такую концепцию можно развивать и успешно использовать в улучшении деятельности российских железных дорог.

Цель работы - создать прототип иммунной интеллектуальной системы для обработки данных на сортировочной станции.

В первом разделе проводится исследование предметной области, связанной с перемещением вагонов на сортировочной станции. Во втором разделе описаны бизнес-процессы. В третьем разделе приведено обоснование выбранной среды разработки. В четвертом разделе представлено подробное руководство пользователя для знакомства с прототипом иммунной интеллектуальной системы.

Степень разработанности в литературе (основные авторы):

* Нуньес де Кастро, Леандро. Искусственные иммунные системы: новый подход к вычислительному интеллекту;
* Джеффри О. Кефарт. Биологически вдохновленная иммунная система для компьютеров;
* Уве Айкелин, Джули Гринсмит, Джейми Твикросс. Подходы иммунной системы к обнаружению вторжений – обзор;
* Руслан Меджитов. Распознавание микроорганизмов и активация иммунного ответа;
* Амир Хуссейн, Кайжу Хуан. Новые искусственные иммунные сети на основе оптимизации классификаторов мелкого машинного обучения (ML);
* С.В. Гладыш. Иммунокомпьютинг в управлении инцидентами информационной безопасности;
* Маркос Цукколотто, Карлос Эдуардо Перейра, Лука Фазанотти, Серджио Кавальери, Джей Ли. Разработка искусственных иммунных систем для интеллектуальных систем технического обслуживания.

Научная новизна данной работы заключается в том, что результаты теоретического обзора по иммунным интеллектуальным системам были применены к актуальному состоянию информационной системы ОАО «РЖД», что позволило сделать выводы относительно использования иммунных интеллектуальных систем для совершенствования информационной системы ОАО «РЖД».

Практическая значимость работы состоит в возможности внедрения результатов исследования в деятельность ОАО «РЖД» в целях подготовки и реализации мер, направленных на совершенствование информационной системы.

При написании работы использовались такие методы исследования как: сбор и анализ информации, обработка данных, методы моделирования (UML), табличный метод представления и обработки информации, метод аналогий, метод графического представления информации (диаграммы, иллюстрации).

Используемые источники: в качестве источников информации использовались научные статьи, учебная и периодическая литература, сайт компании, техническая документация.

# **1 Описание предметной области**

# **Основные термины и определения**

Иммунная интеллектуальная система (ИИС) – адаптивная вычислительная система, использует модели, принципы, механизмы и функции из теоретической иммунологии, применяется для решения прикладных задач. В искусственной иммунной системе в качестве описания взаимодействия компонентов системы (антител) используются меры аффинности, а в качестве механизмов адаптации – иммунные алгоритмы. Чтобы построить такую систему, необходимо знать целевую функцию.

Аффинность – термодинамическая характеристика, количественно описывающая силу взаимодействия антигена и антитела. Применительно к искусственным иммунным системам аффинность – это характеристика, оценивающая степень близости (схожести) генетических наборов антигена и антитела. Для искусственных иммунных систем, основанных на бинарном коде, аффинность оценивается с помощью расстояния Хэмминга. Для систем, основанных на вещественном коде – с помощью евклидова расстояния [1].

Иммунная система представляет большой интерес для специалистов в области программирования и информатики, по сути она является примером децентрализованной обработки информации, выполняет параллельно большой объем сложных «вычислений». При этом иммунная система является примером биологической системы с прекрасными адаптивными механизмами на локальном уровне и эмерджентных механизмов поведения на глобальном уровне [2].

Существует три алгоритма для работы интеллектуальных иммунных систем, но мы рассмотрим два из них. Они опираются на теории о природных иммунных системах. Эти теории описывают взаимодействие элементов в системе и само ее функционирование:

* клональный алгоритм отбора - это класс алгоритмов, основанный на теории клоновой селекции [3]. С точки зрения иммунитета это действует так: антиген проникает в организм и начинает распространяться, поражая здоровые клетки токсинами. Антитела распознают антиген, тоже начинают размножаться и мутируют, что позволяет им лучше бороться с антигеном. С точки зрения информатики антигеном является вирус, вредоносное программное обеспечение и так далее. Механизмы, которые лучше всего реагируют на угрозу – антитела, а их мутация – это обучение. Множество принципов искусственных иммунных систем отлично коррелируют со сферой обеспечения информационной безопасности компьютерных систем;
* негативный алгоритм отбора – относится к нахождению и удалению негативно реагирующих клеток. В организме существуют некие «шаблоны» - лимфоциты, которые создаются организмом. Этот класс алгоритмов используется для распознавания и классификации проблемных областей. Например, для классификации и распознавания аномалий в различных данных [2].

Главным принципом действия человеческой иммунной системы является сравнение определенных «шаблонов» с находящимися внутри организма телами и выявление таким образом инородных тел. «Шаблонами» являются лимфоциты, постоянно генерируемые спинным мозгом и тимусом с учетом информации, содержащейся в ДНК. Благодаря негативной селекции создаются «шаблоны», соответствующие телам, которые внутри организма отсутствуют, и, если какое-то тело подходит под данный шаблон, значит, оно явно чужое. В случае обнаружения лимфоцитами инородного тела на основании соответствующего шаблона вырабатываются антитела, которые и уничтожают его.

# **Общие сведения об иммунной системе**

Иммунная система — это система биологических процессов и структур организма, которая обеспечивает ему защиту от инфекций, вирусов, токсинов и злокачественных клеток. Для хорошей работы иммунной системы она должна уметь распознавать широкий спектр патогенов и отличать их от собственных здоровых тканей организма. У многих видов живых существ есть две подсистемы: врождённая и приобретённая (адаптивная) иммунные системы. И та, и та используют гуморальные и клеточные механизмы. Одним из основных инструментов адаптивной иммунной системы является иммунологическая память, именно из – за нее организм способен развить более сильный иммунный ответ на патоген при повторной встрече. Иммунологическая память лежит в основе вакцинации [4].

Рассмотрим принцип работы защитной системы организма, он состоит из нескольких пунктов:

1) При попадании в организм посторонних частиц (антигенов) B-лимфоциты начинают выделять специальные антитела, которые являются специфическим белком, способным заблокировать антиген.

2) Далее активизируются T-лимфоциты, которые начинают выделять клетки, способные уничтожить заблокированный антиген.

3) После этого антиген запоминается, антитела, выработанные лейкоцитами, остаются в организме. Это помогает предотвратить похожую инфекцию [5].

Рассмотрим виды клеток иммунной системы:

* Т-лимфоциты. Есть несколько видов: Т-киллеры убивают микроорганизмы, Т-хелперы помогают распознавать и убивать микробов;
* В-лимфоциты. Основная их цель – вырабатывать антитела. Антитела – это вещества, которые связываются с белками микроорганизмов, инактивируют их и выводятся из организма человека, тем самым убивая инфекцию внутри организма;
* нейтрофилы. Они разрушают инородную клетку, при этом сами также разрушаются. После появляется гнойное отделяемое. Наглядный пример работы нейтрофилов – воспалённая рана на коже с гнойным отделяемым;
* макрофаги. Они также уничтожают микробов, но не разрушаются и уничтожают их в себе или передают на распознавание Т-хелперам;
* эозинофилы. Клетки, которые вырабатывают вещества для разрушения паразитов в организме.

Есть еще виды клеток, которые выполняют важные, но узкоспециализированные функции. Их изучают только узкие специалисты и ученые [6].

# **Связь с предметной областью**

Рассмотрим в качестве организма одну из информационных систем ОАО «РЖД» - АСУ СТ, а в качестве входящего потока – данные, которые попадают в нее.

АСУ СТ - это очень большая, многопользовательская система, которая включает огромное количество данных. Данные (клетки) должны подаваться в определенные рабочие места (органы) и соответствовать норме (нормализованность). В тоже время при внесении данных в систему человек может что – то упустить, внести некорректно. Это в свою очередь может привести к некорректному расчету показателей и, как следствие, убыткам компании (организации в целом). Поэтому считаем, что внедрение ИИС в ИС ОАО «РЖД» является актуальным.

Система должна обладать способностью проверять вводимые данные на корректность, не давать вносить противоречивые данные или необоснованно их менять. Все это очень напоминает иммунную систему человека.

В ходе изучения данных, связанных с поездами, прибывающими в переработку на сортировочную станцию, были выявлены следующие аномалии в данных:

* появление вагона без отметки о времени начала операции;
* ошибки при внесении геолокации вагона (номер пути);
* нахождение двух объектов на одном пути одновременно;
* изменение данных о вагоне.

Анализ данных показал, что есть ошибки при вводе данных, приводящие к неправильной интерпретации данных и воспроизведению процессов с помощью нейронных сетей. Эти ошибки можно обрабатывать негативным алгоритмом отбора и обучать ИИС с помощью клонального алгоритма.

Создаваемая иммунная интеллектуальная система должна отслеживать ошибки в существующих данных и сопоставлять со своей базой. Уже существующие аномальные данные система может либо удалять, либо предлагать варианты действий с этими данными пользователю.

Для реализации ИИС принято решение об использовании мультиагентных систем.

# **Мультиагентные информационные системы**

Мультиагентная система — это система, которая образована несколькими взаимодействующими интеллектуальными агентами [7]. Обычно эти системы используют для решения проблем, которые сложно или невозможно решить с помощью одного агента или монолитной системы. Например, при онлайн торговле, моделировании, ликвидаций аварий и чрезвычайных ситуаций [8].

Агенты в мультиагентной системе имеют характеристики:

* автономность (агенты всегда, хотя бы частично, независимы);
* ограниченность представления (ни один агент не имеет представления о всей системе в целом);
* децентрализация (нет таких агентов, которые бы управляли всей системой) [9].

В мультиагентных системах может быть самоорганизация и сложное поведение даже если стратегия поведения каждого отдельного агента достаточно проста. Это лежит в основе так называемых муравьиных алгоритмов [10].

Агенты могут делиться полученными знаниями, используя некоторый специальный язык и подчиняясь установленным правилам «общения» (протоколам) в системе [11].

Как уже упоминалось, в иммунной системе человека есть несколько основных типов клеток: Т - лимфоциты, В - лимфоциты, нейтрофилы, макрофаги, эозинофилы. И все они выполняют определенные функции иммунной интеллектуальной системы. При создании предлагается рассмотреть каждый отдельный тип клеток как одного агента в мультиагентной системе.

Интеллектуальная система представляет собой распределенную децентрализованную систему временных коллективов агентов (Т - лимфоцитов, В - лимфоцитов, нейтрофилов, макрофагов, эозинофилов), которая способна к адаптивной интеллектуальной обработке поступающей информации. Рассмотрим основную способность интеллектуальной системы: умение распознавать верно (своих) или неверно (чужих) введенные данные (антигены) с последующей их классификацией и запуском соответствующих защитных механизмов. Также по результатам распознавания система будет обучаться и формировать память к антигену. Знания об антигенах будут использоваться при возникновении новых похожих угроз. Система будет создавать и совершенствовать базу знаний в процессе работы.

Найдем сходства функций естественной иммунной системы с основными функциями, которые должна выполнять проектируемая система распознавания данных:

− выявление и регистрация событий, имеющих признаки инцидента;

− идентификация инцидента на основе оперативного анализа ситуации и принятие решения в условиях не полной определенности имеющейся информации;

− обработка и устранение последствий ошибки путем предупреждения человека и удаления (замены) неверных данных [12].

Будем проектировать данную систему, используя иммунно - мультиагентную технологию. На рисунке 1.1 представлены четыре класса агентов: агенты - детекторы, агенты - идентификаторы, агенты - координаторы, агенты - реакторы.

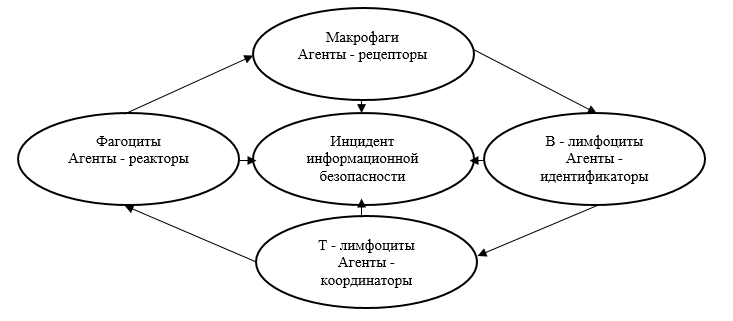


Рисунок 1.1 – Четыре класса агентов

Агенты-детекторы соответствуют макрофагам и другим антиген-презентирующим клеткам, которые выставляют частицы антигена на своей поверхности, привлекая внимание В - лимфоцитов для распознавания. Агенты-идентификаторы соответствуют В-лимфоцитам, которые распознают антиген и заранее подвергались «отрицательному отбору» в тимусе. Агенты-координаторы соответствуют лимфокинам, выделяемым Т-лимфоцитами для активации В-лимфоцитов. Агенты-реакторы соответствуют фагоцитам, имеющим антитела для уничтожения антигена.

Выделим этапы управления инцидентами с помощью ИИС:

* идентификация агентами-детекторами любого ввода аномальных данных;
* распознавание агентами-идентификаторами ненормального ввода как определенного типа инцидента при условии нахождения в базе знаний соответствующей сигнатуры или выявление аномалии по отношению к эталону поведения;
* получение подсистемой реагирования сигнала от IDS об идентифицированном известном или неизвестном инциденте;
* идентификация атакующего набора угроз инцидента при условии наличия в базе знаний корреляции между характеристиками полученного сигнала об инциденте и записями о наборах неверных данных;
* принятие решения относительно выбора инцидентно-ориентированного набора механизмов защиты;
* выдача подсистемой обработки управляющего сигнала агентам-реакторам относительно обработки инцидента с помощью инцидентно-ориентированного набора механизмов защиты;
* самоорганизация и оценка подсистемой обратной связи и агентами - детекторами эффективности использования инцидентно-ориентированного набора механизмов защиты, пополнение баз знаний новым опытом, расследование и анализ инцидента, выработка управляющего сигнала относительно превентивных действий.

Чтобы система работала как единый организм, агенты должны обеспечивать гомеостатическое регулирование системы в целом. Под гомеостатическим регулированием понимается управление инцидентами, поддерживающее целевые характеристики в пределах, обеспечивающих ее безопасность, качество, надёжность и живучесть [13].

# **Описание работы сортировочной станции**

Сортировочная станция — техническая железнодорожная станция, предназначенная для расформирования и формирования различных категорий поездов в соответствии с планом формирования из отдельных вагонов, выполнения операций по пропуску транзитных поездов без переработки, технического обслуживания и коммерческого осмотра составов поездов и устранения выявленных неисправностей вагонов, смены локомотивов и локомотивных бригад [14].

Основная задача сортировочной станции — выполнение переработки вагонопотоков и формирование поездов в оптимальном режиме, с тем, чтобы нахождение вагона на станции было по времени минимальным и технологически обоснованным [14].

На сортировочной станции последовательно происходят следующие операции:

* прибытие вагонов в составе поезда (отдают документы на проверку);
* ожидание осмотра (до 40 минут, зависит от загруженности станции);
* осмотр поезда (15-20 минут);
* завершение обработки поезда по прибытию (работа с документами на этом этапе заканчивается тоже);
* ожидание надвига поездом (15-20 минут);
* надвиг поезда (10 минут);
* ожидание расформирования (15-20 минут максимум);
* расформирование (15-20 минут на весь состав);
* нахождение в сортировочном парке (вагоны стоят несцепленные, но уже на нужных путях. Ждут, когда наберется полный состав. Максимально 3 суток, минимально 40 минут, нормативы 3 часа);
* ожидание выставления поезда в парк отправления (происходит сцепка вагонов и тормозов. Занимает один – два часа);
* выставление в парк отправления (15 минут. Также смотрят на график отправления поезда);
* ожидание локомотива (20-40 минут);
* отправление (10 минут максимум).

Все перечисленные процессы будут смоделированы в рамках разрабатываемого прототипа иммунной интеллектуальной системы. Также будет учитываться время выполнения каждой операции.

# **История изучения иммунных интеллектуальных систем**

Иммунные интеллектуальные системы первый раз стали упоминаться в середине 1980-ых годов. Их описали в своей статье Джей Дойн Фармер (американский ученый в области сложных систем, является профессором математики в Оксфордском университете), Норман Гарри Паккард (физик теории хаоса, является выпускником Рид - колледжа и Калифорнийского университета в Санта-Крусе. Известен своим вкладом в теорию хаоса и в клеточные автоматы) и Алан Перельсон (американский биофизик, иммунолог-теоретик, занимается разработкой моделей иммунной системы и инфекционных заболеваний. Лауреат премии Макса Дельбрюка Американского физического общества — за вклад в теоретическую иммунологию) в 1986 году. После них рассмотрением этих систем занимались Берсини и Франсиско Варела (чилийский биолог, философ и специалист в нейронауках) в 1990 (статьи по иммунным сетям) [15]. Однако только в середине 1990-х годов ИИС стали самостоятельной отраслью. Форрест (отрицательный отбор) и Кепхарт опубликовали первую статью по иммунным системам в 1994, а после этого Дасгупта Д. провел значительные исследования алгоритмов отрицательного отбора. Хант и Кук начали работу над моделями иммунной сети в 1995 году; Тиммис и Нил продолжили эту работу и внесли некоторые улучшения. В 1999 году под издательством Дасгупты была выпущена первая книга по искусственным иммунным системам.

В 2008 году Дасгупта и Нино опубликовали учебник, в котором представлен сборник новейших работ, связанных с методами, основанными на иммунитете [2].

Интеллектуальные иммунные системы активно развиваются в разных направлениях использования искусственного интеллекта. При их реализации возникают проблемы, которые мешают быстрому развитию. Сложно добиться стопроцентно верного распознавания образов, создавать эффективные и простые методики обучения для интеллектуальных систем, научить обрабатывать большие объемы данных. Необходимо, чтобы система из множества факторов выделяла информативные признаки, обучалась и оценивала прогресс обучения. В настоящее время практически не существует интеллектуальных иммунных систем, которые могли бы в реальном времени прогнозировать поведение и управление многомерными объектами с разными видами неопределенности параметров [16].

Масштабные исследования такого рода систем ведутся недавно. Одно из первых достижений в данной области принадлежит ученым Лондонского Королевского колледжа. Они рассказали о разработке в рамках проекта The Computational Immunology for Fraud Detection (CIFD) защитной системы для интернета на базе иммунных интеллектуальных систем. Они считают, что на реализацию уйдет не менее трех лет. Первым же пользователем готовой системы обнаружения вторжений, реализующей принципы иммунной интеллектуальной системы должна стать почта Великобритании.

Искусственные иммунные системы все больше интересуют современных ученых. Это связано с тем, что они содержат в себе лучшие особенности биоинспирированных методов. Из эволюционных алгоритмов они взяли динамическое расположение элементов, а из искусственных нейронных сетей – принципы обучения. Для того, чтобы понять принципы работы ИИС, необходимо рассмотреть, как устроена иммунная система.

# **Машинное обучение**

Машинное обучение (ML) — это использование математических моделей данных, которые помогают компьютеру обучаться без непосредственных инструкций. Оно считается одной из форм искусственного интеллекта (ИИ). При машинном обучении с помощью алгоритмов выявляются закономерности в данных. На основе этих закономерностей создается модель данных для прогнозирования. Чем больше данных обрабатывает такая модель и чем дольше она используется, тем точнее становятся результаты. Это очень похоже на то, как человек оттачивает навыки на практике.

Благодаря адаптивному характеру машинного обучения оно отлично подходит для сценариев, в которых данные постоянно изменяются, свойства запросов или задач нестабильны, или написать код для решения фактически невозможно.

Машинное обучение считается подмножеством ИИ. «Интеллектуальный» компьютер мыслит, как человек и самостоятельно выполняет задачи. Один из способов обучить компьютер имитировать мышление человека — использовать нейронную сеть. Это серия алгоритмов, смоделированных по принципу работы человеческого мозга.

Есть три основных методики машинного обучения:

* контролируемое обучение. Эта методика подходит для наборов данных с метками или структурой. Данные выступают в качестве преподавателя. Они «обучают» компьютер, расширяя его возможности прогнозирования или принятия решения;
* неконтролируемое обучение. Эта методика подходит для наборов данных без меток или структуры. Чтобы определить закономерности и связи, данные группируются в кластеры;
* обучение с подкреплением. Заменяющий оператора агент (программа, которая действует от чьего-либо имени) помогает определить результат на основе цикла обратной связи.

# **2 Моделирование бизнес–процессов**

Бизнес-процесс — совокупность взаимосвязанных мероприятий или работ, направленных на создание определённого продукта или услуги для потребителей. Управленческая концепция BPM рассматривает бизнес-процессы как важные ресурсы предприятия, и предполагает управление ими как одну из ключевых организационных систем.

Процессы в организации могут быть цикличные или однократные. Цикличные направлены на то, чтобы задачи могли выполняться неограниченное количество раз. Однократные же направлены на выполнение одной конкретной задачи один раз.

Диаграмма классов — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма классов является ключевым элементом в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента:

* в верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом. Имена классов начинаются с заглавной буквы. Если класс абстрактный — то его имя пишется полужирным курсивом;
* посередине располагаются поля (атрибуты) класса. Они выровнены по левому краю и начинаются с маленькой буквы;
* нижняя часть содержит методы класса. Они также выровнены по левому краю и пишутся с маленькой буквы.

Язык UML предоставляет механизмы для представления членов класса, например, атрибутов и методов, а также дополнительной информации о них.

Система будет работать с классом подвижной состав и экземпляром класса грузовой вагон. На рисунке 2.1 представлена диаграмма классов, в которой отображены все атрибуты данного класса. Также приведены операции класса.

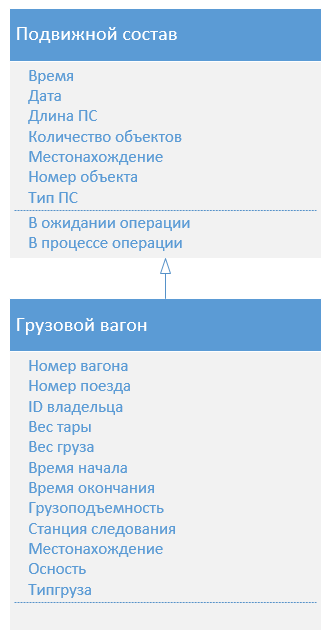


Рисунок 2.1 – Диаграмма классов

Далее опишем бизнес – процесс, с помощью диаграммы State machine унифицированного языка моделирования UML. Конечный автомат (state machine) - модель для спецификации поведения объекта в форме последовательности его состояний, которые описывают реакцию объекта на внешние события, выполнение объектом действий, а также изменение его отдельных свойств. Диаграмма состояний грузового вагона представлена на рисунке 2.2.

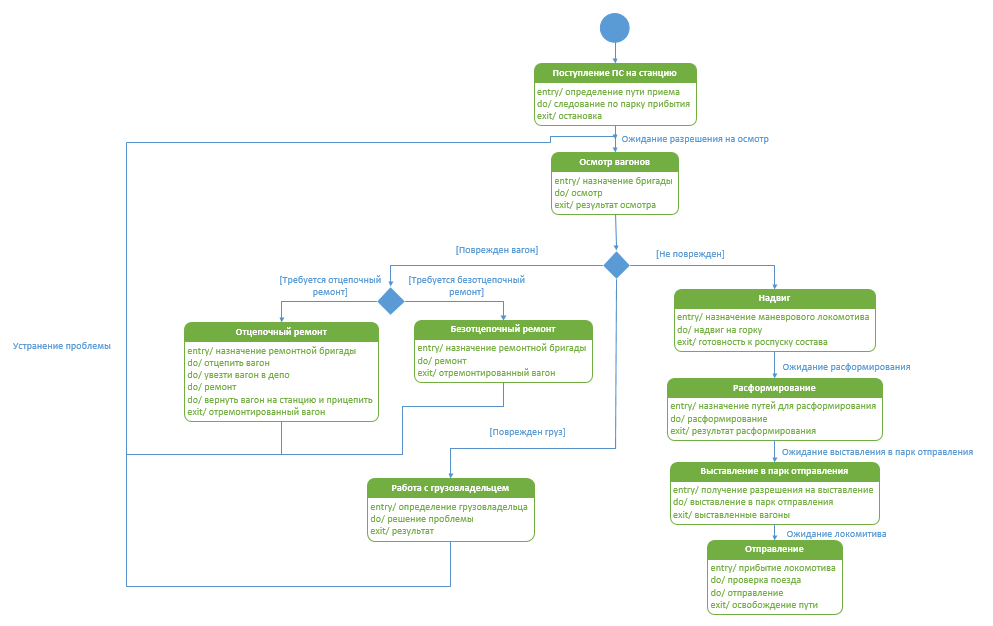


Рисунок 2.2 – Диаграмма состояний грузового вагона

Исходя из рисунка 2.2 видно, что бизнес – процесс обработки транзитного вагона без переработки может идти по стандартному или по альтернативному сценарию.

# **3 Разработка**

# **3.1 Обоснование среды разработки**

Прикладное (специальное) программное обеспечение - это совокупность программ, разработанных при создании конкретной информационной системы.

Предназначено для решения конкретных задач пользователя и организации вычислительного процесса информационной системы в целом.

Интегрированная среда разработки (IDE) — это многофункциональная программа, которая поддерживает многие аспекты разработки программного обеспечения. IDE – это инструмент, используемый для разработки приложений простым, быстрым и надежным способом.

Существует много разных IDE для разных языков программирования, платформ и операционных систем.

Средства разработки программного обеспечения – совокупность приемов, методов, методик, а также набор инструментальных программ (компиляторы, прикладные/системные библиотеки и так далее), используемых разработчиком для создания программного кода программы, отвечающего заданным требованиям.

Выбор среды разработки зависит от поставленной задачи и используемого языка программирования.

Выбор языка программирования во время создания программы является очень важным моментом, от которого зависит очень многое – скорость создания программы, скорость тестирования, возможность переноса на другие платформы, возможность быстрого внесения изменений, быстрота выполнения конечного продукта и так далее.

Рассмотрим возможные среды разработки и языки программирования.

Для начала сравним среды разработки Visual Studio, Project Rider, Eclipse и Visual Studio Code [17]. Составим сравнительную таблицу, в которой рассмотрим важные критерии. Это представлено в таблице 3.1.

Таблица 3.1 – Критерии выбора среды разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Visual Studio | Project Rider | Eclipse | Visual Studio Code |
| Бесплатно | Да | Нет | Да | Да |
| Функциональность | Множество качественных плагинов. С их помощью можно расширить функциональность приложения и подключить другие языки | Project Rider позволяет подключить MSBuild и XBuild, работать с CLI-проектами и организовать отладку приложений .NET and Mono. Множество опций для быстрого создания кода улучшает производительность | Благодаря разработчикам-официалам и членам сообщества с помощью Eclipse можно провести любой C#-продукт по полному циклу разработки | Низкая функциональность. Несмотря на поддержку  .NET-платформы, VCS неудобен для сложных проектов. |
| Поддерживает платформы .NET | Да | Да | Да | Да |
| Кроссплатформенность | Да | Да | Да | Да |
| Сложность | Самостоятельно освоить Visual Studio новичку будет непросто — слишком много доступных функций, спрятанных в подразделах меню | Нет | Как и любой функциональный продукт, Eclipse может показаться новичку слишком сложным. | Да |
| Наличие облачного хранилища | Да | Нет | Нет | Да |
| Встроенный отладчик | Да | Да | Да | Да |
| Подсветка синтаксиса | Да | Да | Да | Да |

Далее сравним языки программирования C# и Python [18]. Также отобразим сравнение их характеристик, представлено в таблице 3.2.

Таблица 3.2 – Критерии выбора языка программирования

|  |  |  |
| --- | --- | --- |
|  | C# | Python |
| Скорость | C# предлагает отличную производительность и скорость благодаря Common Language Infrastructure (CLI). | Python имеет низкую скорость, потому что это интерпретируемый язык, и он выполняет код построчно |
| Надежность | Да | Да |
| Синтаксис | Идентичный языкам C и C ++ | Python легко читать и изучать благодаря синтаксису, похожему на английский. Кроме того, исключается использование точки с запятой после конца оператора и разделителей для начала и конца блока. |

Продолжение таблицы 3.2

|  |  |  |
| --- | --- | --- |
|  | C# | Python |
| Использование памяти | Динамически типизированный характер C# облегчает разработчикам поиск ошибок в коде. C# устраняет проблему утечки памяти. | Это не идеальный выбор для задач с интенсивным использованием памяти, поскольку он потребляет большой объем памяти из-за гибкости типов данных. |
| Многопоточность | Мы легко можем добиться многопоточности в C #. | Из-за глобальной блокировки интерпретатора (GIL) Python требует нескольких процессов для достижения многопоточности. |

В результате сравнения разных сред разработки и разных языков программирования для создания прототипа иммунной интеллектуальной системы с функцией обработки данных на сортировочной станции была выбрана среда разработки Visual Studio (2019), язык программирования – C#.

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Основные преимущества Visual Studio:

* встроенный Web-сервер. Наличие в Visual Studio интегрированного Web-сервера позволяет запускать Web-сайт прямо из среды проектирования, а также повышает безопасность, исключая вероятность получения доступа к тестовому Web-сайту с какого-нибудь внешнего компьютера, поскольку тестовый сервер может принимать соединения только с локального компьютера;
* поддержка множества языков при разработке. Visual Studio позволяет писать код на своем языке или любых других предпочитаемых языках, используя все время один и тот же интерфейс (IDE);
* интуитивный стиль кодирования. По умолчанию Visual Studio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения и менее подверженным ошибкам;
* более высокая скорость разработки. Многие из функциональных возможностей Visual Studio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции IntelliSense (которая умеет перехватывать ошибки и предлагать правильные варианты), функции поиска и замены (которая позволяет отыскивать ключевые слова как в одном файле, так и во всем проекте) и функции автоматического добавления и удаления комментариев (которая может временно скрывать блоки кода), позволяют разработчику работать быстро и эффективно;
* возможности отладки. Предлагаемые в Visual Studio инструменты отладки являются наилучшим средством для отслеживания загадочных ошибок и диагностирования странного поведения. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти;
* Visual Studio также имеет и множество других функций: возможность управления проектом; встроенная функция управления исходным кодом; возможность рефакторизации кода; мощная модель расширяемости.

C# — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET.

Преимущества языка C#:

* поддержка подавляющего большинства продуктов Microsoft;
* типы данных имеют фиксированный размер (32-битный int и 64-битный long), что повышает «мобильность» языка и упрощает программирование, так как вы всегда знаете точно, с чем вы имеете дело;
* автоматическая «сборка мусора». Это значит, в большинстве случаев не нужно думать об освобождении памяти. Общеязыковая среда CLR сама вызовет сборщик мусора и очистит память;
* большое количество специальных конструкций, разработанных для понимания и написания кода. Они не имеют значения при компиляции;
* низкий порог вхождения. Синтаксис C# имеет много схожего с другими языками программирования, благодаря чему облегчается переход для программистов. Язык C# часто признают наиболее понятным и подходящим для новичков;
* с помощью Xamarin на C# можно писать программы и приложения для таких операционных систем, как iOS, Android, MacOS и Linux.

Для хранения информации при работе системы будет реализована связь с базой данных SQL.

База данных (БД) – это совокупность массивов и файлов данных, организованная по определённым правилам, предусматривающим стандартные принципы описания, хранения и обработки данных независимо от их вида.

Система управления базами данных, сокр. СУБД (англ. Database Management System, сокр. DBMS) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

SQL (англ. structured query language — «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. SQL считается языком программирования, в общем случае (без ряда современных расширений) не является тьюринг-полным, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

# **3.2 Анализ данных**

Проанализируем созданную в ходе составления технического задания концептуальную модель предметной области, внесем необходимые изменения.

Итак, нам нужно отобразить такие данные как: номер вагона, номер поезда, ID владельца, вес тары, вес груза, время начала операции, время окончания операции, грузоподъемность, станция следования, местонахождение, осность, ID типа груза. Все эти данные содержатся в таблице «Данные о вагоне».

Далее нам нужно иметь две отдельные таблицы, в которых содержатся описание владельцев вагонов и описание типов грузов. Они называются «Владельцы вагонов» и «Типы грузов» соответственно.

Четвертая таблица содержит данные о локации вагона. Назовем ее «Локация вагона». Она содержит такие данные как: ID парка, наименование парка.

Пятая таблица называется «Операции над вагонами». Она содержит такие данные как: ID операции, наименование операции, ID парка, минимальное время прохождения, максимальное время прохождения.

Концептуальная модель представляется в виде диаграммы сущностей – связей (entity – relationship) или ER-диаграммы. Процесс построения ER-диаграммы называется ER-моделированием.

Построим ER – диаграмму, она представлена на рисунке 3.1.

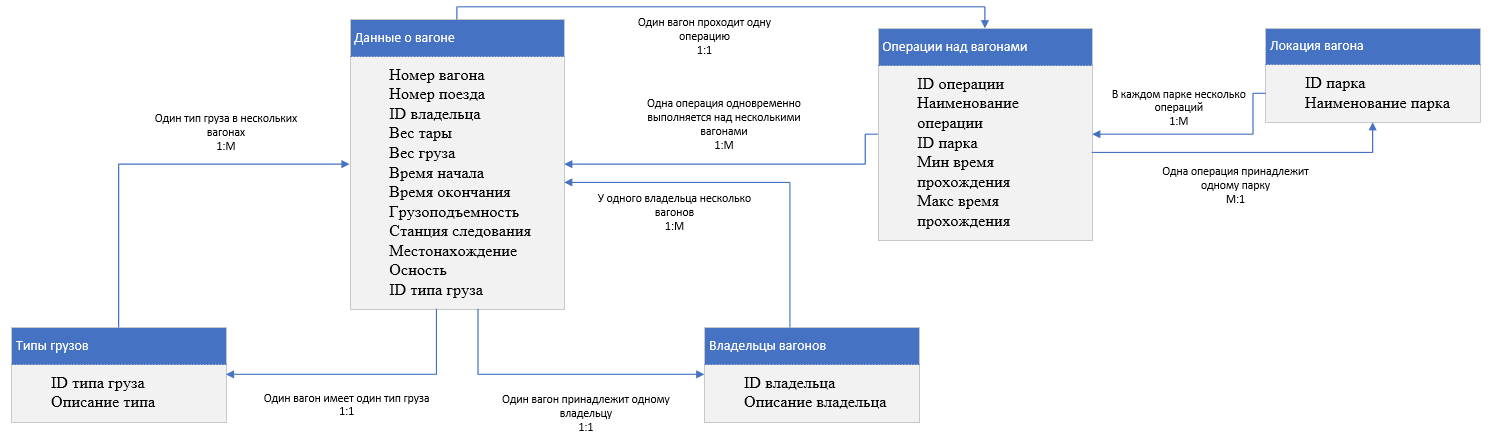


Рисунок 3.1 – ER – диаграмма

Наполним таблицы разработанной БД актуальной информацией, приведем в отчет пример наполнения. На рисунке 3.2 представлен пример заполнения таблицы «Данные о вагоне».

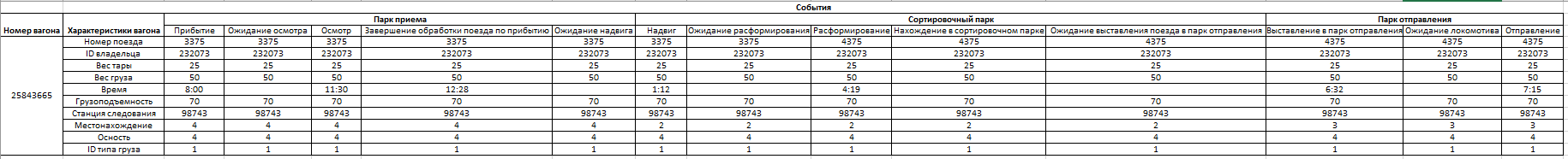


Рисунок 3.2 – Пример заполнения таблицы «Данные о вагоне»

На рисунке 3.3 представлен пример заполнения таблицы «Владельцы вагонов»



Рисунок 3.3 – Пример заполнения таблицы «Владельцы вагонов»

На рисунке 3.4 представлен пример заполнения таблицы «Типы грузов».

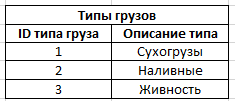


Рисунок 3.4 – Пример заполнения таблицы «Типы грузов»

На рисунке 3.5 представлен пример заполнения таблицы «Локация вагона»

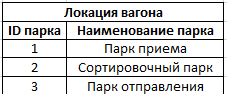


Рисунок 3.5 – Пример заполнения таблицы «Локация вагона»

На рисунке 3.6 представлен пример заполнения таблицы «Операции над вагонами».



Рисунок 3.6 – Пример заполнения таблицы «Операции над вагонами»

# **3.3 Реализация приложения**

Опишем то, какие технологии используются в приложении.

MVVM (Model-View-ViewModel) – паттерн, который позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, задает общую архитектуру приложения.

Данный паттерн был представлен Джоном Госсманом (John Gossman) в 2005 году как модификация шаблона Presentation Model и был первоначально нацелен на разработку приложений в WPF. И хотя сейчас данный паттерн вышел за пределы WPF и применяется в самых различных технологиях, в том числе при разработке под Android, iOS, тем не менее WPF является довольно показательной технологией, которая раскрывает возможности данного паттерна.

На рисунке 3.7 представлено, что MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

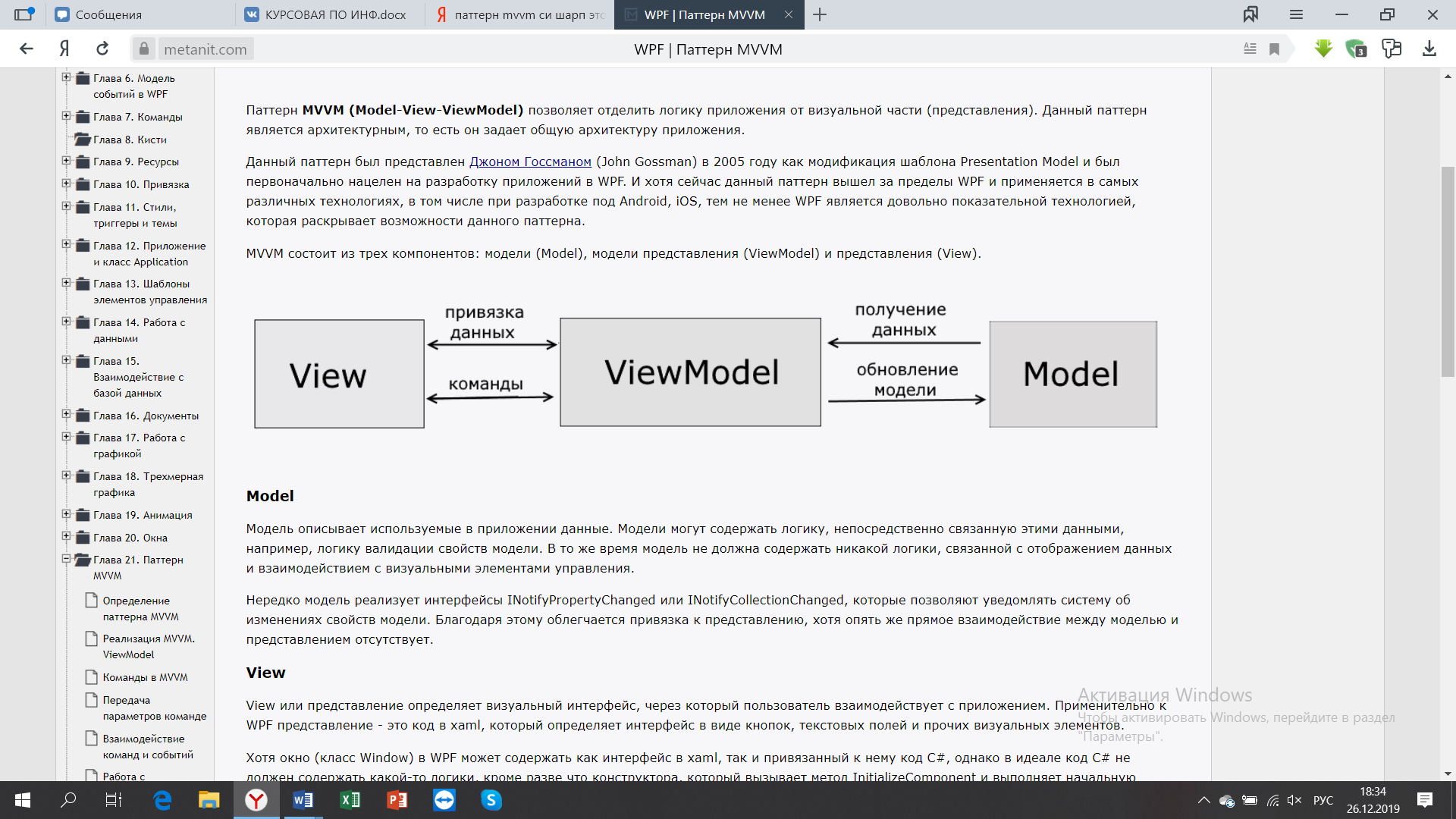


Рисунок 3.7 – Основные компоненты Model-View-ViewModel

View и ViewModel отвечают за отображение данных.

* Model содержит бизнес–логику приложения и связанные с ней данные. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления;
* View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление - это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов. Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel;
* ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также ViewModel определяет логику по обновлению данных в модели. Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд. Например, пользователь хочет сохранить введенные в текстовое поле данные. Он нажимает на кнопку и тем самым отправляет команду во ViewModel. А ViewModel уже получает переданные данные и в соответствии с ними обновляет модель.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

ORM (Object-Relational Mapping) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.

Задача ORM: обеспечить работу с данными в терминах классов, а не таблиц данных, и, напротив, преобразовать термины и данные классов в данные, пригодные для хранения в СУБД. Необходимо также обеспечить интерфейс для CRUD-операций над данными. В общем, необходимо избавиться от необходимости писать SQL-код для взаимодействия в СУБД

Одной из реализаций ORM является Entity Framework, которая также использована в работе.

ADO.NET Entity Framework (EF) — объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services (Astoria), так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования MVC, MVP или MVVM.

Изначально с самой первой версии Entity Framework поддерживал подход Database First, который позволял по готовой базе данных сгенерировать модель edmx. Затем эта модель использовалась для подключения к базе данных. Позже был добавлен подход Model First. Он позволял создать вручную с помощью визуального редактора модель edmx, и по ней создать базу данных. Начиная с 5.0 предпочтительным подходом становится Code First. Его суть - сначала пишется код модели на C#, а затем по нему генерируется база данных. При этом модель edmx уже не используется. В работе используется подход Code First.

Расскажем о реализации подключения к базе данных: в программе создан класс «Session» в папке DataModels, который хранит данные об ID сессии и времени ее начала. Он представлен на рисунке 3.8.

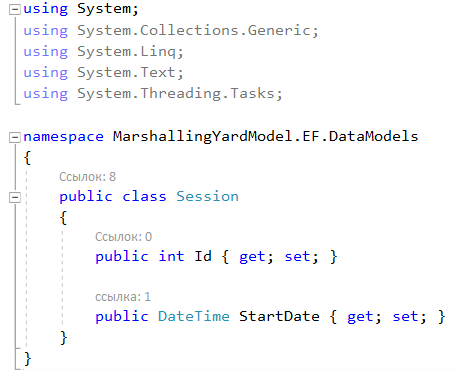


Рисунок 3.8 – Класс «Session» уровня данных

Каждая новая сессия создается в классе «MarshallingYard». Код метода для создания представлен на рисунке 3.9.

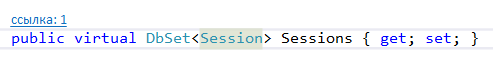


Рисунок 3.9 – Код метода для создания сессии

В программе использован подход Code First. Это значит, что нужные правки вносятся непосредственно в коде модели данных, а потом создается миграция и через нее обновляется структура базы данных. При разработке новых приложений с нуля вместе с базой данных этот подход более удобен и более гибкий, чем подход Database First.

Созданные миграции хранятся в папке «Migrations» и представлены на рисунке 3.10.

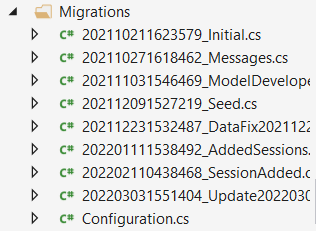


Рисунок 3.10 – Созданные миграции

Мы передаем данные на dto модель чтобы отделить конкретную структуру базы данных от тех структур, которыми бизнес обменивается с данными. Все это реализовано в репозитории. Абстракции репозиториев хранятся в папке Repositories.

Сборка в которой хранится само определение базы данных находится конкретная реализация репозиториев. Это класс EnvironmentRepository. Метод уровня базы данных для создания сессии представлен на рисунке 3.11. Мы задаем имя, создаем сессию и возвращаем результат. У нас строится SQL – запрос, в явном виде мы его здесь не указываем. Используется система ORM.

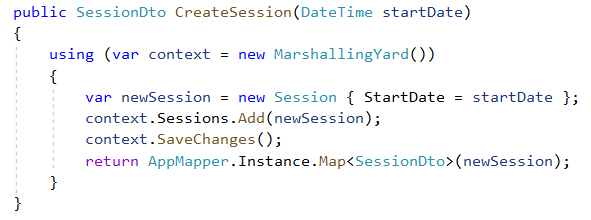


Рисунок 3.11 – Метод уровня базы данных для создания сессии

Самый внешний уровень – это уровень представления (View – модель). Затем идет уровень бизнеса (бизнес логика). Далее идет уровень данных (здесь это репозитории, которые непосредственно обращаются к базе данных). Получается, что в SQL уходят модели уровня данных (в программе - это папка DataModels). Каждый переход между уровнями – это свой уровень модели. Между уровнем представления и View – моделью это уровень классов View – моделей. Между View – моделью и бизнес логикой это бизнес классы. Между бизнес логикой и репозиториями это классы dto. А репозитории непосредственно обращаются к моделям уровня базы данных.

При этом «внешний мир» ничего не знает про класс Session. Знает только про SessionDto. Этим мы четко отделили бизнес уровень и уровень данных.

Далее мы поднимаемся на уровень бизнеса (папка BusinessModels). Здесь также содержится класс Session, идентичный прошлому по составу, но отличающийся по назначению.

Далее от уровня бизнеса мы поднимаемся к View – моделям. В папке ViewModels содержится класс «SessionViewModels», который представляет view – модель сессии, по структуре она также идентична.

Потом view – модель сессии мы включаем в общую view – модель (класс MainWindowViewModel, представлено на рисунке 3.12).



Рисунок 3.12 – Включение в общую view – модель

На рисунке 3.13 представлен код для изменения сессии

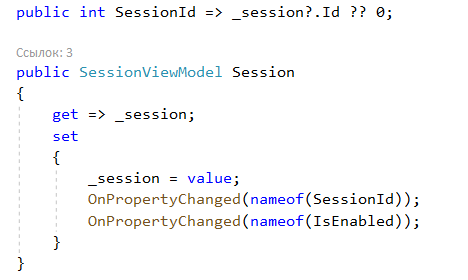


Рисунок 3.13 – Изменение сессии

На рисунке 3.14 представлен код для начала сессии. Это View – модель второго окна, в котором мы задаем параметры начала сессии.

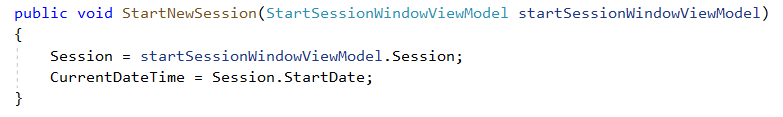


Рисунок 3.14 – Код для начала сессии

Здесь мы обращаемся к классу StartSessionWindowViewModel. На рисунке 3.15 представлен метод StartSession. Это также уровень View – модели второго окна. В нем мы создаем сессию, передавая это в бизнес сервис, который обращается к репозиторию и который создает, а потом по тому же пути возвращается обратно.

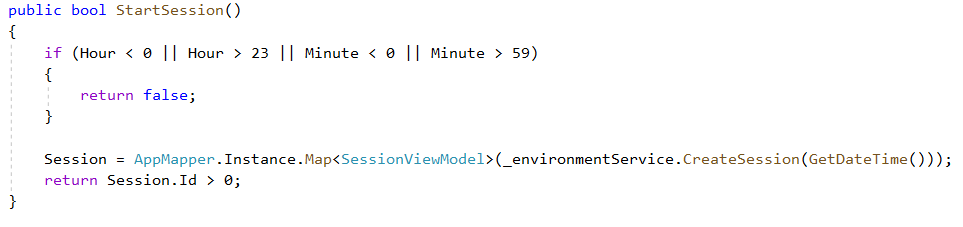


Рисунок 3.15 – Метод StartSession

Для реализации мультиагентности используется технология Dependency Injection/Inversion of Control (в создаваемом приложении используется реализация контейнера Ninject). Эта технология обеспечивает максимальную независимость отдельных модулей друг от друга. Фактически мы обращаемся не к объекту конкретного класса, а к объекту, реализующему интерфейс. Это позволяет нам при необходимости сделать другую реализацию репозиториев (например, можно начать работать не с Microsoft SQL Server, а с другой СУБД). Достаточно будет сделать другие реализации и прописать их, программа сразу будет обращаться к другой БД. Также и с сервисами. Если мы захотим, чтобы бизнес – сервис работал не напрямую, а, например, в сети. Тогда мы вместо физического класса сервиса можем прописать передающую структуру (прокси), она преобразует запросы к нашему сервису в запросы к web – серверу, получает от него результаты, далее преобразует в нужный нам бизнес – класс. Благодаря использованию технологии Dependency Injection/Inversion of Control нам будет достаточно написать другой модуль и заменить привязки.

Опишем реализацию мультиагентной технологии. Поведение агентов реализовано в классе симулятора. Он симулирует поведение иммунной системы, ориентирован на конкретную предметную область. Функции симулятора: генерация исходных данных (сообщений), имитация ошибок, проверка на наличие ошибок, анализ операций по исправлению и внесение предложений.

Проверка ошибок реализуется через набор правил. Каждое правило реализует свой интерфейс, при проверке на наличие ошибок у каждого правила срабатывает свой метод, который исследует данные и в случае фиксации ошибки выдает сообщение об ошибке и варианты исправления. То есть все правила работают независимо друг от друга. Они ничего друг про друга не знают.

Также в прототипе используется технология AutoMapper. При помощи него автоматизируется перевод данных между разными уровнями. Мы можем его настраивать и какие-то из свойств класса брать из иных источников, а не из свойств исходного класса. Какие-то свойства при этом можно просто игнорировать. При работе с данными (и не только) нам часто приходится сталкиваться с необходимостью копирования (маппинга) значений свойств одного объекта в новый объект другого типа. Automapper позволяет проецировать одну модель на другую, что позволяет сократить объемы кода и упростить программу.

Обучение системы производится методом выбора по большинству. Он заключается в том, что сначала мы накапливаем массу выбранных пользователем вариантов исправления. Если ошибка уже встретилась критическое число раз (в программе это 10), то программа уже получила 10 ответов от пользователя на вопрос «Какой из вариантов обработки ошибки нужно выбрать?». Далее она анализирует, какой вариант был выбран чаще и начинает сама его подставлять в поле для исправления возникшей ошибки. Но это не запрещает пользователю выбрать иной вариант для исправления ошибки при необходимости.

В данном разделе проведено сравнение сред разработки и языков программирования. Также выбрана среда для работы с базой данных. В результате для разрабатываемой информационной системы была выбрана среда разработки Visual Studio 2019, язык программирования C#, для хранения информации выбрана система управления реляционными базами данных Microsoft SQL Server.

Также была спроектирована и наполнена актуальной информацией БД для разрабатываемой ИС, описано подключение к базе данных. Еще в данном разделе описаны использованные для создания прототипа библиотеки и технологии.

# **4 Руководство пользователя**

Руководство пользователя – это документ, входящий в состав эксплуатационной документации на программное средство. Цель создания документа направлена на самостоятельное знакомство пользователя со всеми возможностями программного средства. Данный документ рассматривается пользователем как справочник, к которому можно обратиться в процессе работы, или же своего рода учебник, позволяющий познакомиться с программой теоретически, чтобы в дальнейшем грамотно использовать весь его функционал.

Созданный прототип иммунной интеллектуальной системы может применяться для обработки данных на сортировочной станции.

Разработанный прототип обрабатывает исходные данные, выполняет поиск ошибок, предлагает пользователю варианты исправления ошибок, накапливает статистику по исправленным ошибкам и умеет исправлять их самостоятельно в будущем. Исходные данные генерируется на основе обработанного методами статистики массива информации из АСУ СТ.

Пользователем является маневровый диспетчер. Пользователь должен быть обучен, без соответствующего документа об окончании обучения он не допускается к системе.

Интерфейс программы представляет собой таблицу с данными. При запуске программы нужно указать дату сеанса. После запуска начинают генерироваться данные о поступающих в обработку вагонах.

На входе данные генерируются программой в заданных ограничениях. Далее, в процессе прохождения операций, данные меняются и генерируются ошибки. При возникновении ошибки система посредством иммунного алгоритма обнаружения вторжений находит их и представляет пользователю возможные варианты действий.

В конце сеанса полученные данные отправляются и сохраняются в базу данных SQL Server.

Для работы нужно запустить приложение в среде Microsoft Visual Studio 2019. У системы один разработчик, работа может происходить на одном компьютере.

Интерфейс приложения состоит из нескольких окон. Опишем их все. При запуске приложения открывается главное окно. Изначально оно пустое так как еще не задан сеанс. Это представлено на рисунке 4.1.

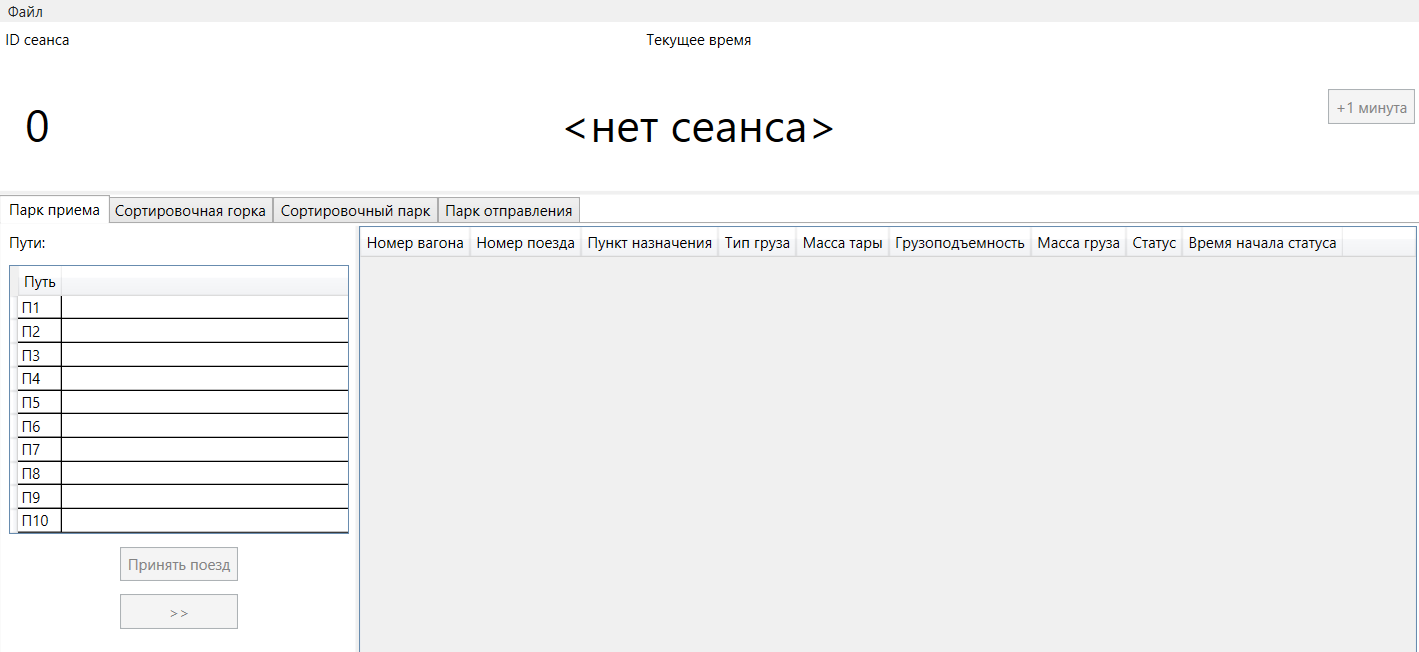


Рисунок 4.1 – Начальное окно приложения

Далее нужно нажать «Файл» - «Начать сеанс», откроется второе окно, в нем можно задать время и дату начала сеанса. По умолчанию задана системная дата (та, что на компьютере) и время 8 утра. Это представлено на рисунке 4.2.

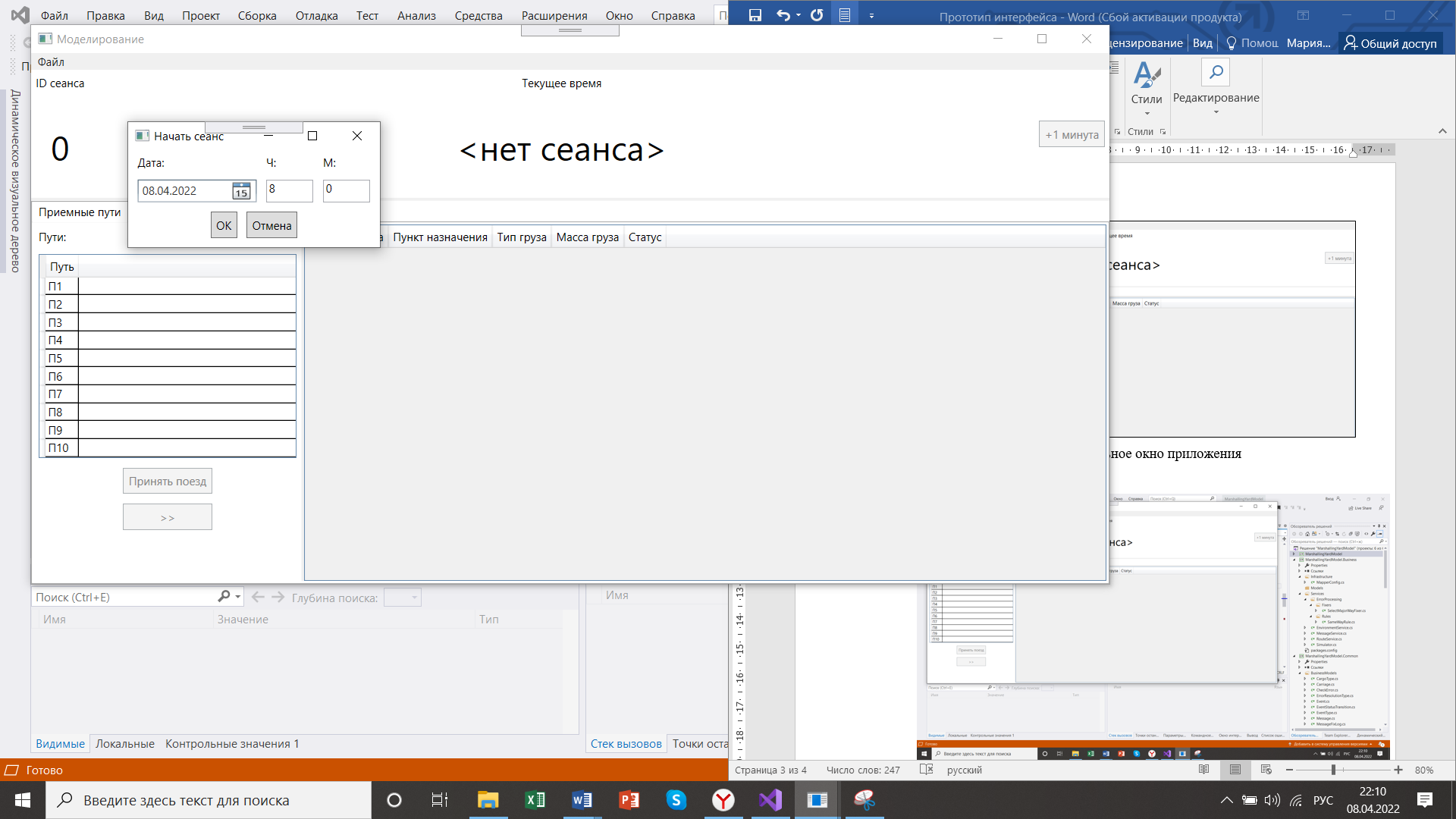
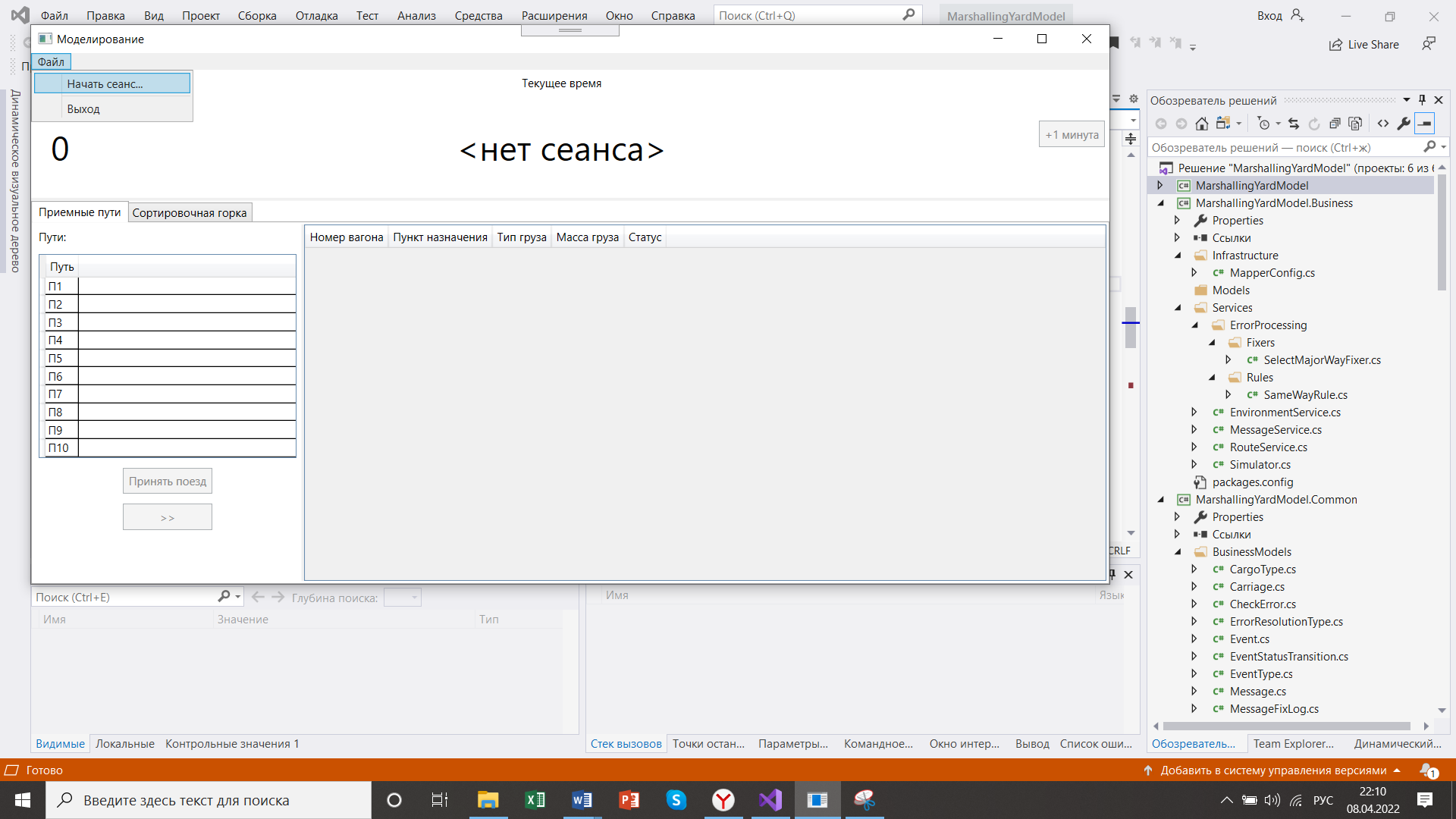


Рисунок 4.2 – Начало сеанса

После начала сеанса в главном окне отобразился номер сеанса, дата и время. Представлено на рисунке 4.3.

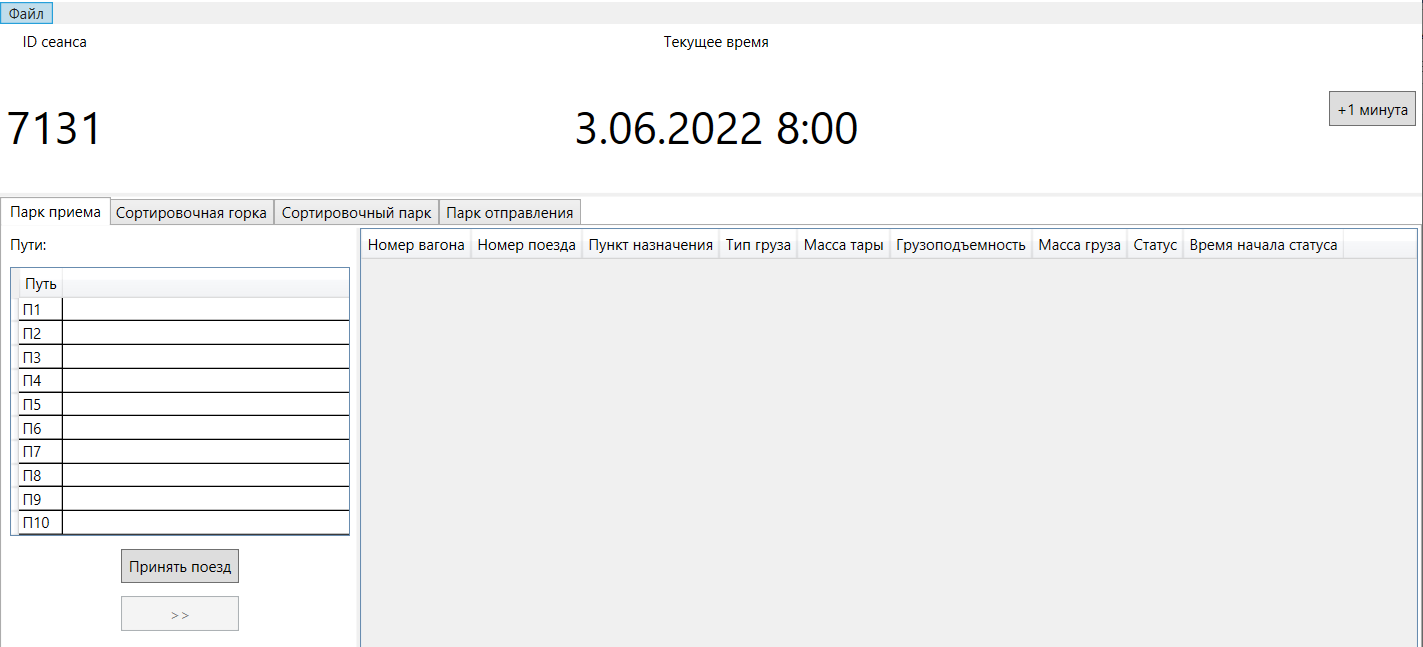


Рисунок 4.3 – Начатый сеанс

Выбираем путь, на который примем поезд и нажимаем «Принять поезд», представлено на рисунке 4.4.

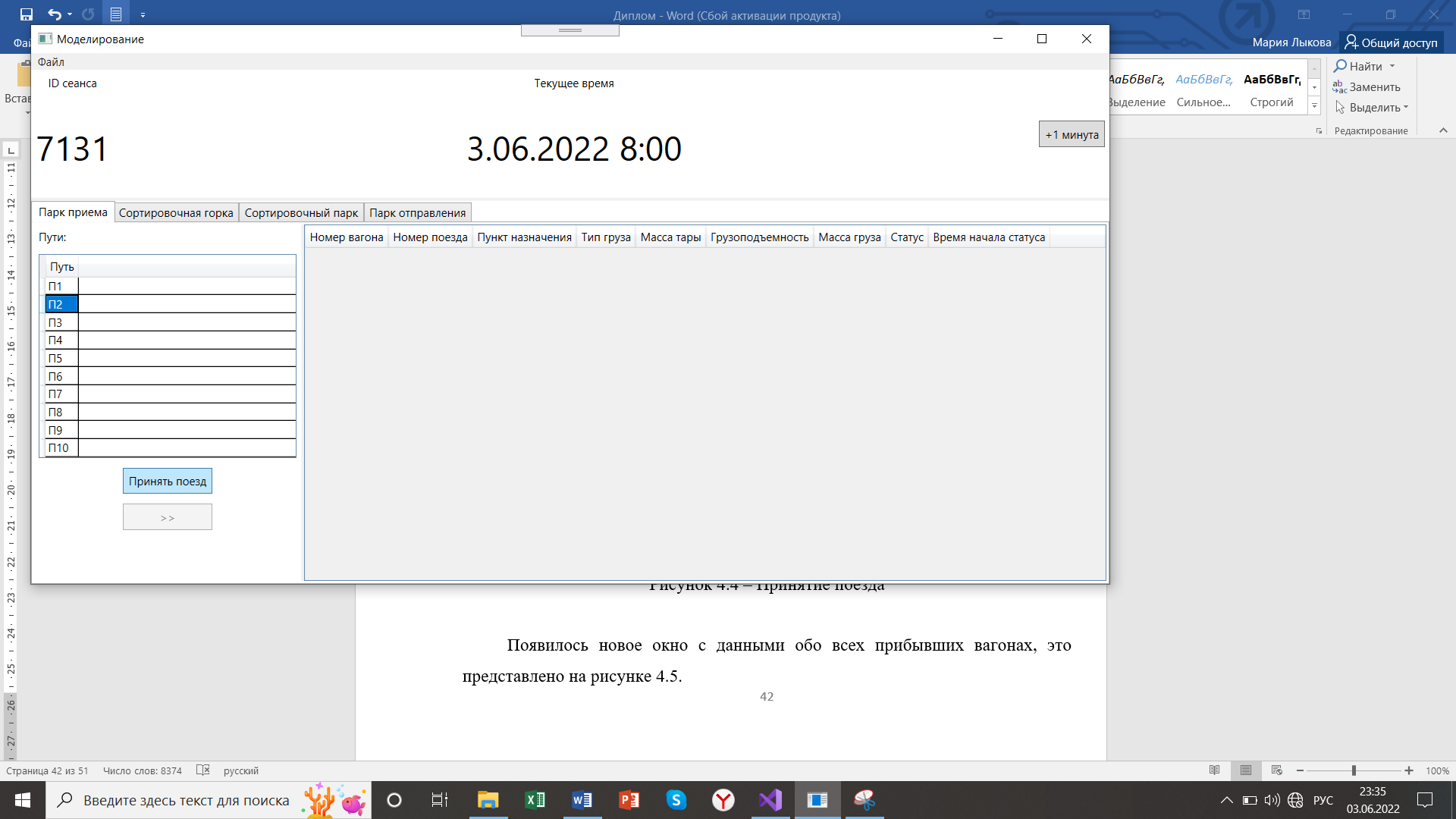


Рисунок 4.4 – Принятие поезда

Появилось новое окно с данными обо всех прибывших вагонах, это представлено на рисунке 4.5.

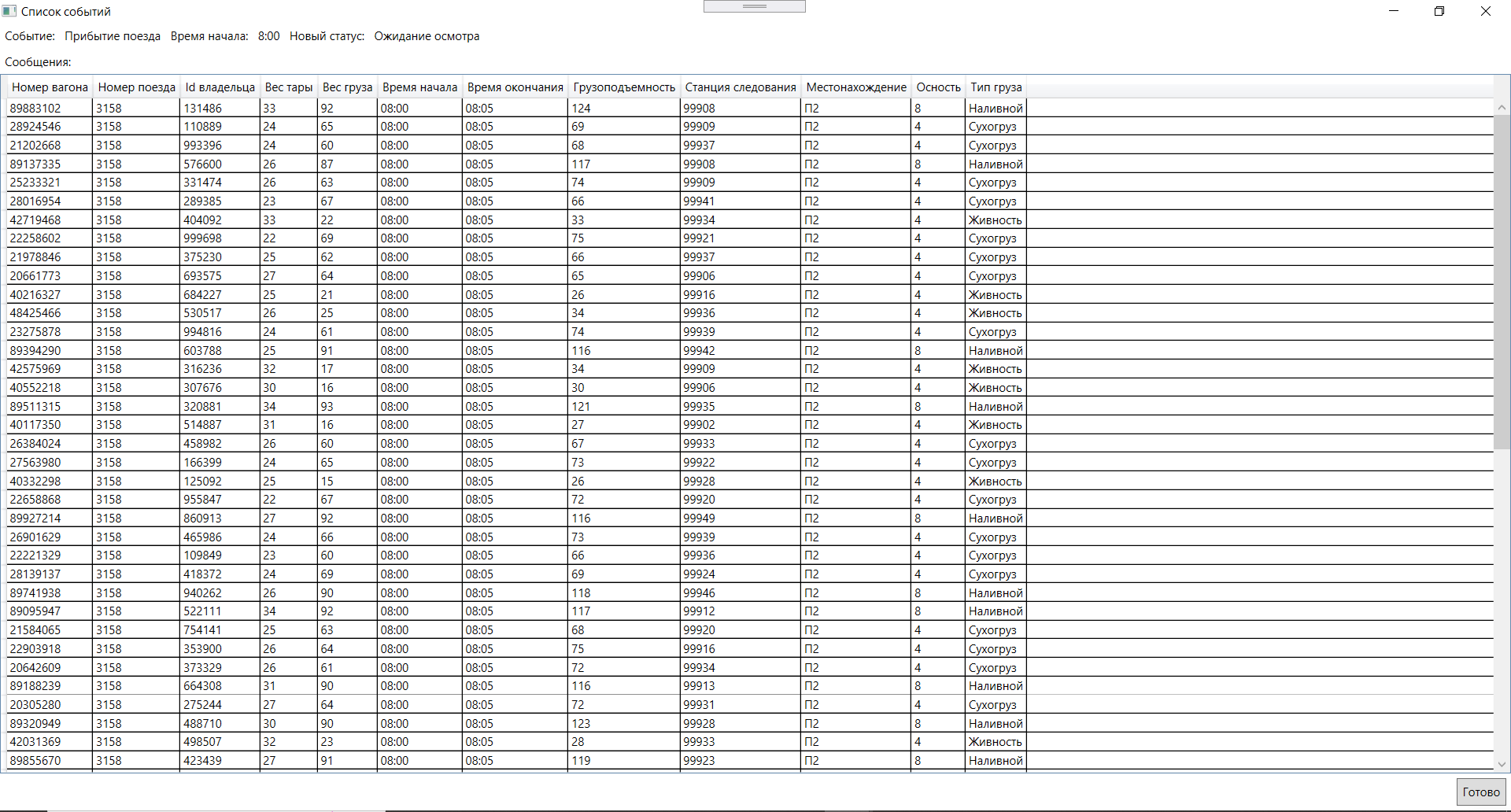


Рисунок 4.5 – Данные о прибывших вагонах

После нажатия на кнопку «Готово» данные перейдут в таблицу из главного окна, представлено на рисунке 4.6.

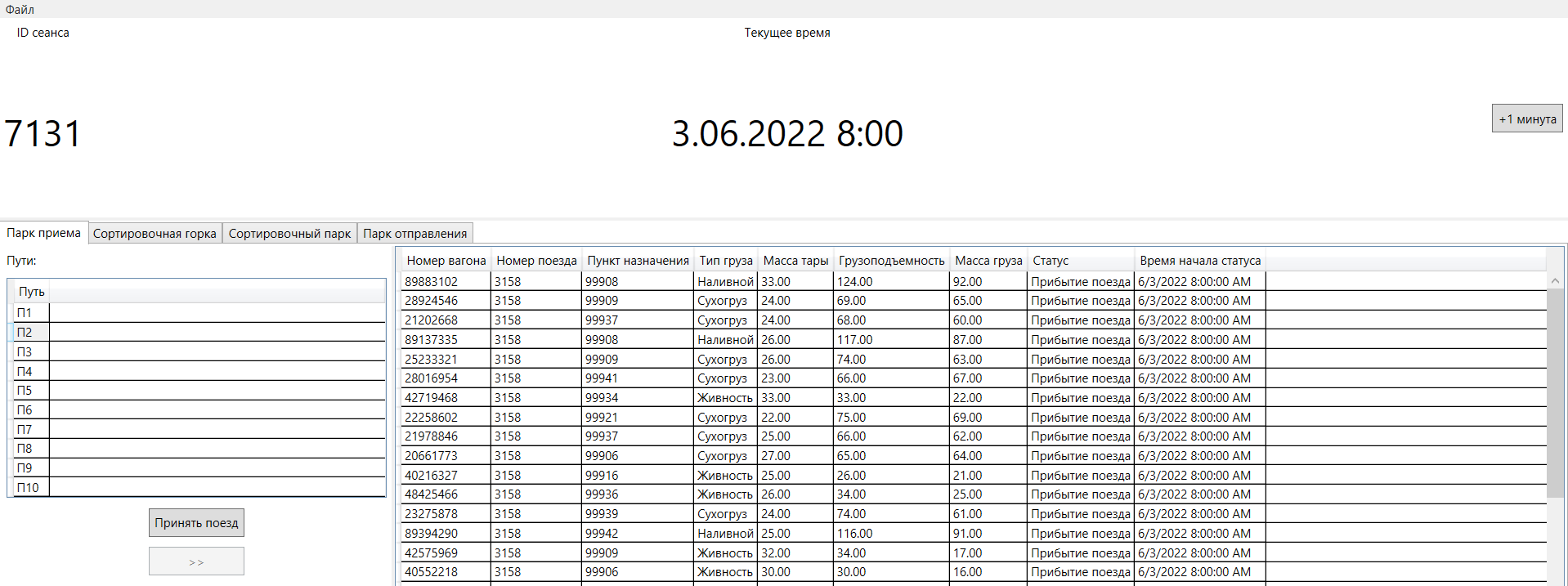


Рисунок 4.6 – Данные о прибывших вагонах в главном окне

Далее можно выбрать следующий путь и также принять поезд на него. Для перевода вагонов на следующее событие нужно нажать кнопку «>>». Кнопка станет активна только после того, как пройдет минимальное время прохождения операции. Для увеличения времени нужно нажимать на кнопку в верхнем правом углу «+1 минута».

При передаче данных могут возникать ошибки, программа сама их отслеживает и предлагает варианты решения. На рисунке 4.7 представлена ошибка в номере пути (местонахождение). Есть два варианта исправления: «Выбрать по большинству» или «Выбрать случайный путь». Так как система самообучающаяся, она сама подставила наиболее часто выбираемый вариант исправления, но его можно изменить.

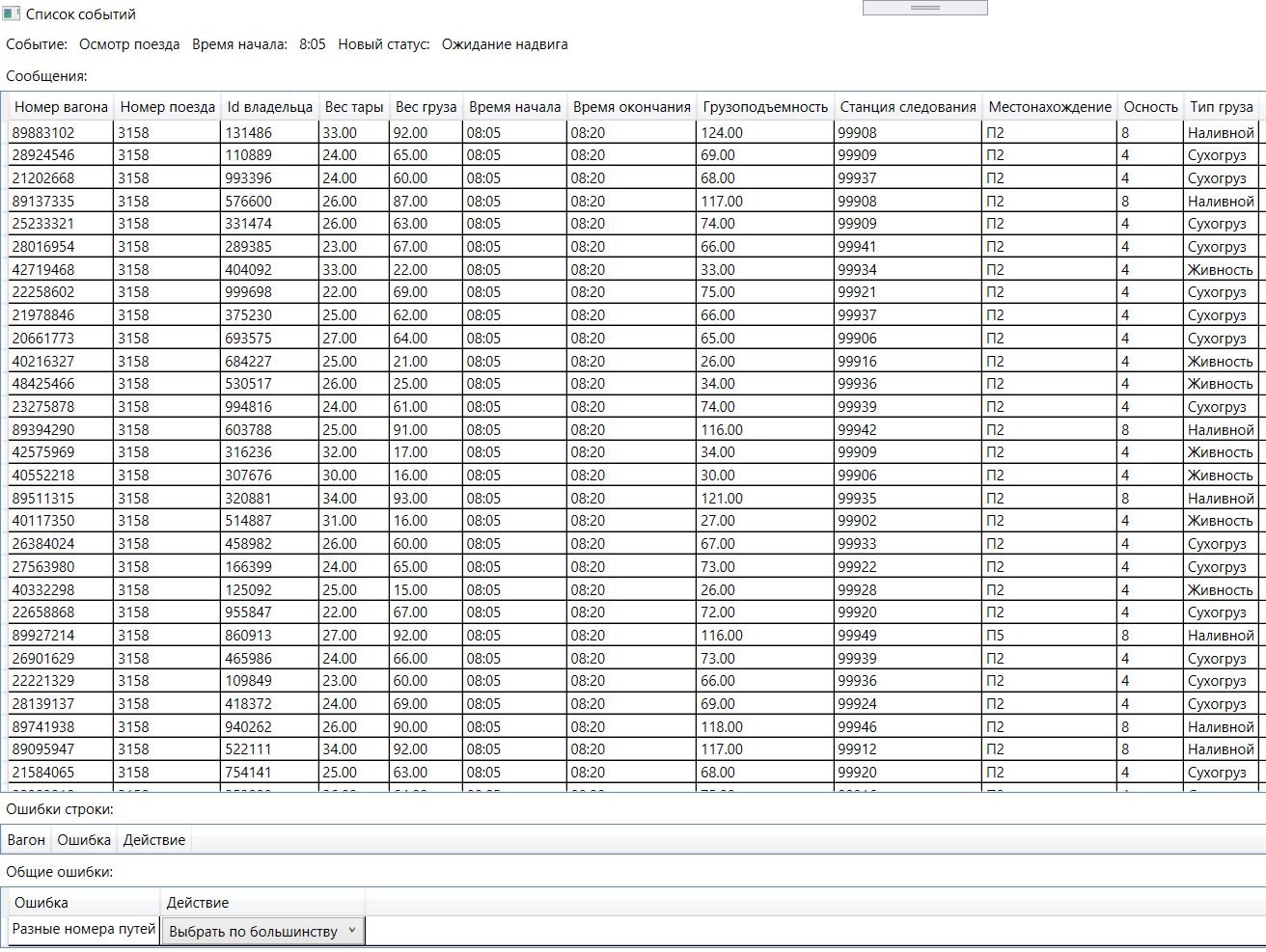


Рисунок 4.7 – Ошибка в столбце «Местонахождение»

На рисунке 4.8 представлено две ошибки – «Разные значения параметров вагона» и «Не задано время начала события». Первая означает, что один из параметров вагона пришел ошибочным, отличным от того, что было во время предыдущего события. Варианты исправления: «Исправить на ранее сохраненные данные», «Исправить данные о вагоне в БД». Вторая ошибка означает, что у одного вагона ошибочно не задано время начала события, здесь один вариант исправления – «Задать такое же, как у других».

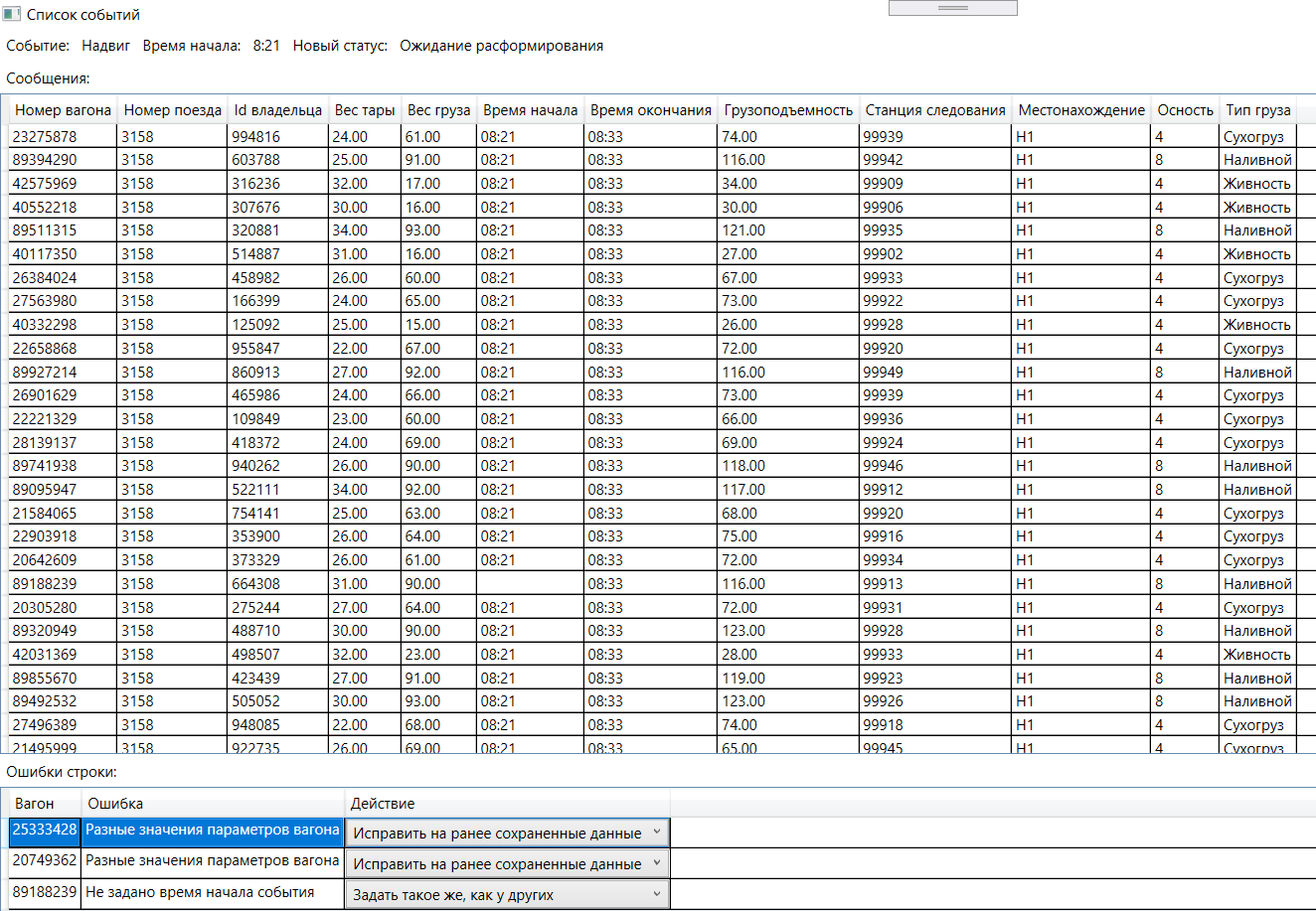


Рисунок 4.8 – Исправление ошибки

Еще одна ошибка возникнет, если пользователь попытается принять поезд на занятый путь. На рисунке 4.9 изображено, что путь номер 3 уже занят.

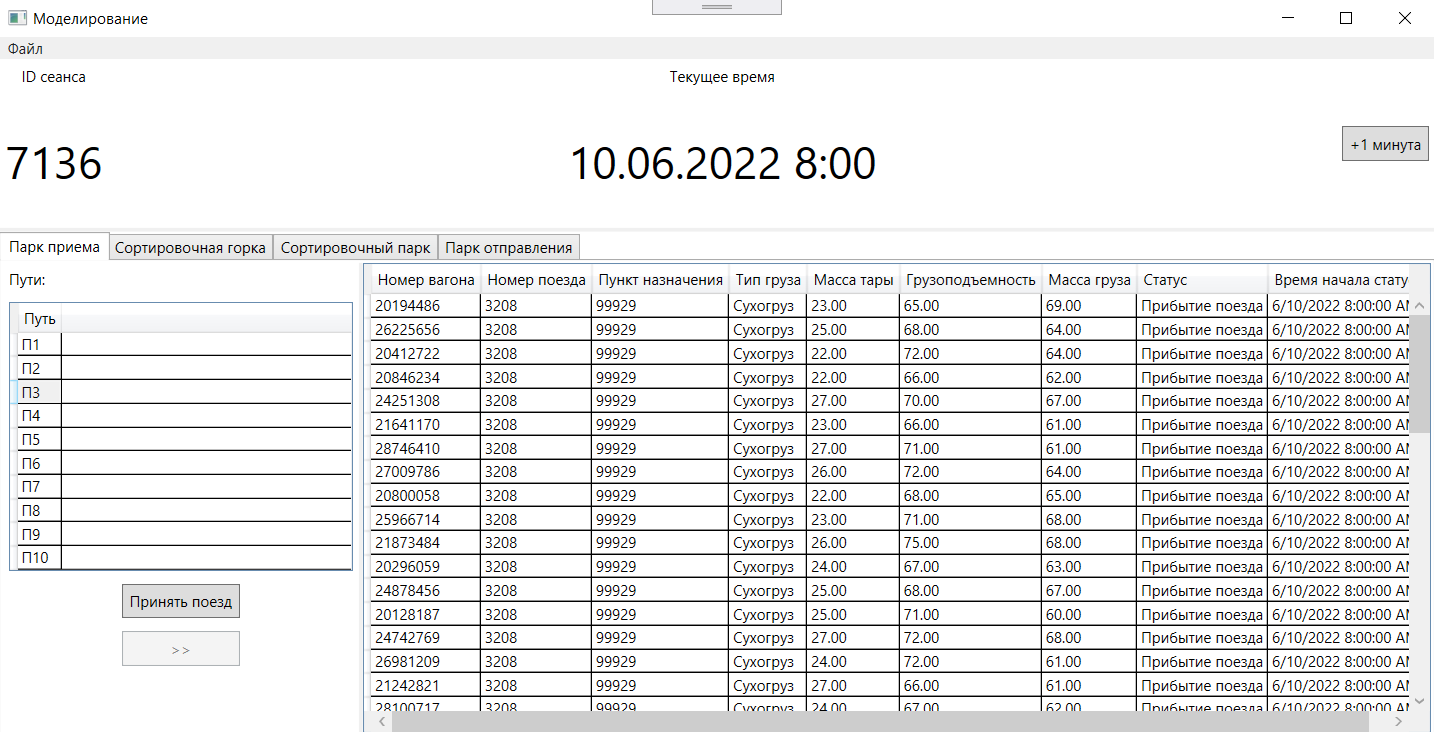


Рисунок 4.9 – Занятый путь

При попытке нажать на кнопку «Принять поезд», появится окно, представленное на рисунке 4.10.

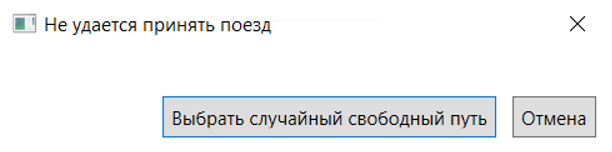


Рисунок 4.10 – Обработка ошибки «Нахождение двух объектов на одном пути одновременно»

Есть два варианта решения. При выборе первого варианта, новый поезд будет принят на любой другой свободный путь. При нажатии «Отмена» все останется как было, новый поезд принят не будет. Синим подсвечен тот вариант, который выбирался чаще.

По мере прохождения вагонами операций, могут возникать ошибки в данных. Система их отслеживает и предлагает варианты решения, самостоятельно подставляя тот, который чаще выбирался до этого. Таким образом поезд проходит парк приема, сортировочную горку, сортировочный парк и парк отправления.

Для изучения программы необходимо прочитать данное руководство пользователя.

# **ЗАКЛЮЧЕНИЕ**

В работе проведено исследование иммунных интеллектуальных систем и алгоритмов их работы. Исследованы бизнес-процессы работы сортировочной станции. Проведен анализ данных из АСУ СТ. При помощи нотации UML разработаны диаграммы бизнес-процессов. Проанализированы разные среды разработки и языки программирования, выбрана среда разработки Microsoft Visual Studio, язык программирования C#.

Используются следующие дополнительные библиотеки:

* EntityFramework (наиболее распространенное готовое решение. Набор технологий в ADO.NET, которые поддерживают разработку программных приложений, ориентированных на данные);
* AutoMapper (при помощи него автоматизируется перевод данных между разными уровнями, позволяет проецировать одну модель на другую, что помогает сократить объемы кода и упростить программу);
* Ninject (технология обеспечивает максимальную независимость отдельных модулей друг от друга. Фактически мы обращаемся не к объекту конкретного класса, а к объекту, реализующему интерфейс).

Выделены ошибки, которые возникают при работе с данными во время прохождения вагонами технологических операций на сортировочной станции. На основании технического задания, представленного в приложении А, разработан прототип самообучающейся иммунной интеллектуальной системы. Составлено руководство пользователя.

Созданный прототип:

* позволяет получить данные по поступающим на сортировочную станцию вагонам:
* исправляет ошибки, которые могут возникать при прохождении вагонов на сортировочной станции;
* является самообучающимся.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Summrina Kanwal, Amir Hussain, KaizhuHuang.: Novel Artificial Immune Networks-based optimization of shallow machine learning (ML) classifiers. Expert Systems with Applications Volume 165, 1 March 2021, 113834 (дата обращения: 02.04.2022).

1. Kephart, J. O. «A biologically inspired immune system for computers». Proceedings of Artificial Life IV: The Fourth International Workshop on the Synthesis and Simulation of Living Systems, MIT Press. (дата обращения: 02.03.2022).
2. Krens E.A., Molendijk L., Wullems G.I., Schilperoort R.A. In vitro Transformation of Plant Protoplasts with Ti-Plasmid DNA // (дата обращения: 25.03.2022).
3. Beck G., Habicht G. S. Immunity and the invertebrates. (англ.) // Scientific American. (vol. 275, no. 5). — doi:10.1038/scientificamerican1196-60. — PMID 8875808 (дата обращения: 10.03.2022).
4. Medzhitov R. Recognition of microorganisms and activation of the immune response. (англ.) // Nature. — 18 October (vol. 449, no. 7164). — doi:10.1038/nature06246. — PMID 17943118 (дата обращения: 08.03.2022).
5. Kevin Comeau, Pierre Paradis, Ernesto L.: Human and murine memory γδ T cells: Evidence for acquired immune memory in bacterial and viral infections and autoimmunity. Cellular Immunology Volume 357, November 2020, 104217 (дата обращения: 12.03.2022).
6. Y. Shoham, K. Leyton-Brown. Algorithmic, Game-Theoretic, and Logical Foundations. — London: Cambridge University Press. (дата обращения: 20.05.2022).
7. Michael Wooldridge, An Introduction to MultiAgent Systems, John Wiley & Sons Ltd, paperback, ISBN 0-471-49691-X (дата обращения: 12.05.2022).
8. Carl Hewitt and Jeff Inman. DAI Betwixt and Between: From «Intelligent Agents» to Open Systems Science IEEE Transactions on Systems, Man, and Cybernetics (дата обращения: 09.04.2022).
9. Jacques Ferber, Multi-Agent Systems: An Introduction to Artificial Intelligence, Addison-Wesley, ISBN 0-201-36048-9 (дата обращения: 10.03.2022).
10. Nataliya N. Bakhtadze, Igor B.Yadykin, Vladimir A. Lototsky, Evgeny M. Maximov, Ekaterina A. Sakrutina.: Multi-agent Approach to Design of Multimodal Intelligent Immune System for Smart Grid. (дата обращения: 27.04.2022).
11. S.V. Gladysh.: Immunocomputing in Information Security Incident Management (дата обращения: 29.04.2022).
12. Marcos Zuccolotto, Carlos Eduardo Pereira, Luca Fasanotti, Sergio Cavalieri, Jay Lee.: Designing an Artificial Immune Systems for Intelligent Maintenance Systems (дата обращения: 12.05.2022).
13. Сортировочная станция. [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Сортировочная\_станция#:~:text=Сортиро́вочная%20ста́нция%20—%20техническая%20железнодорожная,смены%20локомотивов%20и%20локомотивных%20бригад (Дата обращения: 10.05.2022).
14. de Castro, Leandro N.; Timmis, Jonathan. Artificial Immune Systems: A New Computational Intelligence Approach (англ.). — ISBN 1852335947, 9781852335946. (дата обращения: 02.03.2022).
15. Jungwon Kim, Peter J. Bentley, Uwe Aickelin, Julie Greensmith, Gianni Tedesco, Jamie Twycross: Immune System Approaches to Intrusion Detection - A Review, doi: 10.1007/s11047-006-9026-4, pp TBA (дата обращения: 02.03.2022).
16. Лучшие IDE для разработки на C#. [Электронный ресурс] URL: <https://gb.ru/posts/c_sharp_ides> (Дата обращения: 20.03.2022).
17. C# и Python - какой вариант лучше? [Электронный ресурс] URL: <https://devdelphi.ru/blog/c-protiv-python-kakoi-variant-luchshe> (Дата обращения: 22.03.2022).

# **ПРИЛОЖЕНИЕ А**

**Техническое задание**

Цель работы: Создать прототип иммунной интеллектуальной системы для обработки данных на сортировочной станции

Задачи работы:

Теоретические задачи:

* представить обзор иммунных алгоритмов;
* выделить бизнес-процессы работы сортировочной станции;
* определить набор обрабатываемых данных;

Аналитические задачи:

* разработать иммунный алгоритм обработки данных на сортировочной станции;

Проектные задачи:

* разработать прототип иммунной интеллектуальной системы для обработки данных на сортировочной станции;
* сделать анализ результатов.

Разрабатываемый прототип должен обрабатывать исходные данные, выполнять поиск ошибок, предлагать пользователю варианты исправления ошибок, накапливать статистику по исправленным ошибкам и учиться исправлять их самостоятельно в будущем. Исходные данные генерируется на основе обработанного методами статистики массива информации из АСУ СТ.

Опишем бизнес – процесс работы сортировочной станции. Станция делится на три парка: парк приема, сортировочный парк и парк отправления.

В процессе работы на каждой операции мы оперируем следующими данными: номер вагона, номер поезда, вагоновладелец, вес тары, вес груза, время начала прохождения каждой операции, грузоподъемность вагона, станция следования вагона, местонахождение (путь), осность вагона.

В рамках бизнес – процесса парка приема можно выделить основные подпроцессы:

* прибытие поезда;
* ожидание осмотра;
* осмотр;
* завершение обработки поезда по прибытию;
* ожидание надвига.

В рамках подпроцесса «Прибытие поезда» мы получаем данные о прибывших вагонах в составе поезда: номер вагона, номер поезда, вагоновладелец, вес тары, вес груза, время начала прохождения каждой операции, грузоподъемность вагона, станция следования вагона, местонахождение (путь), осность вагона.

В процессе анализа бизнес – процессов и перемещения объекта по подпроцессам изменяются только данные о времени начала прохождения каждой операции, номере поезда и о местонахождении вагона.

В рамках бизнес – процесса сортировочного парка можно выделить основные подпроцессы:

* надвиг;
* расформирование;
* нахождение в сортировочном парке;
* ожидание выставления поезда в парк отправления.

В рамках бизнес – процесса парка отправления можно выделить основные подпроцессы:

* выставление поезда в парк отправления;
* ожидание локомотива;
* отправление.

В данных, которые описаны в каждом подпроцессе могут возникать ошибки. Иммунная интеллектуальная система будет находить ошибку и предлагать пользователю пути ее исправления. Система также будет являться самообучающейся и со временем начнет сама исправлять ошибки, проанализировав и запомнив ответы пользователя в прошлых инцидентах.

Интерфейс программы будет представлять собой таблицу с данными. При запуске программы нужно будет указать дату сеанса. После запуска начнут генерироваться данные о поступающих в обработку вагонах.

На входе данные генерируются программой в заданных ограничениях. Далее, в процессе прохождения операций, данные меняются и генерируются ошибки. При возникновении ошибки система посредством иммунного алгоритма обнаружения вторжений находит их и представляет пользователю возможные варианты действий (оставить как есть или заменить значение).

В конце сеанса полученные данные отправляются и сохраняются в базу данных SQL Server.

Пользователем является маневровый диспетчер. Его функционал описан выше. Также есть администратор, он же разработчик. Отвечает за корректную работу системы и исправляет возможные ошибки.

У системы один разработчик, работа может происходить на одном компьютере. Проект будет разработан в Microsoft Visual Studio 2019. Время на выполнение проекта составит 4 месяца. Конечная дата сдачи проекта – 01.05.2022.

Пользователь должен быть обучен, без соответствующего документа об окончании обучения он не допускается к системе.

Атрибуты качества: надежность, качество, доступность.

Внешние интерфейсы: взаимодействие с АСУ СТ.

Стадии и этапы разработки: сбор исходных данных, разработка, выбор программной среды, функциональное тестирование, регрессионное тестирование, загрузочное тестирование, написание пояснительной записки.

Работа принимается руководителем ВКР Улановым Алексеем Александровичем.