Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Информационная система «Табло авиарейсов»

Преподаватель \_\_\_\_\_\_\_\_\_\_ В. С. Васильев

подпись, дата

Студент КИ19-06Б, 031940348 \_\_\_\_\_\_\_\_\_\_ Е. А. Маслов

подпись, дата

Красноярск 2020

СОДЕРЖАНИЕ

1. Разработка спецификации требований………………………………………..3

1.1. Описание входных и выходных данных 3

1.2 Требования к составу выполняемых функций……………………………....4

1.3 Макет интерфейса…………………………………………………………….4

1.4 Стадии и этапы разработки 13

2. Проектирование 14

2.1 Список идентификаторов и их значения……………………………………15

2.2 Диаграммы пригодности 14

2.3 Диаграммы последовательности 20

2.4 Диаграмма классов 25

3. Программирование……………………………………………………………28

3.1 Реализация программы 28

3.2 Рефакторинг…………………………………………………………………..28

3.3 Документирование 28

4. Заключение……………………………………………………………………..29

Список использованных источников……………………………………………..29

# **1. Разработка спецификации требований**

Информационная система «Табло авиарейсов».

Предназначена для автоматизации работы с пассажирами, позволяет пассажирам просматривать, а также фильтровать рейсы на информационном табло, позволяет диспетчерам вносить изменения на табло авиарейсов (добавлять новые рейсы, удалять и изменять), предупреждать пассажиров о каком либо событии связанным с рейсами (рейс отложен и т.п.).

Разработка информационной системы ведётся в рамках дисциплины “Основы проектирования программного обеспечения”

Согласно учебному плану для получения оценки на экзамене.

Разрабатываемая система поддержки информационного табло, должна обеспечивать следующие функции:

1. Отображение на табло информации о рейсах:
   1. Номер рейса(number).
   2. Пункт вылета(departure).
   3. Время вылета(deparure\_time).
   4. Пункт прилёта(arrival).
   5. Время прилёта(arrival\_time).

1.6. Статус (отложен, вылетел, прилетел).

1. Изменение в состояние табло (информация о задержке рейса и т. д).
2. Ввод и хранение данных о пассажирах:

3.1. Логин(login).

3.2. Пароль(password).

4.Ввод и хранение информации о диспетчерах:

4.1 Логин(login).

4.2 Пароль(password).

С данной информационной системой должны работать следующие группы пользователей:

1. Пассажир(passenger).
2. Диспетчер(cashier).
3. Администратор(admin).

При работе с системой пассажир должен иметь возможность:

1. Просматривать информацию о рейсах.
2. Фильтровать информацию на табло.

При работе с системой диспетчер должен иметь возможность:

1. Вносить изменения в состояние табло, если произошло некоторое событие (например, вылет рейса отложен на N минут, произошла посадка самолета указанного рейса и т. д).

При работе с системой администратор должен иметь возможность:

1. Вносить изменения о пользователях.
2. Вносить изменения в состояние табло, если произошло некоторое событие (например, вылет рейса отложен на N минут, произошла посадка самолета указанного рейса и т. д).

## **1.1. Описание входных и выходных данных**

**Входные данные**:

Файл: **users.bin**.

Формат файла: двоичный файл.

Описание: В файле **users.bin** хранятся данные о пользователях.

(Логин, пароль, статус.)

Типы данных:

Login – тип QString, может содержать числа и буквы.

password - тип QString, может содержать числа и буквы.

При попытке войти в систему, эти данные загружаются и сверяются с введёнными пользователем.

Файл: **planes.bin**.

Формат файла: двоичный файл.

Описание: В файле **planes.bin** хранятся данные о рейсах.

(Номер рейса, Пункт вылета, Время вылета, Пункт прилёта, Время прилёта, Статус (отложен, вылетел, прилетел))

Типы данных:

number – QString.

Departure – QString.

Departure\_time – QString.

Arrival – QString.

Arrival\_time – QString.

Status - QString.

В переменных Departure\_time и Arrival\_time хранятся данные о дате и времени вылета, а так же о дате и вылете прилёта. Эти данные хранятся в формате “ДД.ММ.ГГГГ – ЧЧ.ММ”.

Эти данные диспетчер вводит сам.

При открытии главного окна пользователя, эти данные загружаются и выводятся на экран в виде списка рейсов, в таком же формате, как и хранятся в файле.

**Выходные данные**:

Файл: **users.bin**.

Формат файла: двоичный файл.

Описание: В файле **users.bin** хранятся данные о пользователях.

(Логин, пароль.)

Типы данных:

Login – тип QString, может содержать числа и буквы.

password - тип QString, может содержать числа и буквы.

В файл вносятся изменения о пользователях.

Изменять, добавлять, удалять данные в этом файле имеет возможность только администратор.

Файл: **planes.bin**.

Формат файла: двоичный файл.

Описание: В файле **planes.bin** хранятся данные о рейсах.

(Номер рейса, Пункт вылета, Время вылета, Пункт прилёта, Время прилёта, Статус (отложен, вылетел, прилетел))

Типы данных:

number – QString.

Departure – QString.

Departure\_time – QString.

Arrival – QString.

Arrival\_time – QString.

Status - QString.

В переменных Departure\_time и Arrival\_time хранятся данные о дате и времени вылета, а так же о дате и вылете прилёта. Эти данные хранятся в формате “ДД.ММ.ГГГГ – ЧЧ.ММ”.

Эти данные диспетчер вводит сам.

При изменении рейса, удалении рейса и добавления нового рейса, данные в файле обновляются.

Вносить изменения в файл могут диспетчер и администратор.

## **1.2 Требования к составу выполняемых функций**

На рисунке 1 представлена диаграмма прецедентов администратора и диспетчера.

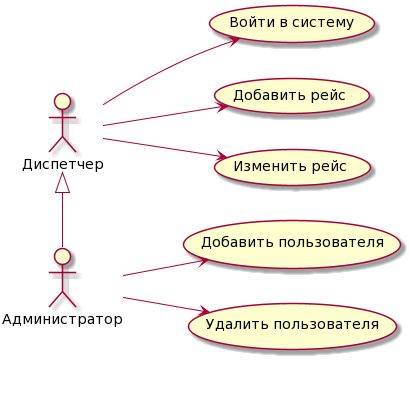


Рисунок 1 – Диаграмма прецедентов администратора и диспетчера

**Название прецедента:**Войти в систему

**Действующее лицо:**Диспетчер

**Цель:**Войти в систему

**Предусловия:**У диспетчера должны быть данные для входа (логин, пароль)

**Главная последовательность:**

1. Диспетчер запускает программу.
2. Диспетчер вводит данные для входа и нажимает кнопку “Войти”. Если данные введённые пользователем корректны, открывается главное окно приложения.

Иначе появляется ошибка о некорректном вводе.

**Название прецедента:**Добавить рейс

**Действующее лицо:**Диспетчер

**Цель:**Добавить новый рейс

**Предусловия:**Диспетчер осуществил вход в систему

**Главная последовательность:**

1. Диспетчер в главном окне переходит в меню диспетчера.
2. Открывается меню диспетчера, диспетчер нажимает кнопку “ Управление рейсами”, после чего переходит в окно управления рейсами.
3. Диспетчер нажимает кнопку “Добавить”.
4. Открывается окно добавления нового рейса.
5. Диспетчер вносит необходимые данные и нажимает

“ Сохранить”.

1. После этого открывается окно управления рейсами.

**Название прецедента:**Внести изменения о рейсах

**Действующее лицо:**Диспетчер

**Цель:** Изменение информации о рейсе

**Предусловия:**Диспетчер осуществил вход в систему

**Главная последовательность:**

1. Диспетчер в главном окне переходит в меню диспетчера.
2. Открывается меню диспетчера, диспетчер нажимает кнопку “ Управление рейсами”, после чего переходит в окно управления рейсами.
3. Диспетчер выбирает конкретный рейс и нажимает кнопку “Редактировать”.
4. Открывается окно редактирования рейса.
5. Диспетчер вносит необходимые поправки в рейс, данные о задержке рейса, вылете рейса и т.д. диспетчер записывает в поле “Статус” и нажимает

“ Сохранить”.

1. После этого открывается окно изменения рейсов.

**Название прецедента:**Добавить пользователя

**Действующее лицо:**Администратор

**Цель:** Добавить нового пользователя

**Предусловия:**Администратор осуществил вход в систему

**Главная последовательность:**

1. Администратор в главном окне нажимает кнопку “Администрирование”.
2. Открывается окно администратора.
3. Администратор нажимает кнопку “Управление пользователями”.
4. Открывается окно со списком всех пользователей.
5. Администратор нажимает кнопку “Добавить”.
6. Открывается окно добавления нового пользователя, в котором администратор вносит необходимые данные, а так же, выбирает статус пользователя(Диспетчер, пассажир). После этого нажимает кнопку “Сохранить”.
7. Открывается окно управления пользователями.

**Название прецедента:**Удалить пользователя

**Действующее лицо:**Администратор

**Цель:** Удалить пользователя

**Предусловия:**Администратор осуществил вход в систему

**Главная последовательность:**

1. Администратор в главном окне нажимает кнопку “Администрирование”.
2. Открывается окно администратора.
3. Администратор нажимает кнопку “Управление пользователями”.
4. Открывается окно со списком всех пользователей.
5. Администратор выделяет пользователя одним нажатием по нему и нажимает кнопку “Удалить”. После чего пользователь удаляется.
6. После открывается окно со списком всех пользователей.

На рисунке 2 представлена диаграмма прецедентов пассажира.

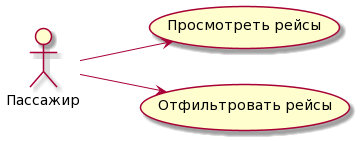


Рисунок 2 – Диаграмма прецедентов пассажира

**Название прецедента:**Просмотреть рейсы

**Действующее лицо:**Пассажир

**Цель:** Посмотреть на список рейсов с целью узнать интересующую информацию

**Предусловия:**-

**Главная последовательность:**

1. Пассажир в главном окне просматривает список рейсов.

**Название прецедента:**Отфильтровать рейсы

**Действующее лицо:**Пассажир

**Цель:** Отфильтровать список рейсов по интересующему параметру.

**Предусловия:**-

**Главная последовательность:**

1. Пассажир в главном окне просматривает список рейсов.
2. В текстовое поле пассажир вводит какой либо параметр, по которому он хочет отфильтровать рейсы.
3. После чего на главном окне выводиться отфильтрованный список рейсов.

**1.3 Макет интерфейса**

Пользовательский интерфейс программы должен содержать:

1. Окно авторизации при запуске программы, а так же окно регистрации, в случае если нет аккаунта в системе, представленные на рисунках 3 и 4 соответственно.

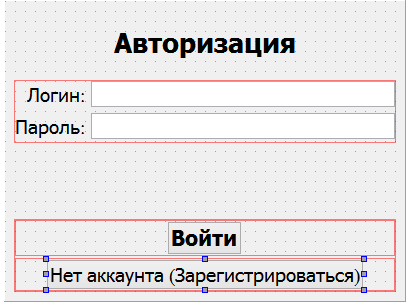
****

Рисунок 3 – Окно авторизации

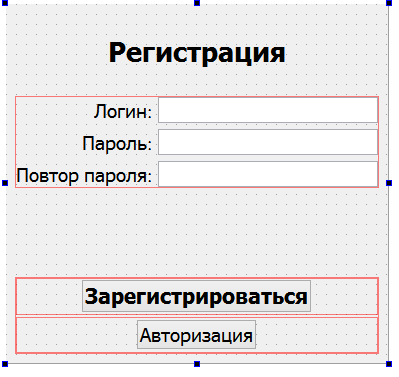
****

Рисунок 4 – Окно регистрации

2. Главное окно со списком рейсов для всех пользователей показано на рисунке 5, у администратора есть кнопка PushButton (Администрирование) в правом верхнем углу, а у других пользователей данная кнопка скрыта.

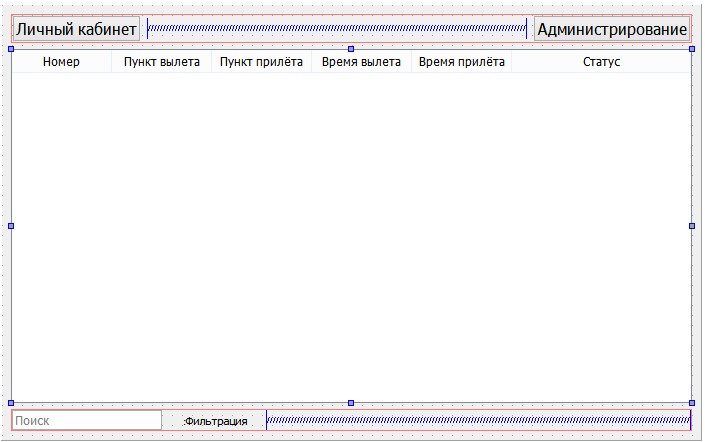
****

Рисунок 5 – Главное окно со списком рейсов

3. Меню администратора открывается после нажатия на кнопку – Администрирование, окно представлено на рисунке 6.

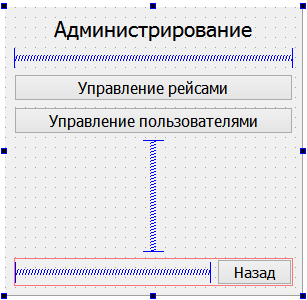
****

Рисунок 6 – Меню администратора

4. Меню диспетчера открывается после двойного клика по любому рейсу, окно показано на рисунке 7.

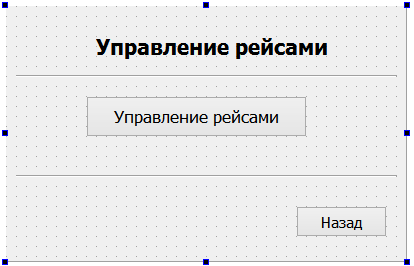
****

Рисунок 7 – Меню диспетчера

5. Как из меню администратора, так и из меню диспетчера можно перейти в окно управления рейсами, окно управления показано на рисунке 8.

В окне управления рейсами можно изменять рейсы (Нужно выделить нужный рейс и нажать кнопку «Редактировать»), так же можно удалять рейсы (Нужно выделить нужный рейс и нажать на кнопку «Удалить»), можно добавлять рейсы (Нужно нажать на кнопку «Добавить»). Окно редактирования/добавления рейса показано на рисунке 9.

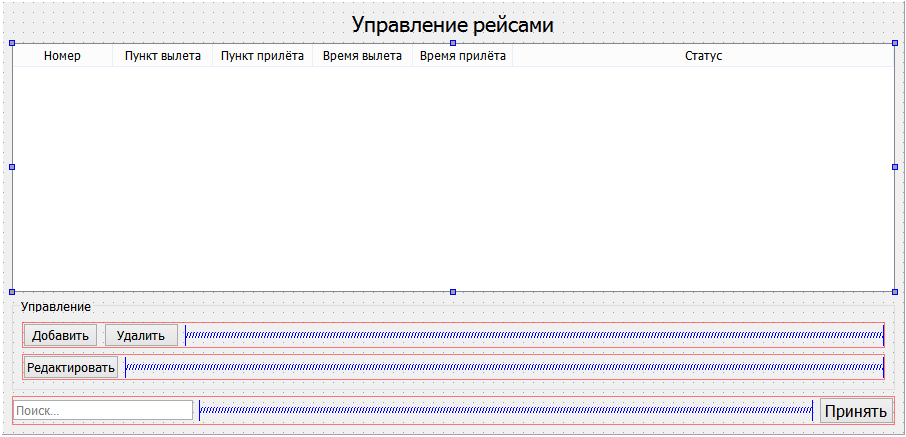
****

Рисунок 8 – Окно управления рейсами

****

Рисунок 9 – Окно редактирования/добавления рейса

6. В окно управления пользователями может попасть только администратор, это окно показано на рисунке 10.

В этом окне отображается список всех пользователей системы, администратор может удалить (Нужно выделить рейс и нажать на кнопку «Удалить») или добавить пользователя (Нужно нажать на кнопку «Добавить», после чего откроется окно добавления пользователя как показано на рисунке 11).

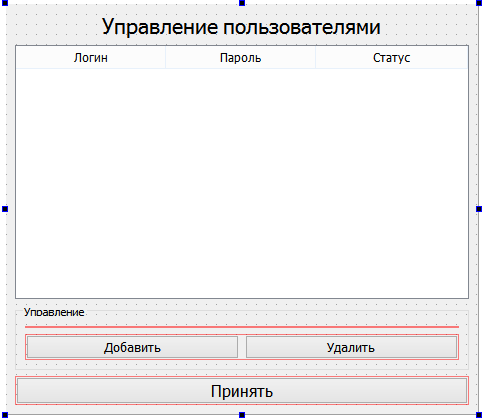
****

Рисунок 10 – Окно управления пользователями

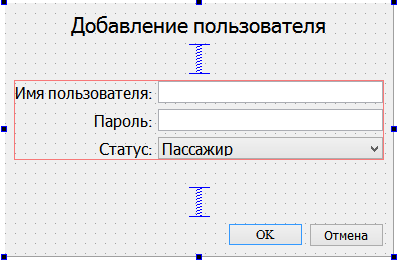
****

Рисунок 11 – Окно добавления пользователя

7. Окно личного кабинета содержит информацию о текущем пользователе который работает с системой, доступ к этому окну имеют все пользователи не зависимо от статуса, окно показано на рисунке 12.

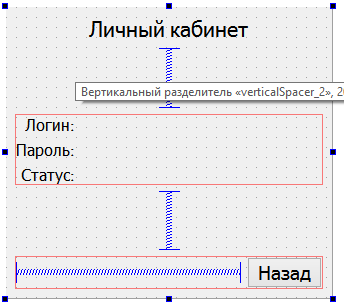
****

Рисунок 12 – Окно личного кабинета

## **1.4 Стадии и этапы разработки**

Этапы разработки и сроки приведены в таблице 1

Таблица 1 – Этапы разработки

|  |  |
| --- | --- |
| **Этап разработки** | **Сроки** |
| Создание графического интерфейса и реализация списка пользователей. | 15.10 – 22.10 |
| Написание классов «WorkWidget» , «MainWidget», «Plane», «User», объекты которых будут созданы для рейсов, пользователей, а также взаимодействия пользователей с системой. | 22.10 – 19.11 |
| Написание графических классов и  взаимодействия слотов и сигналов. | 19.11 – 28.11 |
| Написание пояснительной записки. | * 1. – 3.12 |

## **Проектирование**

Этап проектирования предполагает выполнения декомпозиции путём построения различных диаграмм.

Для информационной системы «Студенческое общежитие» были по порядку построены следующие диаграммы:

* 1. Диаграммы прецедентов(показаны выше);
  2. Диаграммы пригодности;
  3. Диаграммы последовательности;
  4. Диаграмма классов.

**2.1 Список идентификаторов и их значения**

**User** – Класс, созданный для описания пользователя, содержит в себе три параметра, а также функции для получения этих параметров и установки для определенного объекта, для реализации инкапсуляции.

**Plane** – Класс, предназначенный для описания рейсов, которые будут храниться в файле filePlanes = "planes.bin", и, соответственно, отображаться в таблице tableRaces класса QTableView. Содержит в себе 6 параметров (QString number, const QString departure, const QString arrival, const QString departure\_time, const QString arrival\_time, const QString status) а также функции для получения этих параметров (QString number(), QString departure(), QString arrival(), QString departure\_time(), QString arrival\_time(),QString status()) и функции для установки этих параметров (

void setNumber(const QString &number), void setDeparture(const QString &departure),void setArrival(const QString &arrival), void setDeparture\_time(const QString &departure), void setArrival\_time(const QString &arrival), void setStatus(const QString&status))

**AddRace** – Класс графического типа, наследуемый от класса QDialog,

в котором написана основная логика добавления нового рейса в таблицу, а так же редактирования рейса.

**AddUser** – Класс графического типа, наследуемый от класса QDialog, в котором написана основная логика добавления нового пользователя в таблицу.

**InfoUser** - Класс графического типа, наследуемый от класса QDialog, который служит для показа данных пользователю (login, password, status).

**ControlRaces**– Класс графического типа, наследуемый от класса QDialog,

в котором написана основная логика работы с рейсами, то есть добавление и удаление.

С помощью функций void on\_buttonAddRace\_clicked(), void on\_buttonDeleteRace\_clicked().

**ControlUsers**– Класс графического типа, наследуемый от класса

QDialog, в котором написана основная логика работы с пользователями. А именно удаление с помощью функции void on\_buttonDeleteUser\_clicked().

**SignIn**– Класс графического типа, наследуемый от класса

QWidget, в котором написана основная логика регистрации.

**SignUp**– Класс графического типа, наследуемый от класса

QWidget, в котором написана основная логика авторизации.

**MenuAdmin** - Класс графического типа, наследуемый от класса

QDialog, в котором с помощью слотов и сигналов осуществляется выбор администратора, либо перейти в окно изменения рейсов, либо перейти в окно управления пользователями void on\_buttonControlUsers\_clicked(), void on\_buttonControlRaces\_clicked().

**MenuCashier** - Класс графического типа, наследуемый от класса

QDialog, с помощью которого осуществляется переход в окно управления рейсами

## **2.2 Диаграммы пригодности**

На рисунке 13 показана диаграмма пригодности прецедента “Войти в систему” для администратора и диспетчера, а на рисунке 14 показана диаграмма пригодности “Войти в систему” для пассажира.

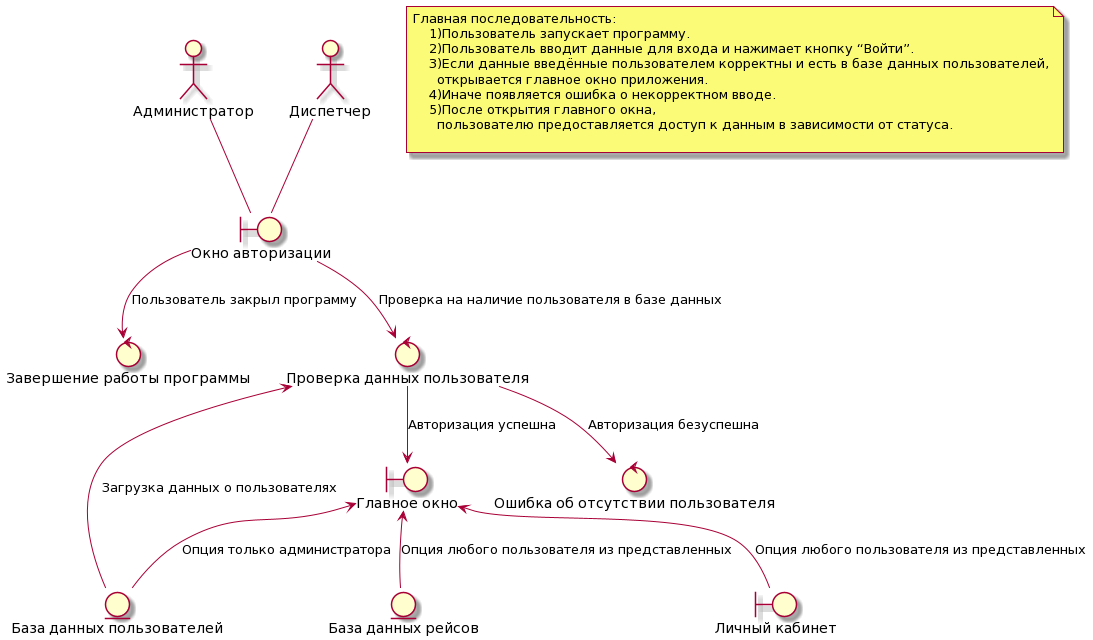


Рисунок 13 – Диаграмма пригодности прецедента “Войти в систему”

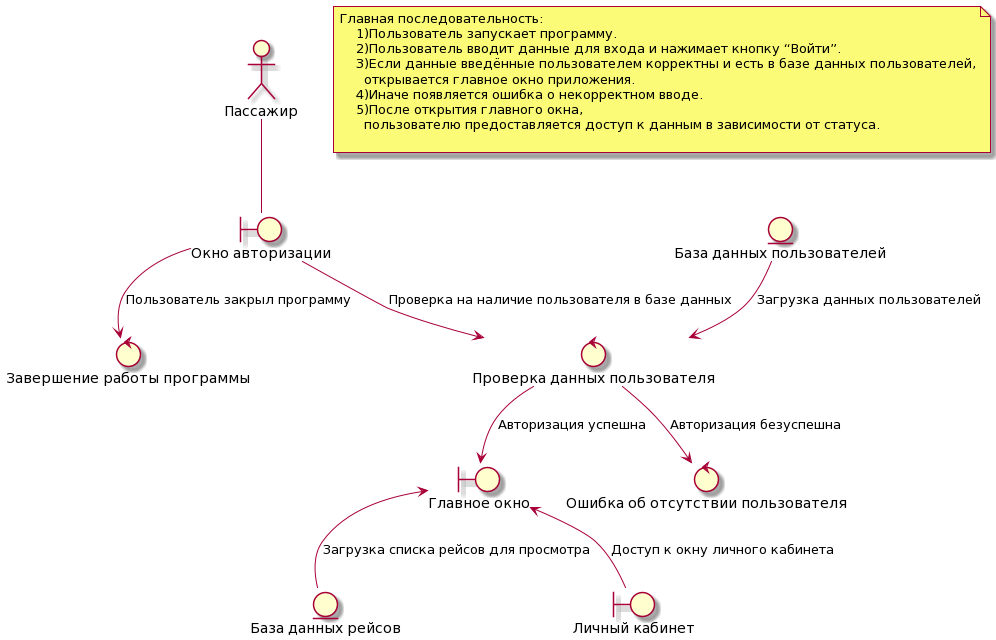


Рисунок 14 – Диаграмма пригодности прецедента “Войти в систему”

Диаграммы пригодности для прецедентов редактирования и добавления рейсов, доступных администратору и диспетчеру, представлены на рисунках 15-16.

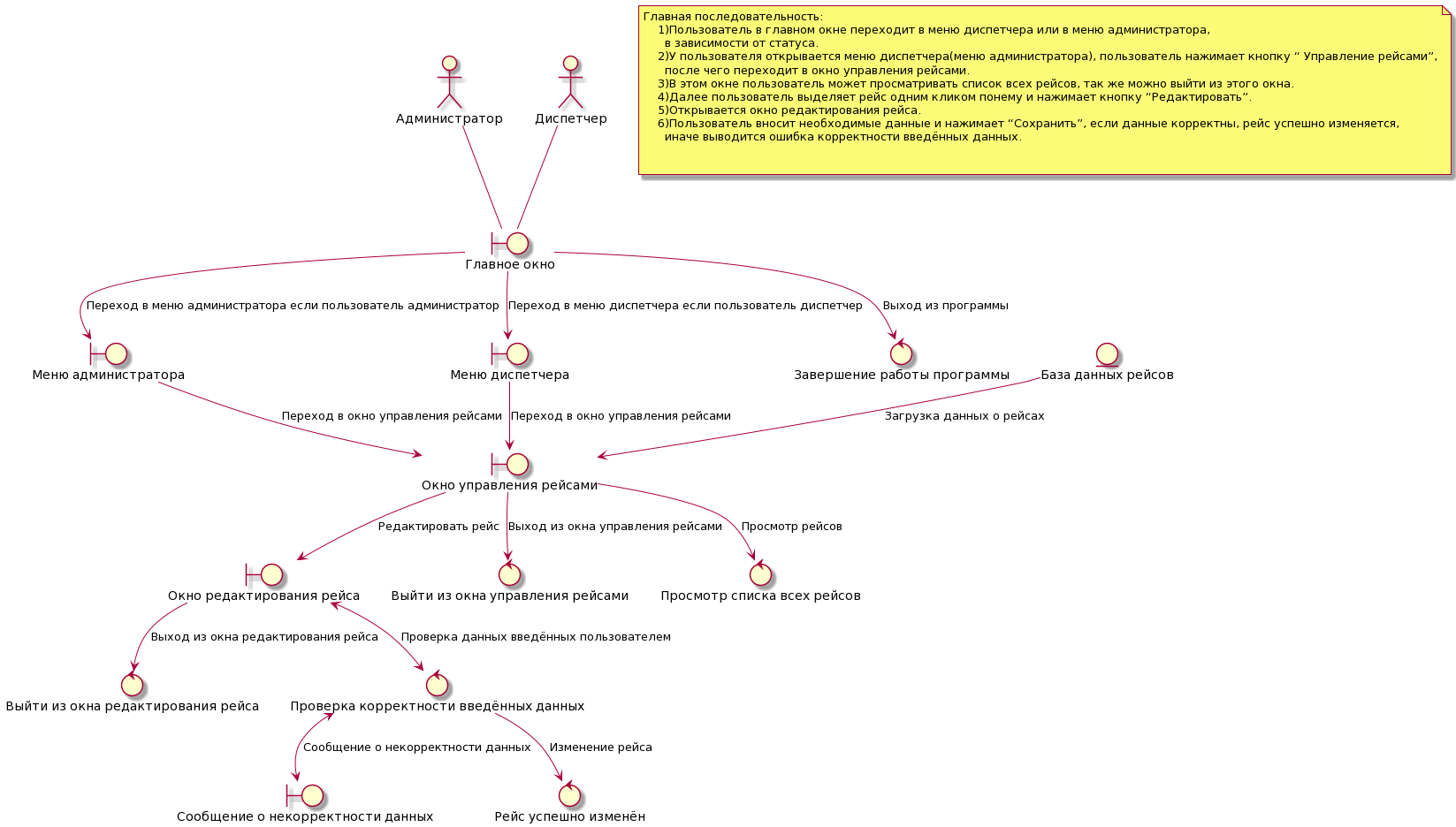


Рисунок 15 - Диаграмма пригодности прецедента “Изменить рейс”

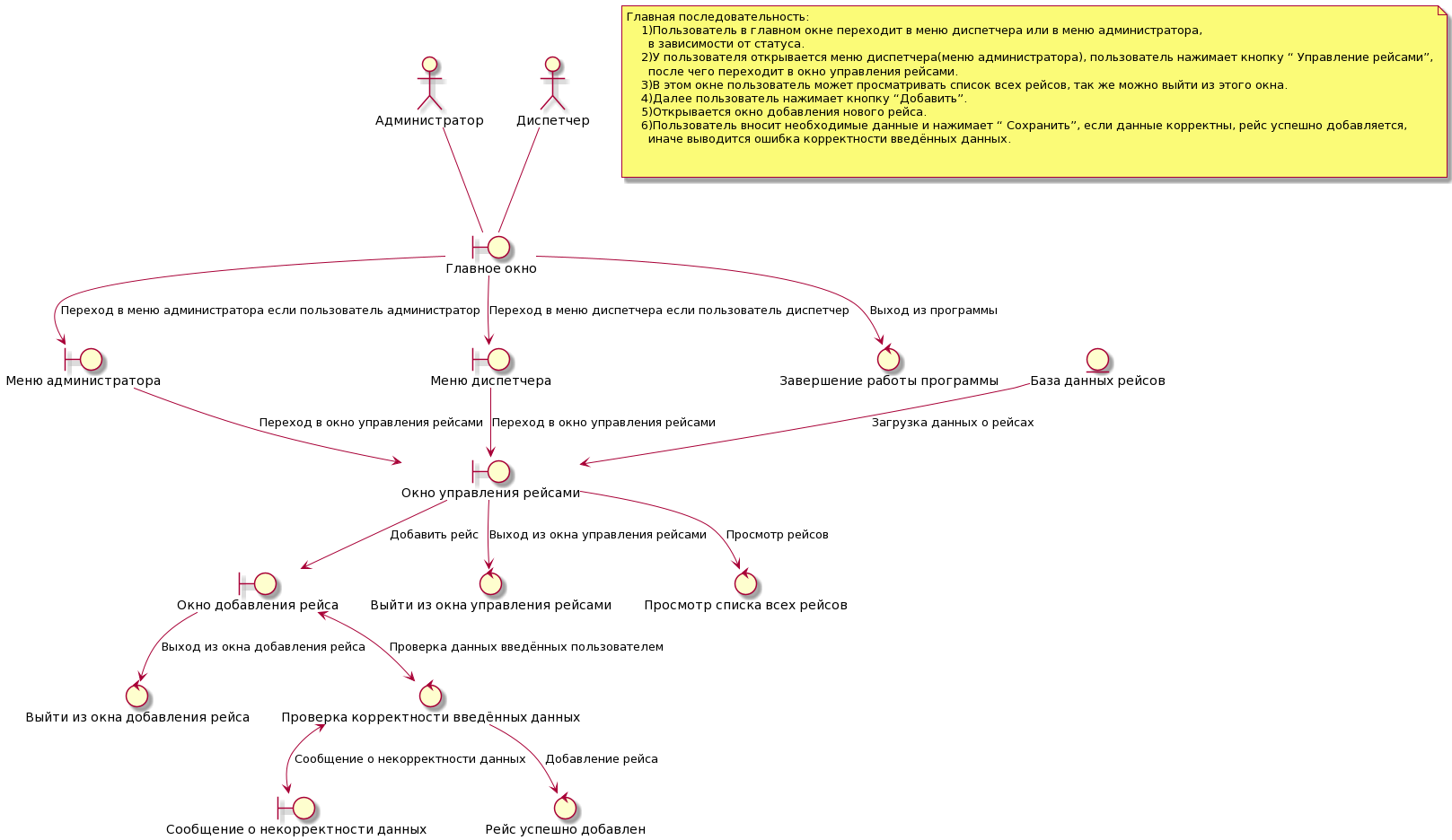


Рисунок 16 - Диаграмма пригодности прецедента “Добавить рейс”

Диаграммы пригодности для прецедентов добавления и удаления пользователя, доступных администратору показаны на рисунках 17-18.

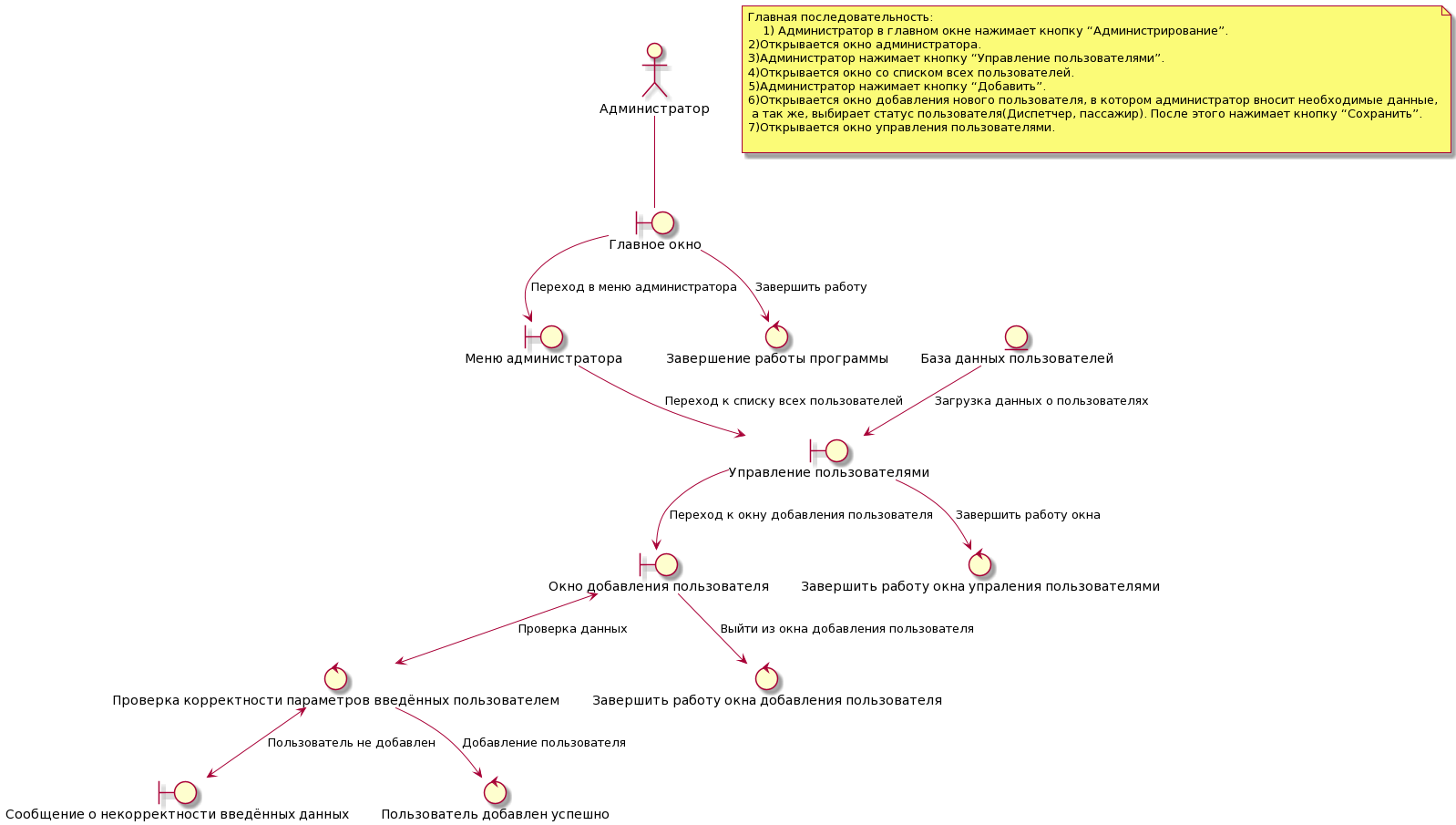


Рисунок 17 - Диаграмма пригодности прецедента “Добавить пользователя”

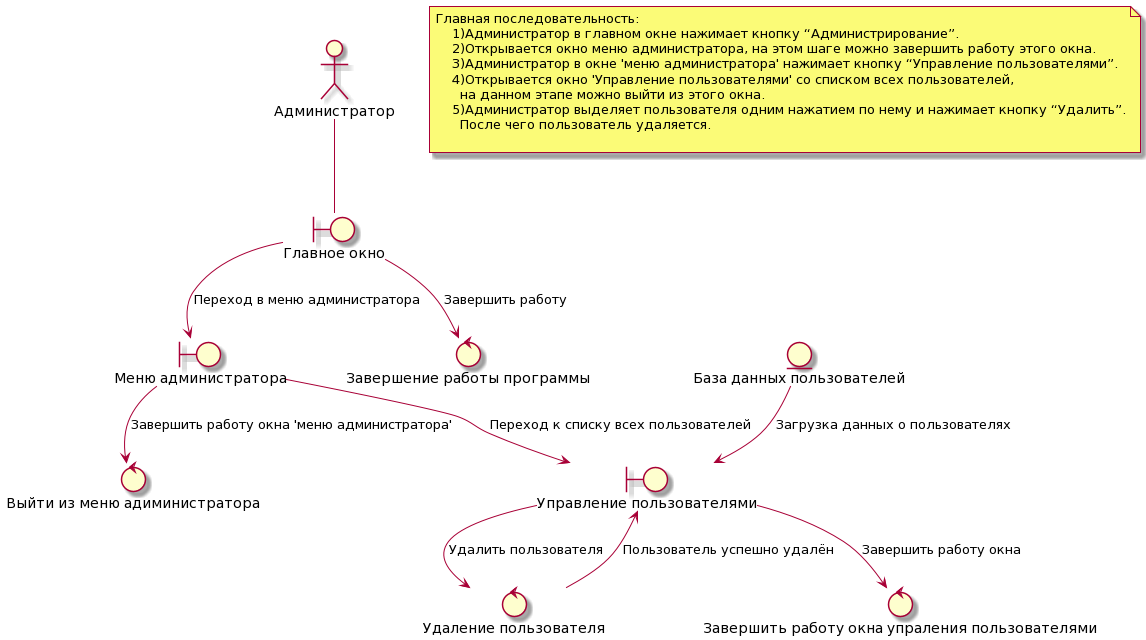


Рисунок 18 - Диаграмма пригодности прецедента “Удалить пользователя”

На рисунке 19 показана диаграмма прецедента “Отфильтровать рейсы”.

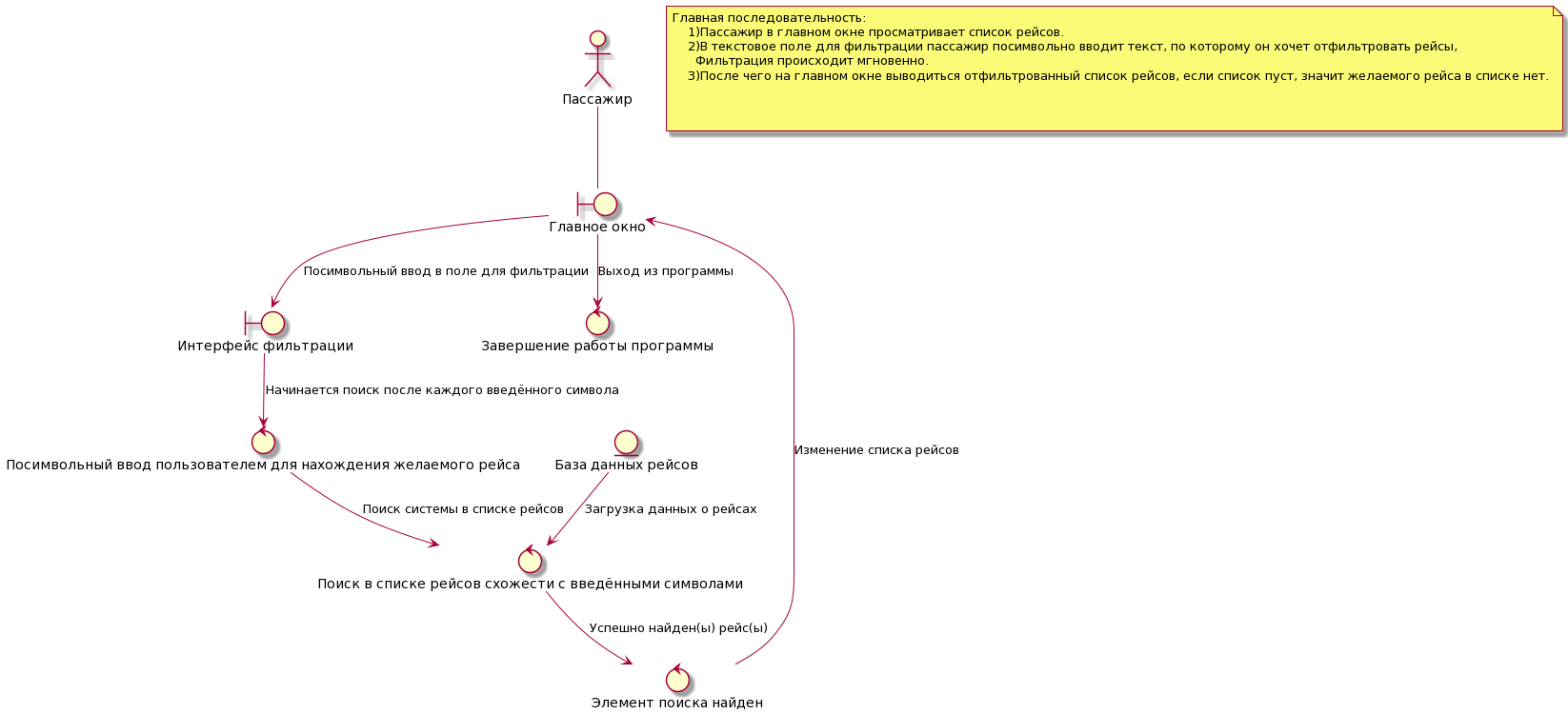


Рисунок 19 - Диаграмма пригодности прецедента “Отфильтровать рейсы”

## **2.3 Диаграммы последовательности**

В соответствии с построенными ранее диаграммами пригодности были разработаны и диаграммы последовательности, наглядно показывающие работу каждого из прецедентов. Диаграммы последовательности представлены на рисунках 20-26.

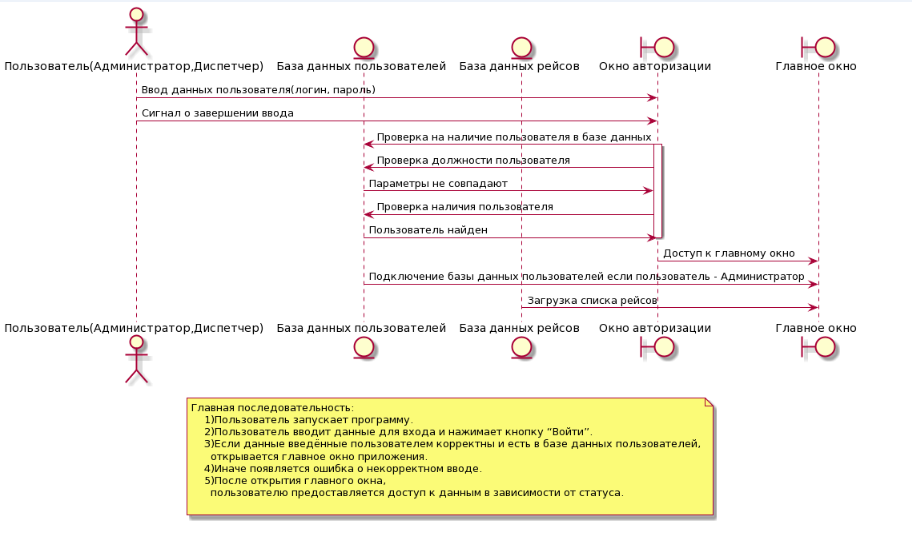


Рисунок 20 – Диаграмма последовательности прецедента “Войти в систему”

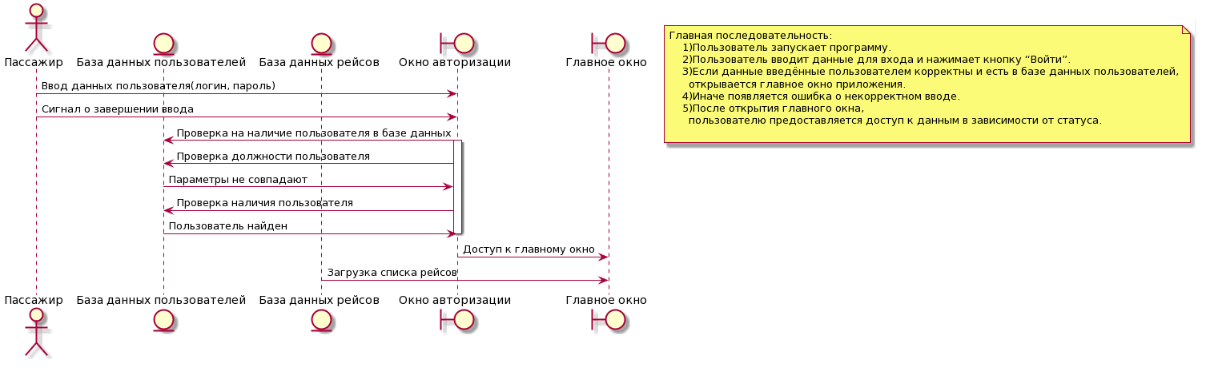


Рисунок 21 – Диаграмма последовательности прецедента “Войти в систему” для пассажира.

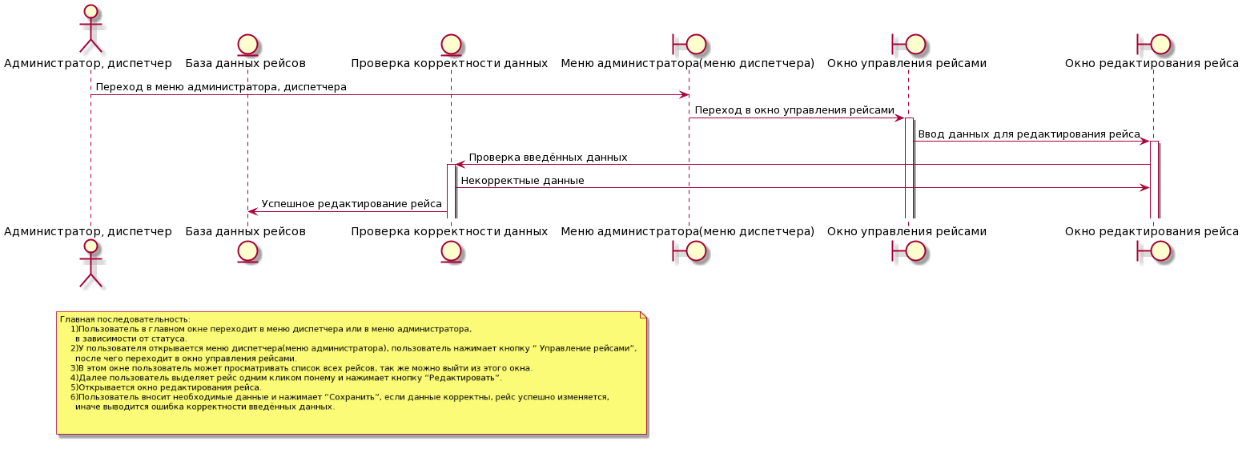


Рисунок 22 - Диаграмма последовательности прецедента “Изменить рейс”

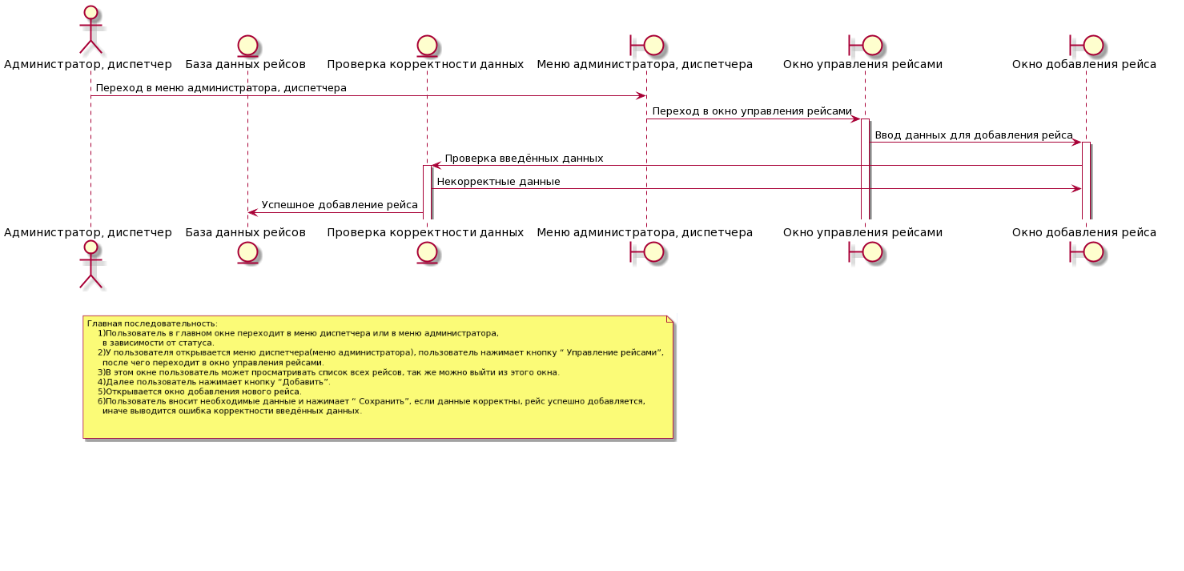


Рисунок 23 - Диаграмма последовательности прецедента “Добавить рейс”

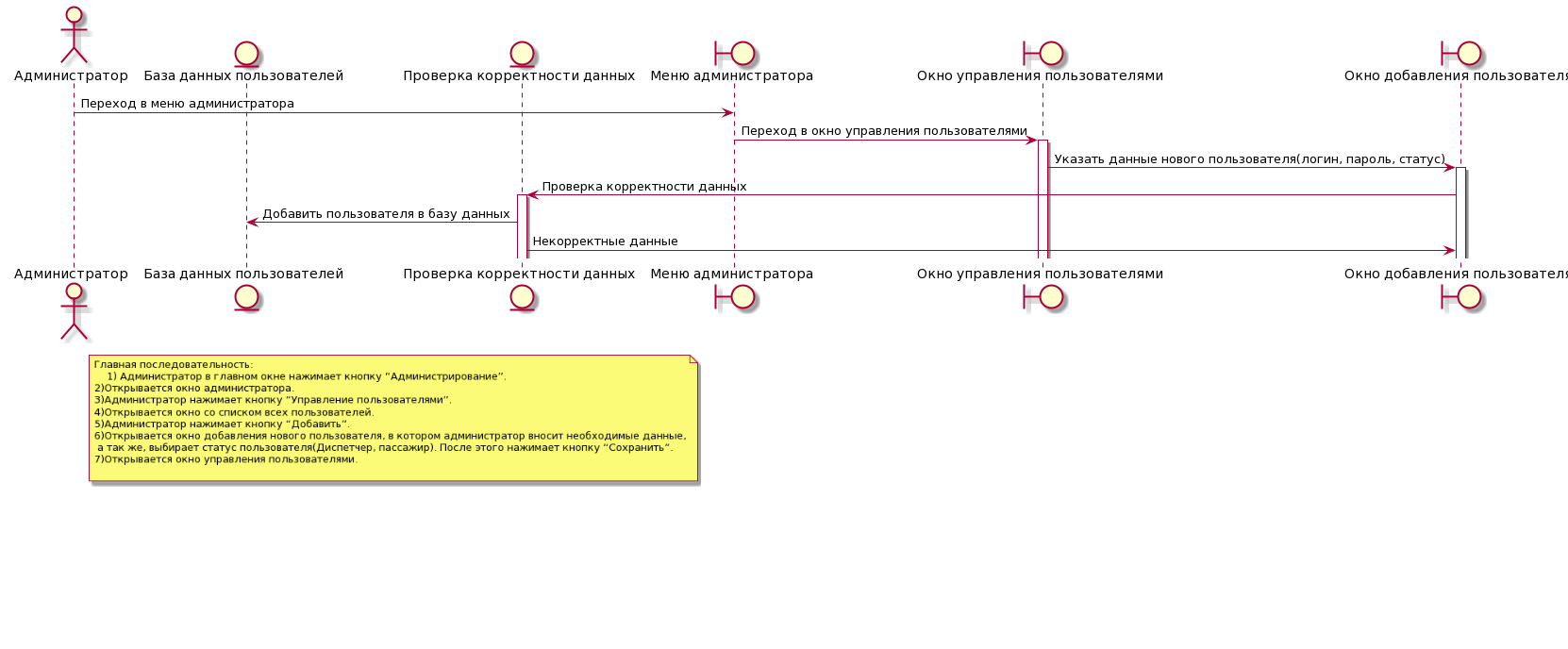


Рисунок 24 - Диаграмма последовательности прецедента “Добавить пользователя”

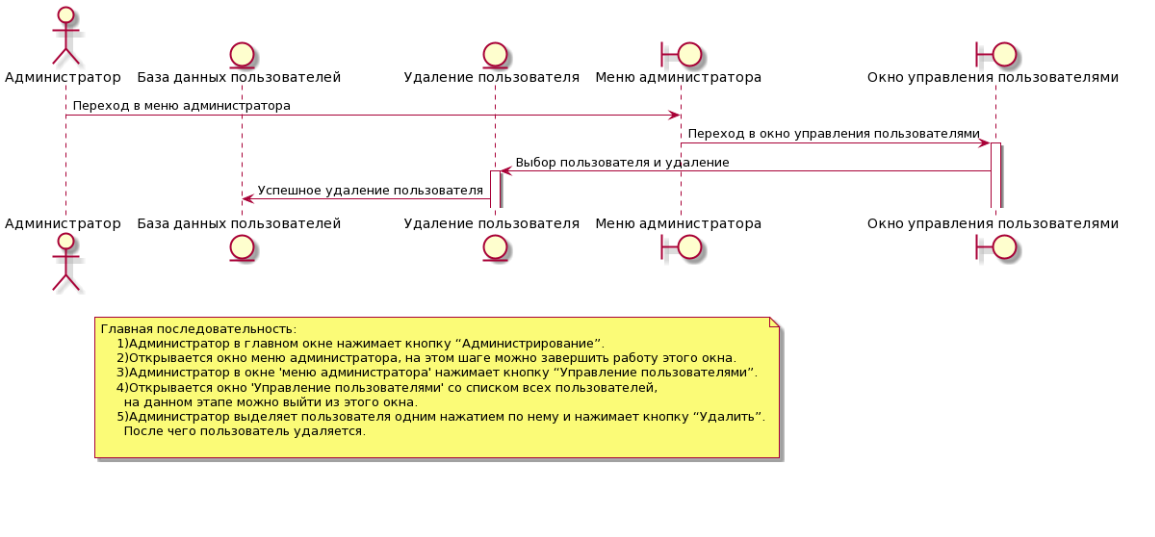


Рисунок 25 - Диаграмма последовательности прецедента “Удалить пользователя”

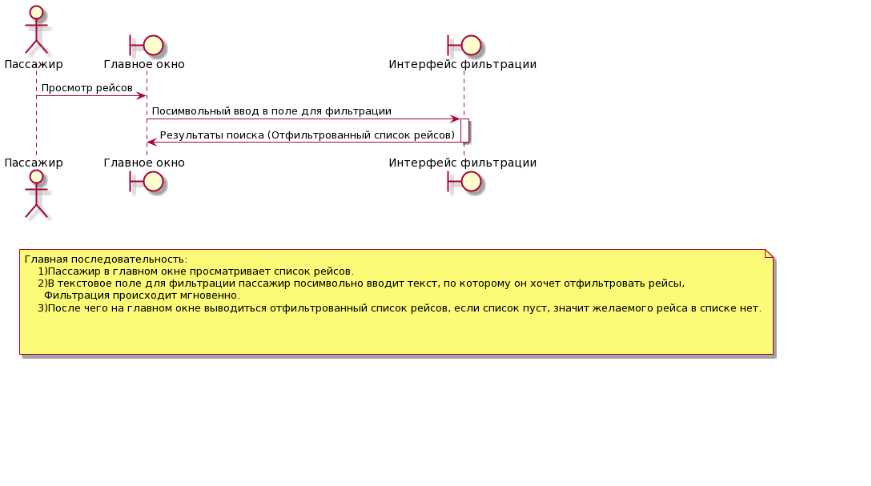


Рисунок 26 - Диаграмма последовательности прецедента “Отфильтровать рейсы”

## **2.4 Диаграмма классов**

На диаграмме классов наглядно показаны взаимодействия между классами системы, диаграмма показана на рисунке 27.

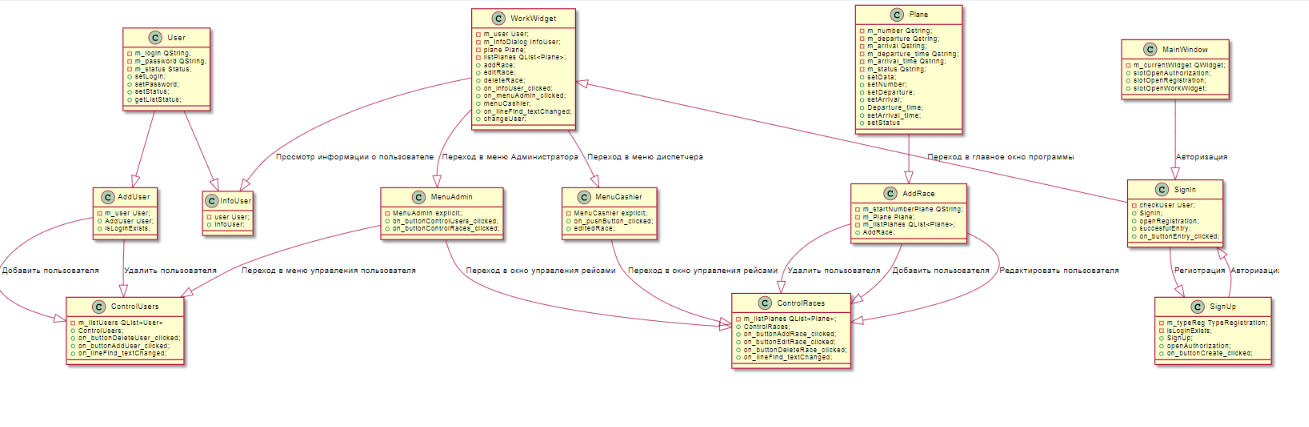


Рисунок 27 – Диаграмма классов

# **3. Программирование**

Для реализации программы была использована среда разработки “QtCreator”.

## **3.1 Реализация программы**

Коммиты проекта можно посмотреть по ссылке. Ссылка на репозиторий с реализованной программой находится в списке использованных источников[1]. По данным коммитам можно отследить развитие проекта и его пошаговое создание.

Инструкция по сборке проекта достаточно проста – требуется выполнить компиляцию проекта с пометкой Release, далее скопировать exe файл проекта и вставить его в отдельную папку. Можно предположить, что программа запуститься и будет работать на любом компьютере, однако это не так, программе будут требоваться определенные dll файлы из Qt framework для работы, поэтому требуется перенести все требуемые файлы, при запуске программы компьютер сам скажет, какие файлы нужны, а также папки stylesheet и т.д. После копирования всех нужных файлов и папок из корневой папки Qt в наш собираемый проект мы сможем его запускать.

Для пользователей у которых нет установленного QTCreator на компьютере, в репозитории есть папка release с готовой программой, в данной папке находиться .exe файл проекта и необходимые файлы с расширением .dll, нужно просто скачать эту папку себе на компьютер и запустить файл программу, ссылка на данную папку находиться в списке использованных источников[2].

При первом запуске программы у себя на компьютере, программа проверит наличие пользователя администратор в базе данных пользователей, если такого нет, откроется окно регистрации администратора, после ввода соответствующих данных, создаётся пользователь “Администратор”, при последующих запусках программы, это окно больше не появиться, так как администратор в системе только один. После создания администратора программа будет работать в стандартном режиме.

## **Рефакторинг**

Проект имеет гибкую архитектуру.

У каждого класса есть свои - set- и get- методы.

Для того чтобы сохранить или загрузить данные рейса или пользователя, нет необходимости прописывать логику операции лишний раз, для этого можно воспользоваться наследованными методами, такими как setData(), setNumber(), setDeparture(), setLogin(), и так далее. Аналогично с load-методами.

Код по большей части соответствует стандарту программирования «Руководство Google по стилю С++».

Так, например, в функциях, где параметры не должны быть явно или неявно изменены, используются квалификатор типов const. Наименование функций имеет camel case-стиль: getListStatus(), isLoginExists () и так далее. Наименование файлов также стандартизировано.

* 1. **Документирование**

Все документы по проекту находятся в папке “documents”, в репозитории GitHub, ссылку на документирование можно найти в списке использованных источников[3].

# **4. Заключение**

Программа соответствует целям и задачам работы, работает устойчиво, без прерываний, с вычислениями справляется без ошибок.

Неоднократно запускалась, так же на разных компьютерах.

Сохранение данных о пользователях и о рейсах в файлы производиться корректно, так же и эти данные извлекаются.

Функции программы выполняются корректно, интерфейс визуально проверен.

Программа работает корректно под разными пользователями.

Корректно выполняются функции фильтрации, добавления, редактирования, удаления рейсов, так же корректно выполняются функции добавления и удаления пользователей.

При использовании программы рекомендуется вводить сразу корректные данные, а именно диспетчер должен корректно вводить дату и время вылета (прилёта) рейса, чтобы избежать каких-либо ошибок, так как эти данные являются строкой и автоматически программой не устанавливаются.

Во время выполнения работы были проведены следующие работы:

1. Разработано техническое задание;
2. Реализована информационная система “Табло авиарейсов”;
3. Разработаны диаграммы для этой системы;
4. Оформлена пояснительная записка.

В процессе выполнения работы над проектом я улучшил свои навыки разработки программ с помощью среды разработки “QTCreator” и написания сопутствующих им документам.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Ссылка на репозиторий GitHub с реализованным проектом : https://github.com/Maslenok23082001/flight\_board.git

2. Cсылка на репозиторий GitHub с реализованным проектом для пользователя (папка release): https://github.com/Maslenok23082001/flight\_board.git

3. Ссылка на документацию по проекту (папка “documents”): https://github.com/Maslenok23082001/flight\_board.git

4. СТО 4.2-07-2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Введ. 30.12.2013. – Красноярск: СФУ, 2014. – 60 с.

5. Междисциплинарный курсовой проект базового уровня: учебно методическое пособие [Текст] / сост. А.В. Редькина, А.В. Редькин – Красноярск: Сиб. федер. ун-т, 2016. – 40 с. – Режим доступа: https://e.sfukras.ru/mod/resource/view.php?id=61924

6. ISO/IEC 19505-1:2012. Information technology – Object Management Group Unified Modeling Language (OMG UML) –

https://www.iso.org/standard/52854.html

7. Проектирование программного обеспечения [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/74330/