МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

ОТЧЁТ

по лабораторной работе

«Разработка информационного сервиса, моделирующего работу программно- аппаратного комплекса»

по дисциплине

«Параллельные вычисления»

7 вариант

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мартынов Д.С.

СТУДЕНТЫ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Седов А.М.

Терешин А.В.

Корзин Е.А

20-ПО

Нижний Новгород 2022

**ЗАДАНИЕ**. Разработать информационный сервис, моделирующий работу  
программно-аппаратного комплекса (ПАК). Моделирующая программа должна быть реализована, как Flask Python-приложение. Сама моделирующая программа должна быть реализована с использованием технологии параллельного программирования.

**ВАРИАНТ 7.** Кабина лифта

**В рамках работы необходимо:**  
1. описать список функциональных требований к разрабатываемой системе;  
2. построить User-Case UML диаграмму;  
3. построить IDEF0 модуль разрабатываемой системы, включая контекстные  
диаграммы по выделенным вариантам использования;  
4. построить диаграммы потоков данных - DFD и диаграммы взаимодействия  
процессов - WorkFlow (IDEF3);  
5. описать модель взаимодействия пользователя с разрабатываемой  
программной системой, разработать экранную веб форму для отображения  
программных данных пользователю и размещения в ней элементов  
управления;  
6. разработать логическую модель базы данных для ведения журнала событий в разрабатываемой системе);  
7. выполнить проектирование классов (используется объектно-ориентированный подход к проектированию);  
8. построить диаграммы взаимодействия (Interaction Diagrams) для  
выделенных компонент программной системы;  
9. построить диаграммы состояний системы для определенных вариантов  
использования;  
10. построить диаграммы пакетов, компонентов и размещения;  
11. развернуть разработанный программный комплекс на локальной машине и провести тестирование работы;  
12. составить инструкцию пользователя для работы с разработанным ПО.

1. **Список функциональных требований**

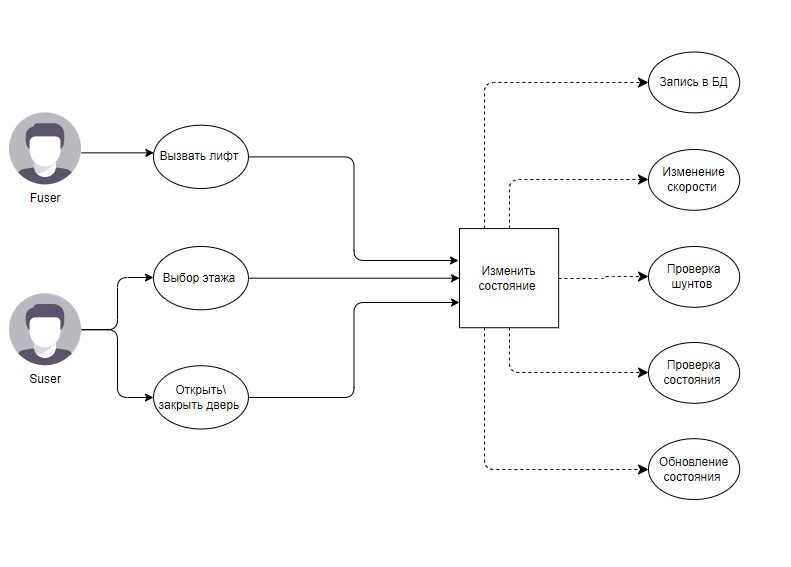
К функциональным требованиям разрабатываемой системы можно отнести следующие пункты:

1. Моделировать работу лифта (реализация системы контроля последовательного передвижения по этажам)
2. Настраивать датчиков для каждого состояния по запросу
3. Осуществлять переход на следующее/предыдущее состояния в каждый момент времени
4. Вести базу данных, хранящую в себе параметры, для каждого состояния в определения момент времени
5. Автоматическое закрывание двери(ей) должно зависеть:
   1. от нажатия кнопки направления для передвижения кабины;
   2. или дополнительных выключателей, защищенных от непреднамеренного срабатывания, для управления механизмом дверей;

6) Кабина не должна выходить за крайние пределы нормального пути движения, то есть не выходить за зону точной остановки при нормальной работе;

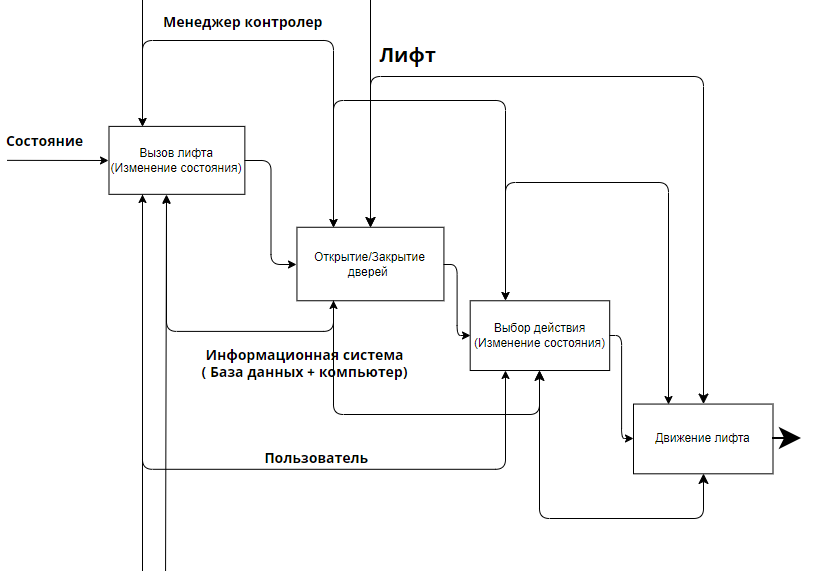
1. **User-Case UML диаграмма**

Диаграмма вариантов использования Flask – приложения отображает какой функционал разрабатываемой программной системы доступен каждой группе пользователей.



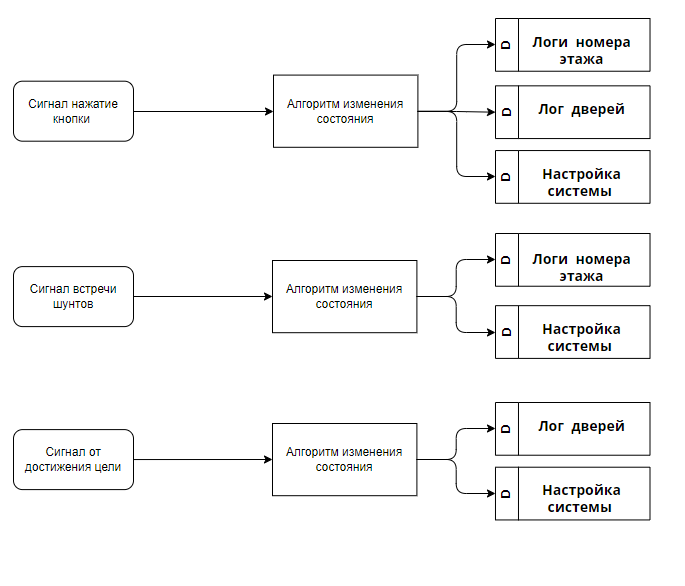
Пользователю, находящемуся в лифтовом холле (FUser) доступна, только кнопка запроса для вызова лифта на тот этаж, на котором он сейчас находиться. Пользователю, находящемуся в кабине лифта (SUser) доступны кнопки выбора этажа, так же кнопки закрытия и открытия дверей. Действия тех и других пользователей приводят к изменению алгоритма состояний. Значит, изменится последовательность движения лифта, обновится состояния для каждого светофора, данные будут отправлены в подключенную базу данных

1. **IDEF0 модуль разрабатываемой системы**

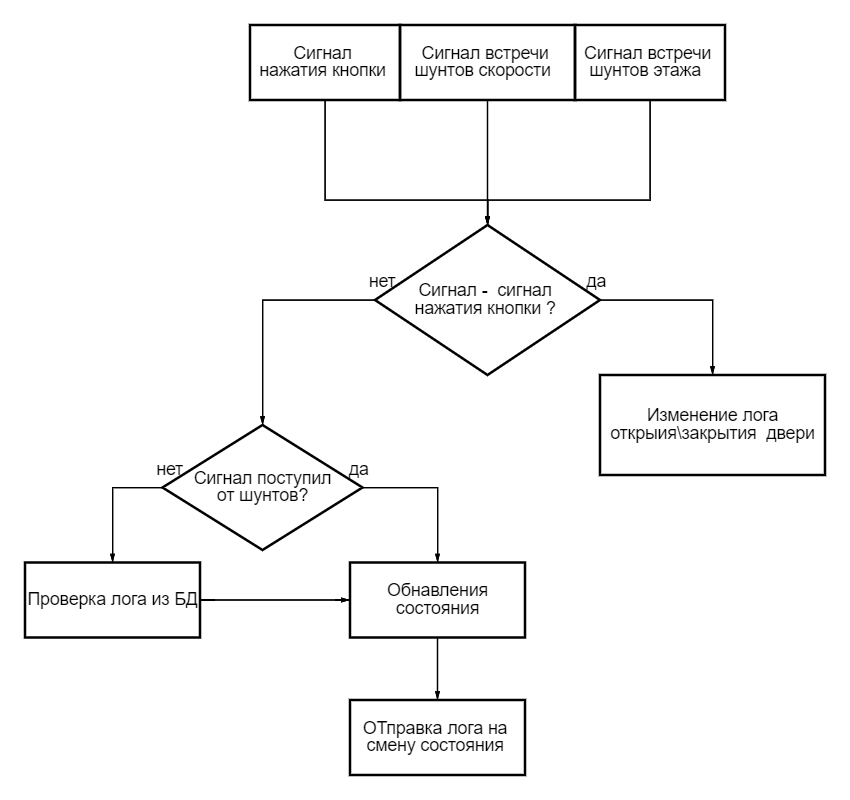
****

1. **Диаграммы потоков данных - DFD и диаграммы взаимодействия  
   процессов - WorkFlow (IDEF3)**

Диаграммы потоков данных (Data Flow Diagrams — DFD) представляют собой иерархию функциональных процессов, связанных потоками данных Все потоки приводят к алгоритму изменения состояний.



**Диаграмма IDEF3**  показывает взаимодействия потоков данных и процессов, происходящих в системе.



1. **Модель взаимодействия пользователя с разрабатываемой  
   программной системой**

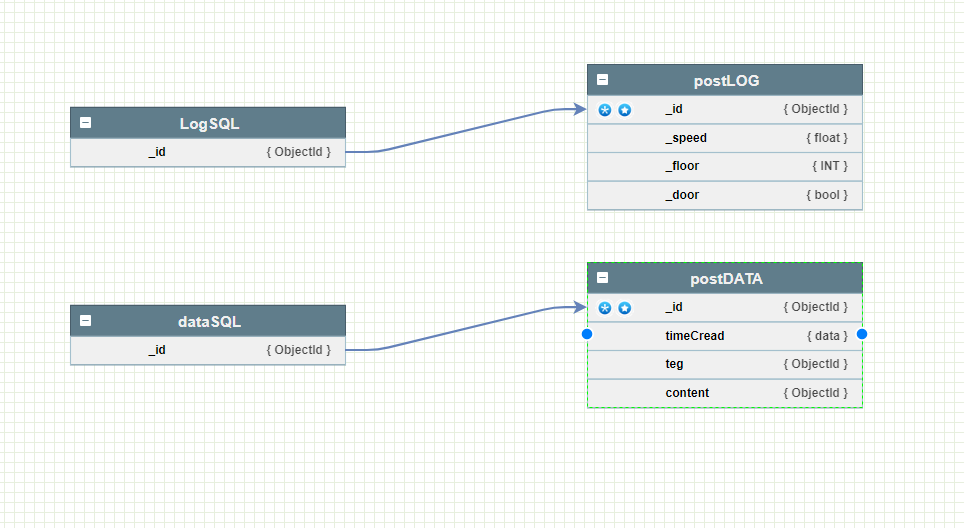
Пользователь может взаимодействовать с программной системой двумя способами.

Первый моделирует ситуацию, когда пользователь находится в кабине лифта. В ней он может нажать: кнопку нужного ему этажа (или нескольких этажей), кнопку закрытия или открытия дверей.   
При выборе любого этажа, его номер добавляется в список, на которые должен попасть лифт. Этот список сортируется по возрастанию, или по убыванию, в зависимости от того куда движется лифт, вверх, или вниз. Лифт выбирает направление по первому введенному этажу. Например, если пользователь находится в кабине лифта на третьем этаже и нажмет на кнопки в следующей последовательности: 5 – 4 – 1 – 2, то лифт поедет следующим маршрутом: 4 – 5 – 2 – 1. Такое поведение позволяет обслуживать запросы нескольких пользователей.  
Двери лифта открываются на определенное время, кнопка закрытия дверей – обнуляет таймер, кнопка открытия – увеличивает таймер.

Второй способ взаимодействия моделирует ситуацию, когда пользователь стоит на этаже и вызывает лифт. По логике своей работы кнопка на этаже идентична той, что находится в кабине лифта.

Подразумевается, что с программной системой взаимодействует множество пользователей, при этом каждый пользователь сначала вызывает лифт на свой этаж, а потом управляет им в кабине.

1. **Модель базы данных**

****

Базы данных LogSQL, dataSQL имеют по одному полю – id. Ключевое поле id связано с таблицами логами статуса и логами команд связью один к одному.

На каждое состояние в базу данных записывается время, скорость \_speed соответствует скорости лифта , \_door - положению дверей , \_floor – выборному этажу

1. **Проектирование классов**

В исходном коде программы определены следующие классы:

- Manager

Он управляет логикой работы лифта. При нажатии кнопок вызываются методы этого класса. Класс имеет ссылки на классы Elevator, Timer, DB. Объект этого класса создается в единственном экземпляре.

- Elevator

Класс содержит в себе поля, описывающее текущее состояние лифта. Класс имеет ссылки на классы DB, LiftShaft. Объект класса создается в единственном экземпляре.

- DB

Этот класс предназначен для работы с базой данных. С помощью него в одну таблицу базы данных заносится информация о текущем состоянии лифта, а в другую таблицу логируются все действия лифта. Объект этого класса создается при каждом обращении к БД.

- LiftShaft

Данный класс моделирует шахту лифта. Он содержит в себе список шунтов. Объект этого класса создается в единственном экземпляре.

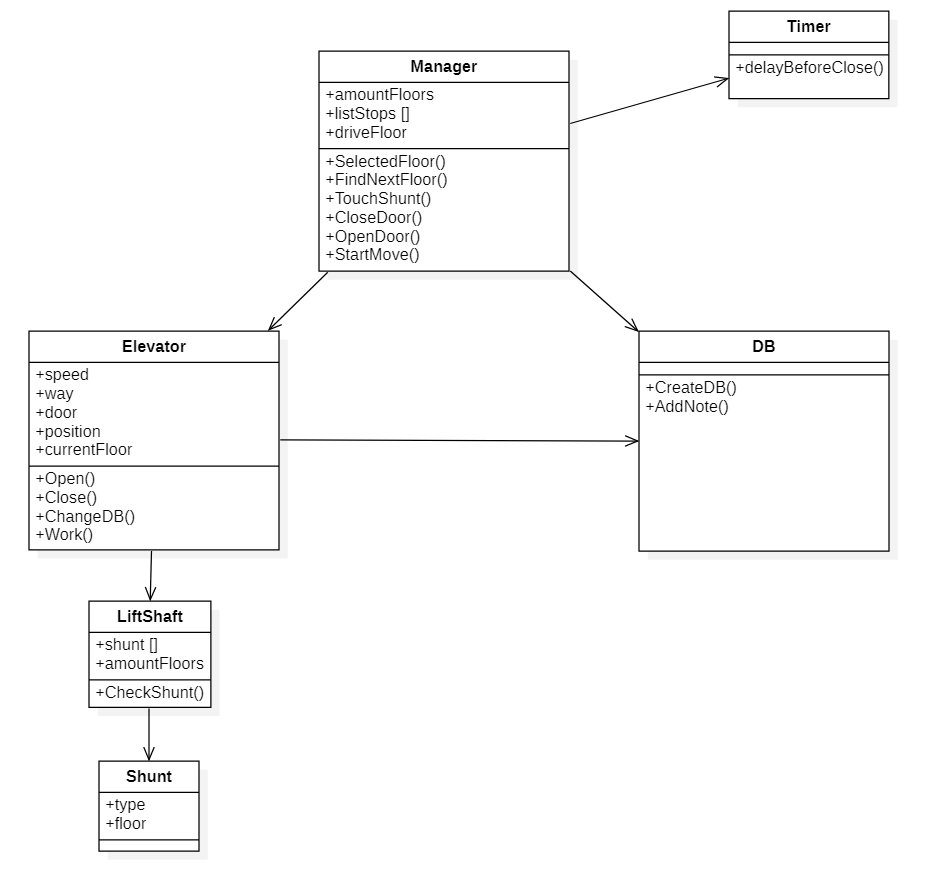
- Shunt

Класс содержит атрибуты, описывающие шунт.

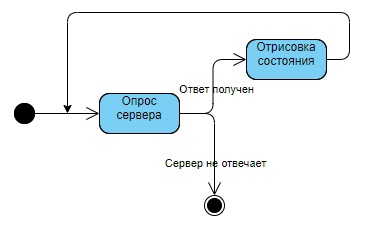
-Timer

Отвечает за работу таймера закрытия дверей. Объект этого класса создается в единственном экземпляре в конструкторе класса Manager.

1. **Диаграмма взаимодействия (Interaction Diagrams)**

****

1. **Диаграммы состояний системы**



1. **Диаграммы пакетов, компонентов и размещения**

Диаграмма пакетов:

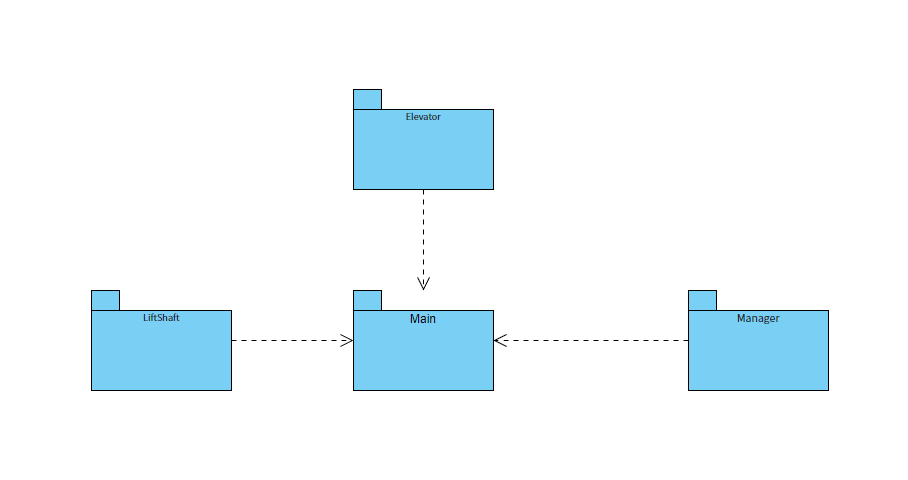
****

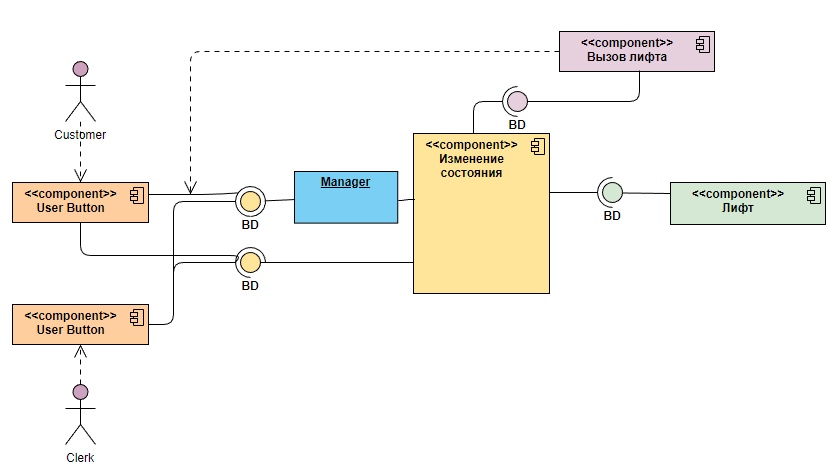
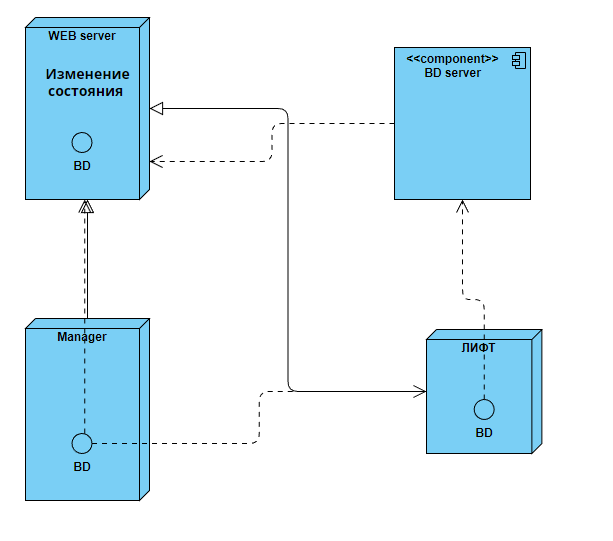
Диаграмма компонентов**:** ****

Диаграмма размещения: 

1. **Инструкция пользователя для работы с разработанным ПО**

**Инструкция пользователя:**

Для вызова лифта на конкретный этаж нужно нажать на кнопку с его номером. Как было описано выше, это можно сделать в кабине лифта, либо на любом из этажей. Для преждевременного закрытия дверей нужно нажать кнопку закрытия дверей. Если нужно, чтобы двери оставались открытыми дольше, следует нажать кнопку открытия дверей. Дверь возможно открыть только когда лифт не движется.