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**

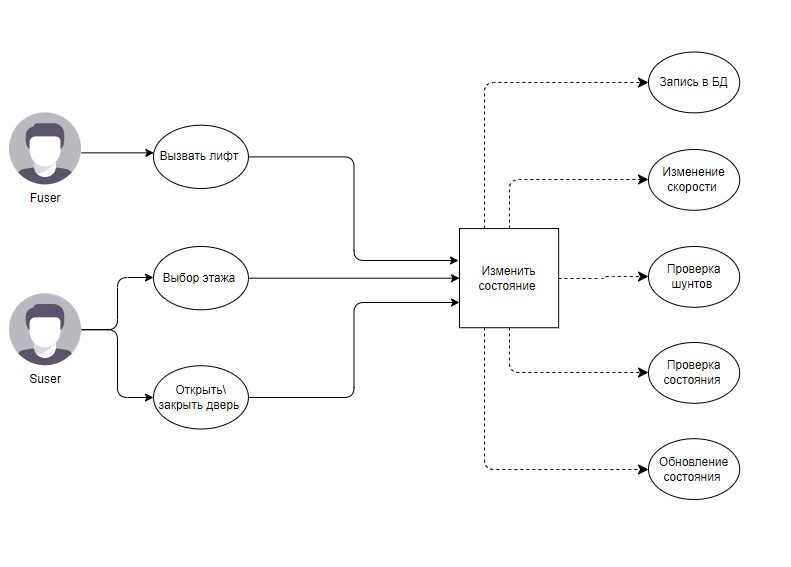
К функциональным требованиям разрабатываемой системы можно отнести следующие пункты:

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;

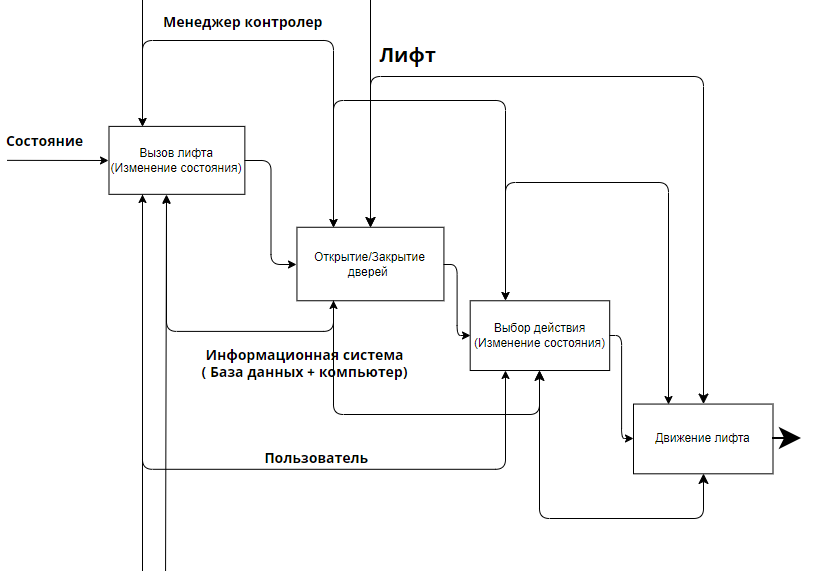
1. **User-Case UML диаграмма**

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



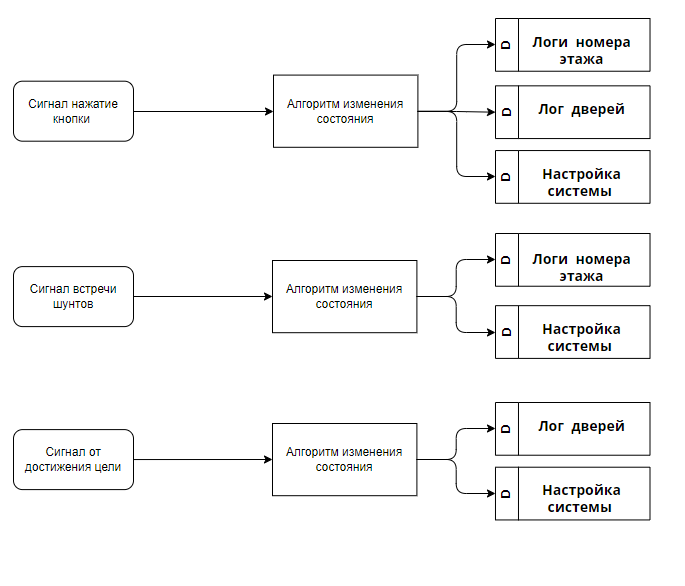
Пользователю, находящемуся в лифтовом холле (FUser) доступна, только кнопка запроса для вызова лифта на тот этаж, на котором он сейчас находиться. Пользователю, находящемуся в кабине лифта (SUser) доступны кнопки выбора этажа, так же кнопки закрытия и открытия дверей. Действия тех и других пользователей 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**

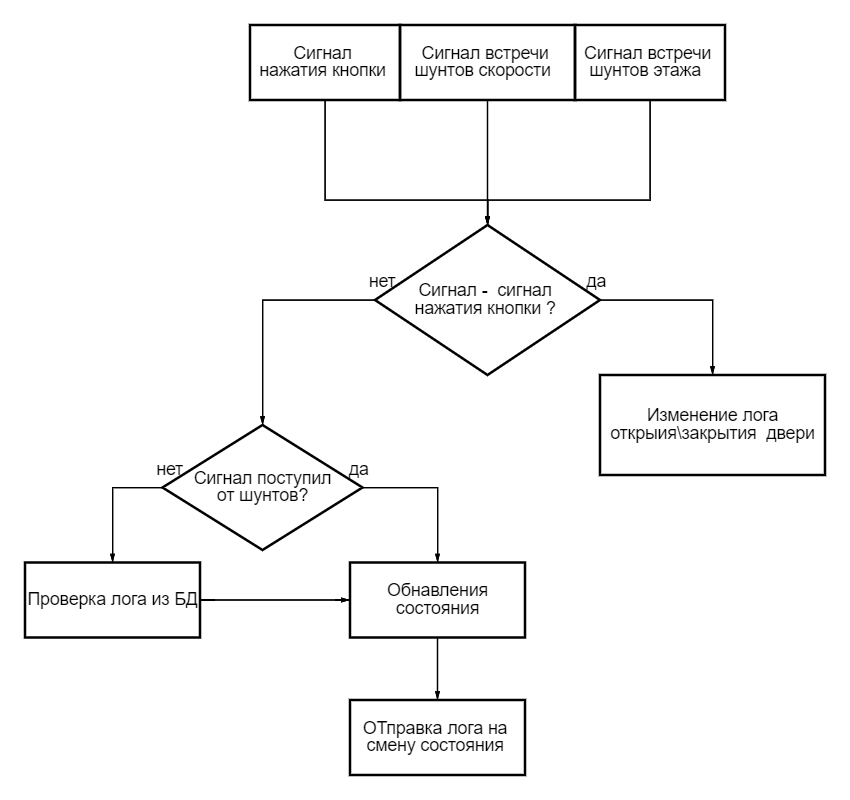
****

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 наблюдать модель работы пассажирского лифта

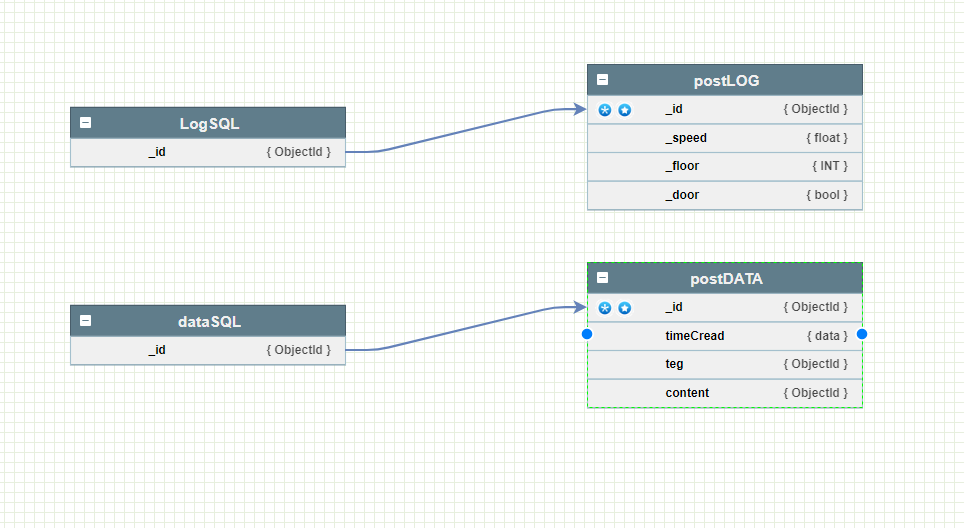
Лифт для автомобилей меняет состояния, исходя из указанного времени по таймеру. Состояние пешеходного светофора зависит от состояния автомобильного. Также он может регулироваться запросом – либо пользователя, либо диспетчера. Запрос реализуется через кнопки, предназначенные для пешехода и для диспетчера.

Существуют ограничения для кнопки пешеходов, которые зависят от следующих параметров. Состояние автомобильного светофора – кнопка будет нажата только в случае, если на этом светофоре горит зеленый свет. Причем она нажмётся через четверть оставшегося времени по таймеру. Во время ожидания пользователю выведена просьба «Пожалуйста, подождите». Если на таймере остаётся время, меньшее четверти максимального в момент нажатии кнопки, отсчет продолжается, не изменяя значения. Также прописывается через сколько секунд светофор переключится в другой режим.

Диспетчер относительно автомобильного светофора может менять его состояния кнопками «следующее состояние» и «предыдущее состояние» для наглядности работы модели. Также диспетчер имеет право самостоятельно задавать время таймера, от которого зависит состояние автомобильного светофора, при движении ползунка.

Для диспетчера реализован доступ к базе данных, где хранится время каждого состояния.

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.

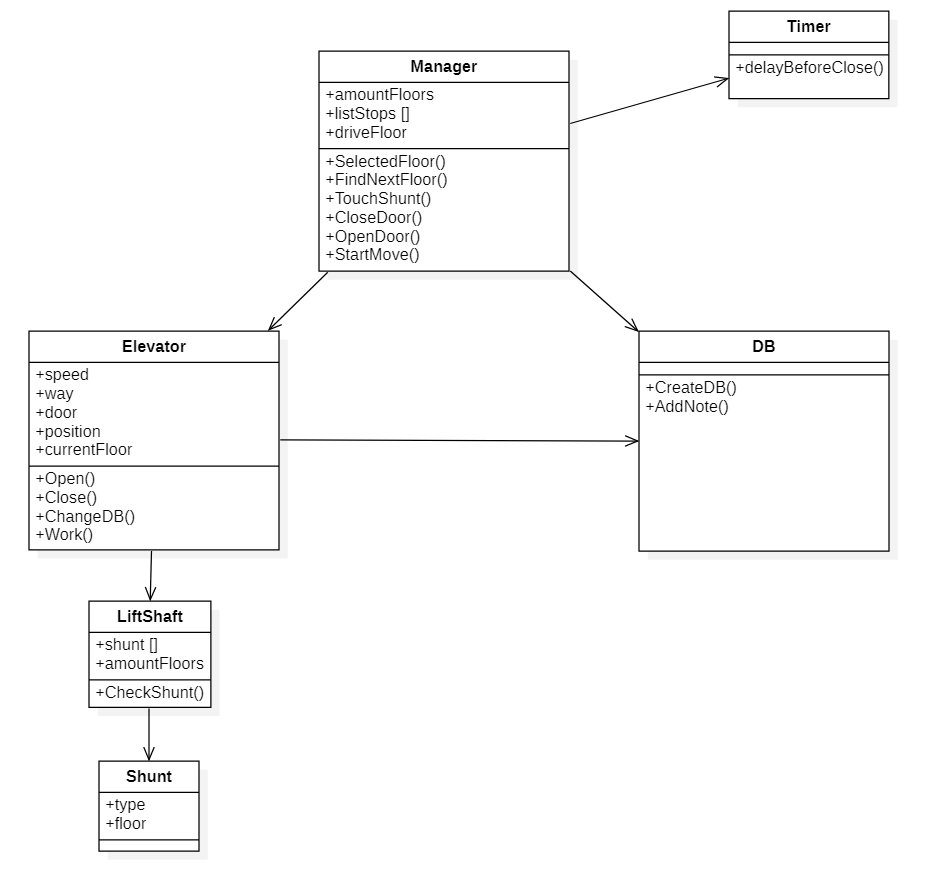
На каждое состояние в базу данных записывается время, скорость \_speed соответствует скорости лифта , \_door - положению дверей , \_floor – выборному этажу

1. **Проектирование классов**

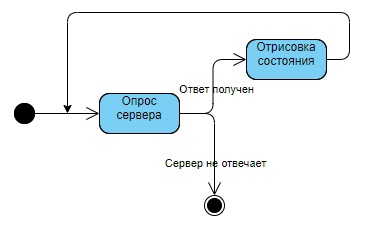
В коде программы реализован класс TrafficLights. Инициализируются экземпляры класса TimeArray, CarState, PeopleState, ResponsePeopleCD, TrafficOn.

Определены функции GetPeopleState – для получения состояния пешеходного светофора, getTime – для получения времени по таймеру, setTime – задать время по таймеру, UpdateTimer – обновить значение времени, TrafficStart – запустить светофор, PeopleButtonPressed – для нажатия кнопки пешеходом, DisableTrafficLight – выключает светофор, EnableTrafficLight – включает светофор, PrevState – возвращает предыдущее состояние, NextState – следующее состояние, ResponseState – ответное состояние.

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

****

1. **Диаграммы состояний системы**



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

Диаграмма пакетов:

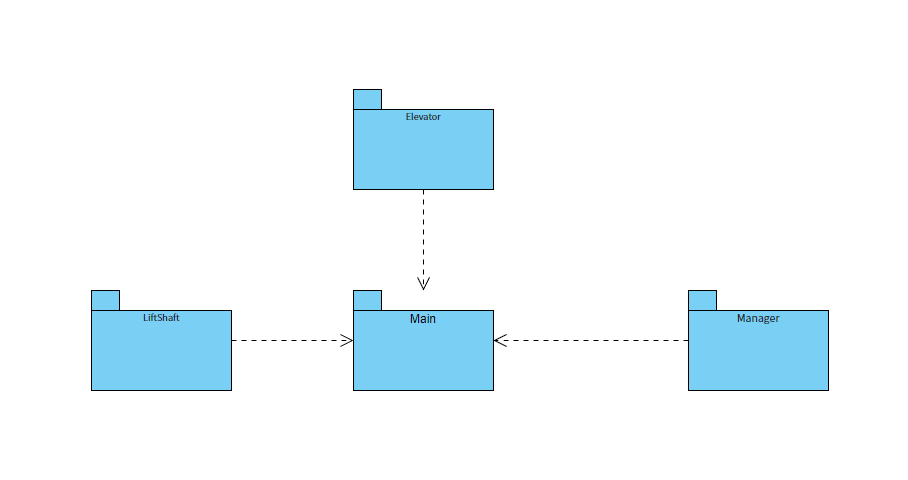
****

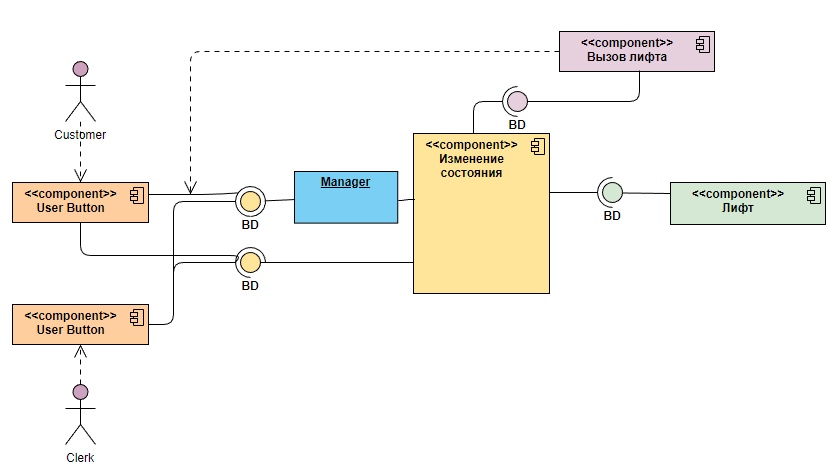
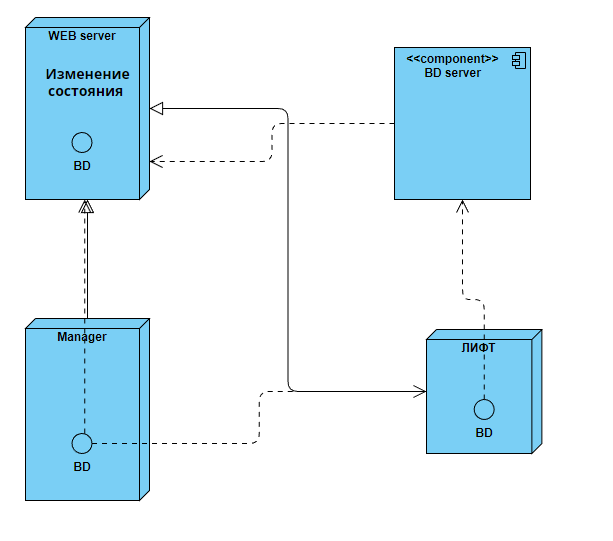
Диаграмма компонентов**:** ****

Диаграмма размещения: 

1. **Программный комплекс на локальной машине**
2. **Инструкция пользователя для работы с разработанным ПО**

**Инструкция пользователя:**

На кнопку нажми