МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

БАКАЛАВРСКАЯ РАБОТА

на тему Проектирование и реализация виртуальной системы управления лифтом

Студент К.И. Горшков _____

Руководитель Г.А. Тырыгина _____

Допустить к защите
Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский _____

«_____»_____20_____г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав.кафедрой «Прикладная
математика и информатика»
_____А.В.Очеповский

«____»_____2016 г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент Горшков Кирилл Игоревич
1. Тема Проектирование и реализация виртуальной системы управления лифтом
2. Срок сдачи студентом законченной выпускной квалификационной работы 23 мая 2016
3. Исходные данные к выпускной квалификационной работе Ранняя версия программы, реализующей виртуальную систему управления лифтом
4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов)
   - программа, реализующая виртуальную систему управления лифтом;
   - алгоритм поведения лифта;
   - модули программы для тестирования эффективности алгоритма.
5. Ориентировочный перечень графического и иллюстративного материала Презентация
6. Дата выдачи задания «11» января 2016 г.

Руководитель выпускной
квалификационной работы _____ Г.А. Тырыгина

Задание принял к исполнению _____ К.И. Горшков

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав.кафедрой «Прикладная
математика и информатика»
_____А.В.Очеповский

«____»_____2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента Горшкова Кирилла Игоревича
по теме Проектирование и реализация виртуальной системы управления лифтом

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Библиография	12.02.2016	12.02.2016	выполнено	
Обзор основных результатов	17.02.2016	17.02.2016	выполнено	
Введение	19.02.2016	19.02.2016	выполнено	
Общая информация	25.02.2016	25.02.2016	выполнено	
Система управления лифтом	2.03.2016	2.03.2016	выполнено	
Оформление главы 1	5.03.2016	5.03.2016	выполнено	
Анализ исходной версии программы	10.03.2016	10.03.2016	выполнено	
Определение задач для доработки программы	11.03.2016	11.03.2016	выполнено	
Оформление главы 2	16.03.2016	16.03.2016	выполнено	
Доработка программы	10.04.2016	10.04.2016	выполнено	
Усовершенствование алгоритма поведения лифта	15.04.2016	15.04.2016	выполнено	

Анализ эффективности алгоритма	25.04.2016	25.04.2016	выполнено	
Оформление главы 3	2.05.2016	2.05.2016	выполнено	
Оформление заключения	8.05.2016	8.05.2016	выполнено	
Представление ВКР	11.05.2016	11.05.2016	выполнено	
Подготовка презентации	11.05.2016	11.05.2016	выполнено	
Проверка работы на наличие заимствованных частей (антиплагиат)	30.05.2016	30.05.2016	выполнено	
Предзащита бакалаврской работы	6.06.2016 – 10.06.2016	31.06.2016	выполнено	
Сдача на кафедру отзыва научного руководителя и ознакомление с ним	20.06.2016	20.06.2016	выполнено	
Сдача необходимого комплекта документов бакалаврской работы	24.06.2016	24.06.2016	выполнено	
Защита выпускной квалификационной работы	27.06.2016 – 30.06.2016	29.06.2016	выполнено	

Руководитель выпускной квалификационной работы _____ Г.А. Тырыгина

Задание принял к исполнению _____ К.И. Горшков

Аннотация

Тема: Проектирование и реализация виртуальной системы управления лифтом.

Работа выполнена студентом Тольяттинского государственного университета, института математики, физики и информационных технологий, группы ПМИб-1201, Горшковым Кириллом Игоревичем.

Объект работы: программа, реализующая виртуальную систему управления лифтом.

Предмет исследования: алгоритм поведения лифта, заложенный в программе, реализующей виртуальную систему управления лифтом.

Цель работы: усовершенствовать алгоритм поведения лифта, заложенный в программе, реализующей виртуальную систему управления лифтом.

Цель работы требует решить следующие задачи:

1. Проанализировать текущее состояние исходной версии программы, реализующей виртуальную систему управления лифтом.
2. Проанализировать алгоритм поведения лифта, заложенный в программе.
3. Спланировать и осуществить дальнейшее усовершенствование программы.
4. Сравнить эффективность работы лифтов с разными версиями алгоритма поведения лифта.

Работа состоит из введения, трёх глав и заключения.

Во введении описываются актуальность, цели и задачи данной работы.

Первая глава работы посвящена общим теоретическим вопросам. В ней описываются общее устройство лифта и его системы безопасности, система управления лифтом и её значение, а также контроллер лифта и значение алгоритма поведения, заложенного в нем.

Во второй главе описывается процесс разработки исходной версии программы, проводится анализ её состояния, а также определяются задачи по доработке программы.

Третья глава раскрывает процесс реализации нового функционала программы. В ней также проводится сравнение эффективности работы лифтов с новым и старым алгоритмами поведения на основе статистических показателей.

Выпускная квалификационная работа представлена на 51 странице, включает 26 иллюстраций, 4 таблицы, список используемой литературы, состоящий из 24 источников.

Оглавление

Введение

Трудно представить современный город без лифтов. Ни один многоэтажный жилой дом не обходится без лифтового оборудования, а в общественных заведениях этот механизм выглядит настолько естественно, что скорее вызывает удивление отсутствие лифта, чем его присутствие. Последнее время появилась потребность в пассажирских лифтах для индивидуальных домов, потому что этот механизм не просто облегчает жизнь и делает ее более комфортной, но иногда бывает просто насущной необходимостью – это видно на примере людей, ограниченных в способности самостоятельно передвигаться. Помимо обыкновенных лифтов и эскалаторов на заводах изготавливают специальные подъёмные механизмы для инвалидов, которые устанавливаются в общественных зданиях, жилых домах, аэропортах, на авто- и железнодорожных вокзалах, в автобусах.

Первые лифты были известны еще в Древнем Риме в I в. до н. э. Упоминание о лифтах более позднего периода относится к VI в. (например, лифт Синайского монастыря в Египте), к первой четверти XIII в. (во Франции) и XVII в. (лифт Виндзорского замка в Англии, “летающий стул” Велайера в одном из парижских дворцов).

В середине XIX в. в США появились лифты Э. Отиса с ловителями, удерживающими кабину от падения в случае обрыва канатов.

С 60-х годов XIX в. в практику вошли лифты с паровым приводом, затем с гидравлическим, и только к началу XX в. широкое и преимущественное развитие получили электрические лифты.

С ростом количества выпускаемых лифтов совершенствуется и их конструкция. Главная особенность лифтов заключается в том, что они представляют собой автоматизированную систему, действующую по командам пассажиров. При этом все операции по доставке пассажиров на требуемый этаж и по обеспечению безопасности перевозок выполняются

автоматически. Поэтому важно, чтобы лифт работал максимально эффективно. Это определяет актуальность данной работы.

Данная выпускная квалификационная работа основывается на ранней версии программы, реализующей виртуальную систему управления лифтом. Программа имеет трехуровневую архитектуру и написана на языке C# с применением таких технологий, как Entity Framework и Microsoft Unity.

Объект работы: программа, реализующая виртуальную систему управления лифтом.

Предмет исследования: алгоритм поведения лифта, заложенный в программе, реализующей виртуальную систему управления лифтом.

Цель работы: усовершенствовать алгоритм поведения лифта, заложенный в программе, реализующей виртуальную систему управления лифтом.

Цель работы требует решить следующие задачи:

5. Проанализировать текущее состояние исходной версии программы, реализующей виртуальную систему управления лифтом.
6. Проанализировать алгоритм поведения лифта, заложенный в программе.
7. Спланировать и осуществить дальнейшее усовершенствование программы.
8. Сравнить эффективность работы лифтов с разными версиями алгоритма поведения лифта.

В данной работе будет проанализирована ранняя версия программы, реализующей виртуальную систему управления лифтом, её состояние, алгоритм поведения, использующейся в ней, а также спланирована и осуществлена доработка программы. В готовой программе появится возможность оценить эффективность работы лифта, что позволит нам сравнить эффективность работы лифтов со старым и новым алгоритмами.

Глава 1 Анализ теоретического материала

1.1 Лифт, его назначение и принцип работы

1.1.1 Лифт и его разновидности

Лифт – разновидность грузоподъёмной машины, предназначенная для вертикального или наклонного перемещения грузов на специальных платформах, передвигающихся по жёстким направляющим.

Лифт – очень удобное изобретение, которое позволяет сильно облегчить перемещение грузов или людей. За всю его историю он претерпевал большое количество изменений, но в целом лифты можно разделить на *пассажирские* и *грузовые*. Вне этой классификации находятся подъемники, эскалаторы и траволаторы.

Традиционно пассажирские лифты делят на шесть классов:

1. Пассажирские лифты: предназначены для перевозки пассажиров в домах различного типа и этажности (жилых, общественных, административных или производственных). Грузоподъёмность таких лифтов – от 225 до 1000 кг. Наиболее крупные фирмы-производители пассажирских лифтов на российском рынке – KLEEMANN, OTIS, ThyssenKrupp. В низком ценовом сегменте находятся российские производители КМЗ, МЭЛ и ЩМЗ.
2. Коттеджные лифты: размещаются в загородных домах и коттеджах. В связи с развитием строительства таких домов, использование коттеджных лифтов становится все популярнее. И даже если в доме не более 2-х этажей, такой лифт, несомненно, облегчит людям жизнь. Особенно это важно, если в доме проживают люди пожилого возраста или люди с ограниченными возможностями.
3. Панорамные лифты всегда изготавливаются индивидуально, т. к. заказываются обычно для офисных помещений, торговых центров или гостиниц. Кабина такого лифта имеет прозрачные стены; очень часто это является частью архитектурного решения здания или

дизайна помещения. Панорамные лифты можно установить и в таком здании, в котором они были не предусмотрены проектом. В данном случае лифтовая шахта устанавливается снаружи здания, что позволяет пассажирам любоваться внешним пространством. Как упрощённый вариант панорамных лифтов существуют также обзорные. Это те же обычные лифты, несколько стен и двери которых сделаны из ударопрочного стекла. Стоимость их до 2-х раз ниже панорамных и во многом они не уступают лифтам с панорамным обзором.

4. Больничные лифты очень важны для качественного медицинского обслуживания. Их особенности – это размещение в кабине лежачего больного и транспортирование его на любой этаж без остановки по вызову с других площадок. Для кого-то отсутствие таких функций может стоить жизни. Также к лифтам данного типа применяются повышенные требования безопасности. А специальные системы позволяют доехать до этажа и открыть двери даже в том случае, если возникнут проблемы с электроэнергией.
5. Лифты для инвалидов – это разновидность обычных лифтов с небольшими изменениями. Так, кнопки пульта управления располагаются на уровне инвалидной коляски горизонтально, а ширина дверного проёма устанавливается не менее 900 мм. Для инвалидов также разработаны вертикальные подъёмники. Они используются там, где установка лифта из-за небольшой высоты не целесообразна. Такие устройства просты в установке и эксплуатации, а высота подъёма – не более 4 м.
6. Пожарные лифты – это разновидность пассажирских, которые имеют дополнительные функции и системы высокой надёжности. Так, в кабину устанавливают датчики тепла и дыма, а при производстве используются негорючие материалы.

Грузовые лифты бывают трех видов:

1. Грузовой лифт. Такие лифты предназначены для установки в жилых домах и супермаркетах, торговых центрах и промышленных зданиях, словом, везде, где требуется перемещение тяжёлых грузов. Грузоподъёмность таких лифтов – от 50 до 5000 кг, а скорость передвижения не превышает 0,5 м/с. На рынке грузовых лифтов большую нишу занимают российские производители ЩЛЗ и КМЗ, а также иностранные компании BKG и SKG. Грузовые лифты подразделяются на большие и малые. Большие предназначены для транспортировки габаритных грузов, таких как автомобили, контейнеры или станки. К малым (или техническим) лифтам относятся устройства с грузоподъёмностью не более 250 кг.
2. Малый грузовой лифт. Такие устройства используются в ресторанах, кухнях, библиотеках, магазинах и складских помещениях. Они могут быть установлены в существующую шахту или идти в комплекте с несущей шахтой из металлокаркаса. Эти устройства поглощают мало энергии и практически бесшумны. Ещё один плюс – их не нужно регистрировать в органах Ростехнадзора.
3. Автомобильный лифт. Такие лифты позволяют парковаться в многоэтажных парковках или автосервисах. Устройством может управлять как сам водитель, так и сопровождающий человек. Данные лифты могут быть как горизонтального, так и вертикального типа, а их использование экономит площадь для заезда автомобилей.

Для перемещения большого количества пассажиров с этажа на этаж изобретены такие устройства, как эскалаторы и траволаторы.

Эскалатор – это устройство с непрерывно движущейся ступенчатой лентой. Бывают двух типов: поэтажные и тоннельные. Поэтажные используются в крупных торговых центрах или аэропортах, тоннельные же – в метро.

Траволатор отличается от эскалатора тем, что не имеет ступеней. Он представляет из себя непрерывно движущуюся ленту с небольшим углом наклона и предназначен для перемещения не только людей, но и небольших грузов.

Последний из нерассмотренных типов лифтов – подъёмники. Они подразделяются на 3 типа:

- шахтный – монтируется в шахте, а подъём груза осуществляется с помощью тали, которая перемещает платформу;
- строительный – является одним из видов мачтовых подъёмников и применяется при строительстве зданий, отделочных работах и ремонте. Устанавливается чаще всего вдоль фасадов где крепится с помощью специальных крюков;
- мачтовый – используется для перемещения грузов очень широко, потому что не требует каких-либо дополнительных конструкций для крепления, кроме основания. Он намного дешевле грузового лифта, поэтому его можно встретить как на складах или в магазинах, так и в автосервисе и даже в ресторанах.

Лифты вошли в нашу жизнь везде, и теперь уже практически невозможно представить себе существование без этого замечательного устройства.

1.1.2 Принцип работы лифта

Главной частью лифта является подъёмный механизм (лебедка). Она установлена в машинном помещении и с помощью подъемных канатов и подвески перемещает кабину на различные этажи обслуживаемого здания, имеет возможность останавливаться на каждом этаже таким образом, чтобы пол кабины был по возможности на уровне пола этажной площадки. Для уравновешивания кабины предусмотрен специальный противовес. Он крепится к тем же канатам, что и кабина.

Подвижные части лифта, например, кабина, перемещаются в шахте – специально оборудованном сооружении – которую со стороны этажных площадок оборудуют дверями шахты. В верхних и нижних частях каркаса кабины устанавливают так называемые башмаки. Они четко фиксируют кабину в горизонтальном направлении, не давая ей отклоняться в стороны и соприкасаться с противовесом.

В аварийных ситуациях, например, когда кабина лифта развивает скорость выше дозволенной (предельной) срабатывают специальные ловители, установленные на кабине и, иногда, на противовесе. Ловители прочно удерживают кабину на направляющих. Срабатывание ловителей при превышении скорости кабины обеспечивается ограничителем скорости с канатом ограничителя скорости и его натяжным устройством. Если ослабляется хотя бы один подъемный канат срабатывает электрический выключатель. Он обесточивает цепи управления лифтом и лебёдку главного привода, тем самым прекращая дальнейшую работу лифта. При отказе системы управления кабина или противовес могут пройти ниже нижнего рабочего положения. Для предотвращения жесткого удара о пол шахты предусмотрены упоры, или буфера, которые находятся в нижней части шахты, называемой приямком, и смягчают удар при посадке. В машинном помещении размещаются подъемный механизм, ограничитель скорости и станция управления. В некоторых лифтах предусмотрено блочное помещение, расположенное под машинным помещением, над шахтой. В нём устанавливают контрблоки (контршкивы). Подробнее устройство лифта изображено на рис. 1.1.

Рисунок 1.1 – Устройство и система безопасности лифта

Для управления движением кабины служат кнопки, расположенные на стене внутри кабины (кнопки отправки) или на стене этажной площадки здания (кнопки вызовов). Сигнал от кнопок передаётся по кабелям и проводам в станцию управления.

Если кабина лифта стоит на этаже, на котором нажата кнопка, то сигнал подаётся в станцию управления, а затем обратно в кабину лифта, к дверному механизму – двери открываются или закрываются. После отработки дверей, кнопки вызовов или отправки будут включены в работу.

Если пассажир нажимает кнопку вызова (кабина находится на другом этаже) или кнопку отправки (кабина не находится на целевом этаже), тогда сигнал передаётся в станцию управления, задействуются многие лифтовые механизмы: реле движения, реле точной остановки, реле контроля дверей, этажное реле, лебёдка главного привода и т.д. Кабина приходит в движение. На большой скорости кабина движется до тех пор, пока не отключится контактор большой скорости и своими блок-контактами не подаст напряжение на катушку контактора малой скорости. Тогда включается малая скорость движения лифта. Движение лифта на малой скорости продолжается до тех пор, пока кабина не доедет до датчика точной остановки расположенного на направляющей движения кабины. По сигналу датчика точной остановки кабина затормаживается и удерживается тормозом в неподвижном состоянии. Одновременно подаётся питание на электродвигатель привода дверей кабины. Двери автоматически открываются совместно с дверями шахты. Они остаются открытыми после выхода пассажиров из кабины в течение небольшого промежутка времени. Затем реле времени замыкает свои контакты и подаёт питание на электродвигатель привода дверей кабины - двери закрываются. Лифт свободен и готов к работе по вызову, о чём свидетельствуют погасшие сигнальные лампочки вызывных аппаратов, установленные на каждом посадочном этаже.

1.2 Система управления лифтом

1.2.1 Роль системы управления в работе лифта

Наблюдаемая в последнее время тенденция к повышению этажности зданий в городах, а также к комфорту передвижения в лифтах ведёт к усложнению систем управления процессом передвижения. Благодаря развитию современных микропроцессорных систем управления данные задачи успешно решаются в настоящее время.

Современный лифт – это сложное электромеханическое устройство, работающее в полуавтоматическом режиме по установленной программе.

Программа работы лифта определяется действиями пассажиров, местонахождением и положением (свободна или занята) кабины и регламентируется при помощи системы управления лифтом.

Система управления лифтом должна решать задачи безопасного и комфортного передвижения пассажиров, для чего необходимо получать информацию о положении и скорости движения кабины с помощью различных датчиков, таких как датчики движения, веса, положения дверей и т.д. Обрабатывая получаемую информацию, система управления отдает команды различным лифтовым механизмам, которые, в свою очередь, выполняют какую-либо одну узкую задачу.

Система управления состоит из следующих основных частей:

- шкафа управления;
- контроллеров этажных, устанавливаемых в шахте лифта;
- контроллера приказного, установленного в посту приказов кабины лифта.

В шкафу управления располагается плата контроллера и микрочип для осуществления управления лифтом. Пассажиры используют кнопки вызова и кнопки приказа для передачи сигналов контроллеру. Контроллер принимает эти сигналы и определяет дальнейшие действия лифта согласно заданному алгоритму.

Таким образом, система управления является связующим звеном между пассажиром и лифтом. Она отвечает за корректное взаимодействие всех механизмов лифта и обработку сигналов по вызову и отправке, поступающих от пассажиров. В контроллере записан алгоритм работы, благодаря которому лифт выполняет свои задачи по перевозке людей наиболее оптимально, что уменьшает время, затрачиваемое на транспортировку людей на целевые этажи.

1.2.2 Зависимость системы управления от различных факторов

Система управления лифтом зависит от типа лифта, его назначения, системы электропривода и некоторых других факторов. Например, управление пассажирским лифтом отличается от управления грузовым лифтом, а управление пассажирским лифтом в жилых зданиях отличается от управления лифтом, установленным в административных зданиях.

Выделяют следующие основные признаки системы управления:

- тип командного аппарата, с помощью которого управляют лифтом;
- расположение командного аппарата относительно кабины;
- тип привода дверей кабины и шахты лифта;
- очередность выполнения вызовов свободной кабины;
- количество лифтов, управляемых по единой схеме.

По типу командного аппарата различают лифты с рычажным и кнопочным управлением. В лифтах с рычажным управлением пуск и остановка кабины на этажах выполняется путем ручного воздействия на рычажный командный аппарат, что установлен в кабине. На крайних этажах остановка кабины выполняется принудительным (с помощью отводок в шахте) воздействием на рычажный командный аппарат, который ставится в нулевое положение. Этот вид управления требует обязательного присутствия проводника в кабине.

В лифтах с кнопочным управлением пуск кабины и ее последующая остановка на требуемом этаже выполняется нажатием кнопки отправки или вызова пустой кабины, а остановка кабины на заданном этаже происходит автоматически.

По расположению командного аппарата относительно кабины различают лифты с внутренним управлением и лифты с наружным управлением. При внутреннем управлении командные аппараты для осуществления отправок располагают внутри кабины. При наружном управлении все командные аппараты (как для выполнения отправок, так и

для выполнения вызовов) располагают вне кабины на этажных площадках. Рычажное управление делают только внутренним, однако кнопочное управление бывает как наружным, так и внутренним. Управление малыми грузовыми лифтами и магазинными только наружное, а управление пассажирскими лифтами, наоборот, всегда внутреннее. Грузовые лифты делают как с наружным, так и с внутренним управлением.

По типу привода дверей и шахты различают лифты с автоматическим приводом и лифты с ручным приводом дверей. Современные пассажирские лифты делают с автоматическим открыванием дверей после прихода кабины на заданный этаж и автоматическим закрыванием дверей после нажатия на кнопку отправки. Такой привод дверей повышает удобство пользования лифтом и безопасность его работы. Ручной привод дверей применяют в малых грузовых, больничных лифтах и пассажирских лифтах старой конструкции.

По очередности выполнения вызовов кабины различают лифты, не выполняющие попутные вызовы, лифты, выполняющие попутные вызовы вниз, и лифты, выполняющие попутные вызовы как вверх, так и вниз. В лифтах, не выполняющих попутные вызовы предусмотрена система управления, в которой каждый следующий вызов пустой кабины на требуемый этаж может быть зарегистрирован и осуществлен только после окончания предыдущего приказа, т. е. когда лифт свободен. Такой способ управления применяют в пассажирских лифтах жилых зданий средней этажности, а также грузовых и больничных лифтах.

Лифты, выполняющие попутные вызовы при движении кабины вниз, устанавливают для перевозки пассажиров в жилых зданиях высокой этажности, так как эти лифты отличаются большой пропускной способностью. При таком способе управления этажными кнопками вызова регистрируются все вызовы с любого этажа, а выполняются они попутно при движении кабины вниз. Такой вид управления лифтами часто называют односторонним собирательным управлением по вызовам.

В лифтах, выполняющих попутные вызовы при движении кабины как вверх, так и вниз, кабина автоматически останавливается на всех этажах, где зарегистрированы вызовы, совпадающие с направлением движения кабины. Управление с выполнением попутных вызовов вверх и вниз предусматривают в пассажирских лифтах общественных и административных зданий и применяют двойные кнопки, на которые нанесены стрелки для указания направления вызова. Этот вид управления называют двусторонним собирательным управлением по вызовам.

По количеству лифтов, управляемых по единой схеме, различают лифты с индивидуальным (одиночным), с парным и с групповым управлением.

Работа лифта с индивидуальным управлением никаким образом не связана с работой другого лифта, даже если тот расположен рядом.

Лифты с парным управлением устанавливают в жилых и административных зданиях повышенной этажности. Система парного управления лифтами выполнена с таким расчетом, чтобы время ожидания кабин пассажирами было минимальным. Для этого работу обоих лифтов контролирует специальная программа, разработанная под парное управление лифтами. Вызов кабин обоих лифтов происходит от одного блока кнопок вызова, расположенного на каждом этаже.

В лифтах с групповым управлением, устанавливаемых в общественных или административных зданиях, система управления обеспечивает совместную программную работу трех, четырех лифтов и более.

Система группового управления выполнена так, чтобы максимально повысить производительность лифтов данной группы, уменьшить время ожидания кабины пассажирами и сократить ее холостой пробег. Программа, контролирующая работу таких лифтов, напоминает программу для лифтов с парным управлением, однако отличается большей сложностью, ввиду возросшего объема оптимизационных работ. Для всей группы лифтов на каждом этаже установлен только один блок кнопок вызова, причем на

промежуточных этажах устанавливают блок с двумя кнопками: одной для вызова кабины с целью движения вниз, другой – вверх.

Электрические схемы лифтов отличаются друг от друга системами электропривода и управления. Кроме того, электрические схемы с одной и той же системой управления могут различаться применением разных электрических аппаратов с одинаковым назначением. Например, электрическая схема лифта с этажным переключателем отличается от схемы лифта с шаговым этажным аппаратом, хотя назначение обоих аппаратов одно и то же.

Таким образом, система управления должна учитывать тип и назначение лифта, с которым она работает, что также подразумевает адаптацию алгоритма поведения лифта, записанного в контроллере.

1.2.3 Контроллер управления и алгоритм поведения лифта

Алгоритм поведения лифта – это программа, принимающая решения, связанные с управлением лифта, в различных ситуациях.

Алгоритм поведения записан в плате контроллера лифта (рис. 1.2). Он, как и плата контроллера, создается компанией-производителем и адаптируется под определенную конфигурацию лифта. Компания-производитель также зачастую выпускает контроллеры и для лифтов других типов, разрабатывая и записывая в плату подходящий алгоритм поведения.

Контроллер можно заменить на другой, подходящий под эту конфигурацию лифта. Это дает возможность компании-производителю совершенствовать контроллер, повышая эффективность и стабильность его работы, а также оптимизировать алгоритм поведения, использующийся в нем, так как покупателю для этого потребуется заменить лишь небольшую плату, не затрагивая при этом остальные составляющие лифта.

Рисунок 1.2 – Плата контроллера управления лифтом Октаграм L5L32

Алгоритм может учитывать множество факторов: не зря именно здесь сосредоточены все оптимизационные работы, особенно в лифтах с групповым управлением. Разработчики должны заложить в систему алгоритмы поведения для достижения максимальной эффективности и комфорта при перевозке пассажиров: какому лифту ехать на вызов, с какой скоростью, как лифты должны координировать свою работу, где находиться в режиме ожидания. Современные лифты также взвешивают пассажиров — и учитывают эту информацию, хотя и не выводят вес на общий дисплей.

Свои особенности есть в каждой стране. Например, в городе Мекка (Саудовская Аравия) нужно гарантировать, что лифты смогут пять раз в день достаточно быстро полностью очистить здание от людей, поскольку большинство из них идёт молиться. Соответственно, в этом городе приходится ставить в здания больше лифтов, чем в других, и, возможно, корректировать алгоритм поведения.

Нужно учитывать и количество пассажиров. В Азии в кабину заходит, в среднем, больше человек, чем в Европе или Нью-Йорке, потому что там у людей меньшее личное пространство.

Одно из изобретений американского математика Терезы Кристи – поддержка на уровне контроллера резервации пустого лифта после ввода пароля с цифровой клавиатуры. Например, один отель на Гавайях выдаёт такие коды сёрфингистам, чтобы они голые с досками не беспокоили

остальных постояльцев. В этом случае кабина гарантированно приедет к вам пустой и нигде не будет останавливаться по дороге.

Очевидно, что алгоритм поведения лифта играет важную роль в работе всей системы, и её эффективность в первую очередь зависит от того, какой алгоритм используется.

Глава 2 Анализ имеющейся версии программы

2.1 История разработки имеющейся версии программы

2.1.1 Место и команда разработки

Данная программа была разработана в рамках производственной практики в Тольяттинском офисе компании «ЭПАМ Системз». Команда разработки насчитывала 5 человек. Целью разработки было обучение работать в команде, а её результатом стала простенькая виртуальная система управления лифтом, получившая название «Elevator Control System», или коротко – ECL.

Команда много времени провела за изучением технологий и приемов, которые планировалось применять в разработке. Нелегким оказался и сам процесс работы в коллективе: ребятам пришлось научиться понимать друг друга, доносить свою мысль или идею до остальных, слушать и слышать своих товарищей. Из-за возникавших разногласий, финальная версия программы увидела свет намного позже, чем это планировалось изначально, но тем не менее, трудности были преодолены, а разработка успешно завершена.

Любой из членов команды мог бы написать эту программу ощутимо быстрее, если бы работал в одиночку, однако целью было именно обучение работы в коллективе, с применением соответствующих технологий, например системы контроля версий Subversion (SVN).

2.1.2 Определение требований к программе

Как правило, программное обеспечение разрабатывается для человека, или группы лиц, именуемых заказчиком. В качестве заказчика выступил руководитель практики от организации. После проведения опроса «заказчика», команда составила общие требования к программе:

1. Лифтов может быть несколько под управлением одной системы, но начнем с одного лифта. Все лифты управляются одним алгоритмом, «мозгом» лифта.

2. Имеются ограничения на грузоподъёмность и, следовательно, количество человек.
3. Каждый лифт имеет параметры и определённый набор оборудования:
    3.1. Кнопки вызова на этажах. Могут быть по одной на этаж или две на этаж (вверх и вниз соответственно).
    3.2. Кнопки отправки внутри лифтов. Кнопка отправки на этаж сразу запускает лифт на выбранный этаж. Лифт единовременно может обрабатывать только один вызов, игнорируя остальные.
4. Система работает в виртуальном времени, что позволяет ускорять процесс для быстрого сбора статистики. В первой версии программы, виртуальное время будет синхронизировано с реальным, т.е. события будут происходить с нормальной скоростью, что позволит тестировать систему через графический интерфейс.
5. Каждое виртуальное событие (к примеру, нажатие кнопки вызова или открытие дверей) занимает определённое виртуальное время. Длительность каждого события хранится в специальном справочнике. Стоит учесть, что во время задержки, вызванной одним событием, может прийти другое. К примеру, пока двери открываются, кто-то может вызвать лифт.
6. Систему рекомендовано разбить на четыре модуля:
    6.1. Симулятор железа. Представляет собой лифтовое «железо»: кабинку, двери, кнопки и т. д. Отсылает события типа «открываю дверь», «дверь открыта» и «принял нажатие на кнопку».
    6.2. Симулятор людей. Задаются временем прихода, терпением, стремлением попасть на определённый этаж, причём в разное время суток этажи отбытия и прибытия меняются. К

примеру, в жилом доме утром почти все едут на первый этаж, а вечером – наоборот.

6.3. Симулятор контроллера лифта. Программа, выполняющая контроль за отдельным лифтом. Управляет его железом, не допускает внештатных ситуаций, вроде открытия дверей на ходу.

6.4. Слой доступа к данным, осуществляющий полное взаимодействие с БД.

7. Все события, происходящие в системе, записываются в лог с пометкой времени и выводятся на экран.

2.1.3 Проектирование структуры программы

Определившись с требованиями, команда приступила к проектированию будущей программы. Были выделены основные модули программы, а также их функции.

User Interface (UI, пользовательский интерфейс).

В программе присутствует UI для эмуляции нажатия кнопок в лифте и на этажах. Также в UI отображается лог действий лифта (перемещение, открытие/закрытие дверей, вызов лифта на этажах, нажатие кнопок внутри лифта), текущий этаж, текущее состояние лифта (в движении/в простое) и дверей (открыты/закрыты).

Business Access Layer (BAL, слой бизнес-логики).

UI сообщает о нажатии кнопок слою бизнес-логики. Слой логики выполняет необходимые действия, отправляя команды остальным модулям. Слой логики является «связующим звеном» между всеми остальными модулями. Его основные задачи: обработка нажатия кнопок из UI, сбор данных для логирования (ведения журнала событий). Часть BAL, ответственная за логирование, подписывается на все события в системе.

Все действия системы выполняются за определенное кол-во реального времени. Предлагается использовать эмуляцию реального времени: «тик», равный 0.1 сек. Каждый тик обрабатывается всеми модулями

последовательно. Таким образом, следующий тик не будет сгенерирован, пока не закончится обработка предыдущего. При этом есть возможность обрабатывать «тики» друг за другом с максимальной скоростью, ускоряя виртуальное время, или же установить задержку, равную 0,1 секунде, что синхронизирует виртуальное время с реальным.

Data Access Layer (DAL, слой доступа к данным).

Для взаимодействия с БД используется слой доступа к данным, к которому обращается BAL при необходимости взаимодействия с БД.

Elevator Control System (ECL, слой управления лифтом).

Для детального управления лифтом используется слой управления лифтом, который выполняет контроль над «железом», выполняя элементарные проверки (чтобы лифт не ехал с открытыми дверьми, останавливать лифт на нужном этаже и т.д.).

Hardware (слой симуляции железа).

ECL управляет непосредственно оборудованием лифта (Hardware). Оборудование умеет выполнять лишь примитивные операции: открыть/закрыть двери, двигать лифт вверх/вниз до команды СТОП. Также к оборудованию относятся датчики лифта: состояние лифта (едет/стоит), дверей (открыты/закрыты), перегрузки (пока не используется), информация о текущем этаже.

Была составлена примерная схема модулей и их взаимодействия друг с другом. Полученная схема изображена на рис. 2.1.

Таким образом, пользовательский интерфейс общается со слоем бизнес-логики через интерфейс *IUser*, отвечающий за функционал, связанный с пользователем (пассажиром) лифта. Все данные, связанные с лифтом (его тип, настройки, характеристики и т.п.) передаются внутри программы реализацией интерфейса *IElevator*. BAL является связующим звеном в системе, ECL контролирует железо лифта, а DAL «общается» с базой данных.

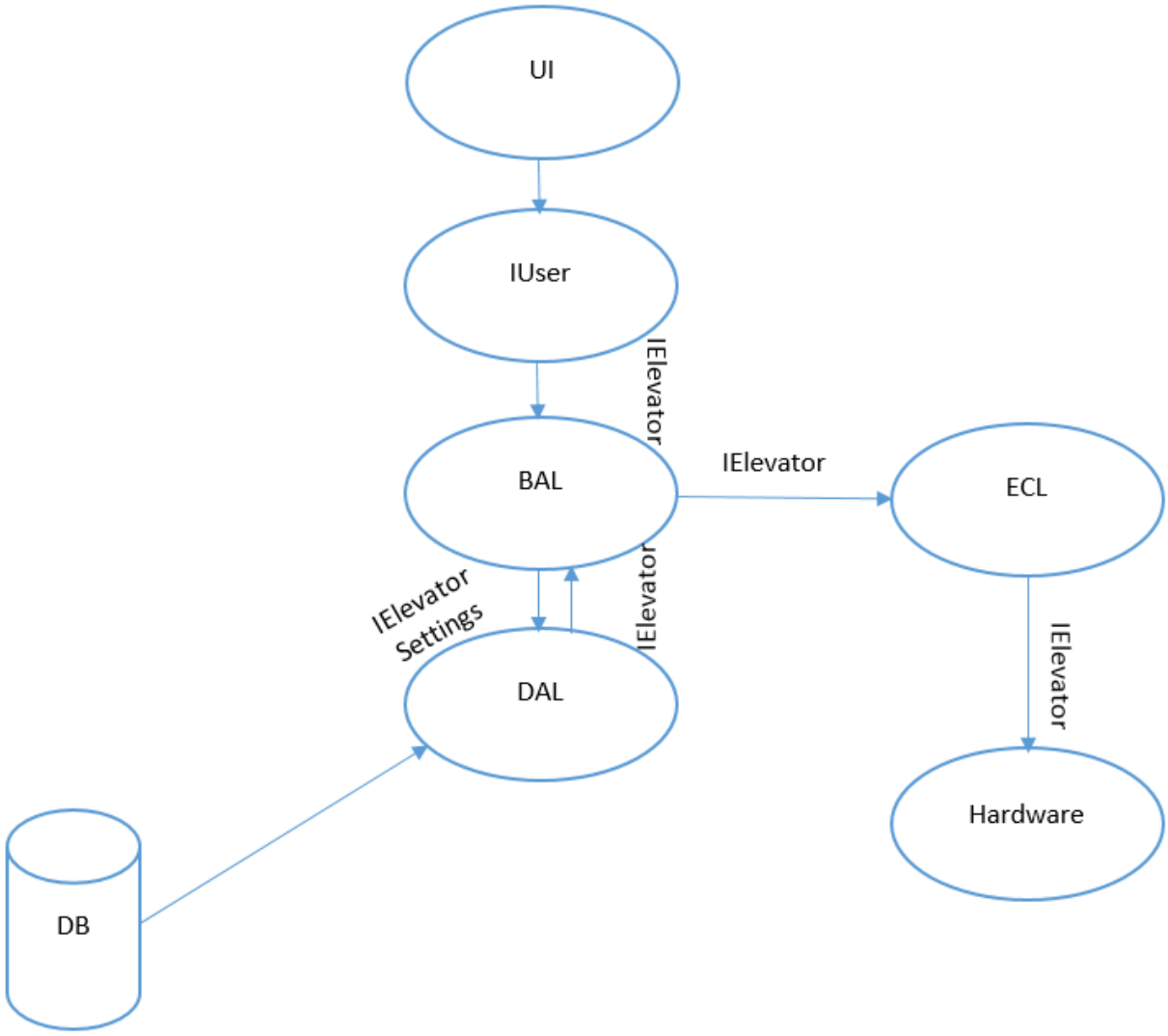

Рисунок 2.1 – Схема разрабатываемой программы

2.1.4 Проектирование базы данных

Прежде чем приступать к созданию БД, необходимо её спроектировать. Команда проанализировала будущую программу и вывела список сущностей, которые будут представлять собой таблицы базы данных. Этот список представлен в таблице 2.1.

Таблица 2.1 – Сущности БД

№ П/П	Название сущности	Описание сущности	Название реквизита	Тип реквизита	Признак ключа
1	**Вид события**	Перечень видов событий	**Код вида**	**Целочисленный, счетчик**	**Первичный, простой**
			Название вида	Строковый(100)	
2	**Здание**	Перечень обслуживаемых зданий	**Код здания**	**Целочисленный, счетчик**	**Первичный, простой**
			Адрес	Строковый(250)	
			Количество этажей	Целочисленный	
3	**Подъезд**	Список подъездов каждого из зданий	**Код подъезда**	**Целочисленный, счетчик**	**Первичный, простой**
			Код здания	***Целочисленный***	***Внешний, простой***
			Номер подъезда	Целочисленный	
4	**Лифт**	Перечень обслуживаемых лифтов	**Код лифта**	**Целочисленный, счетчик**	**Первичный, простой**
			Код подъезда	***Целочисленный***	***Внешний, простой***
			Максимальное количество пассажиров	Целочисленный	
5	**Событие**	Перечень событий, возникающих в процессе функционирования лифта	**Код события**	**Целочисленный, счетчик**	**Первичный, простой**
			Код лифта	***Целочисленный***	***Внешний, простой***
			Код вида события	***Целочисленный***	***Внешний, простой***
			Дата/время события	Дата/время	
			Текущий этаж	Целочисленный	

Определившись с сущностями базы данных, команда составила схему их взаимодействия. Эта схема продемонстрирована на рис. 2.2.

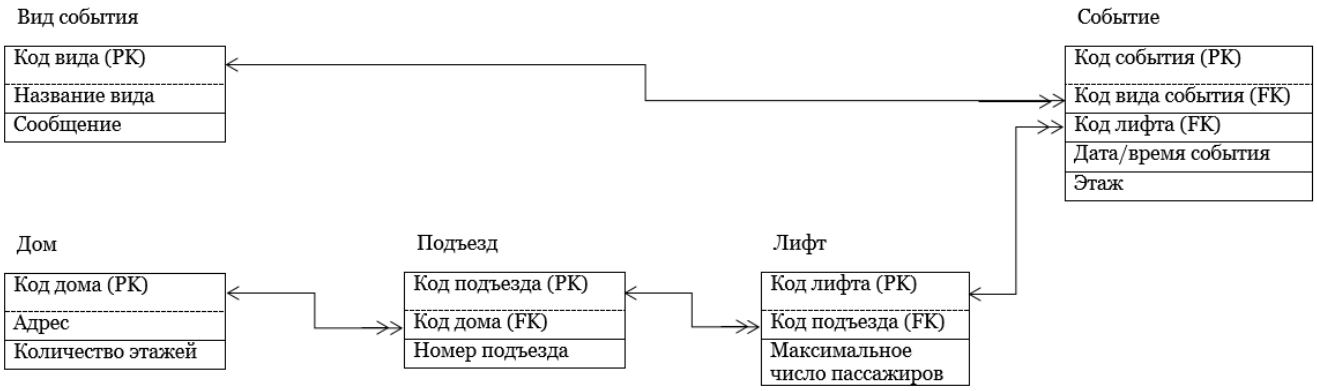

Рисунок 2.2 – Схема базы данных

В целом, схемы и списка сущностей достаточно для создания БД, однако удобнее будет предварительно сформировать «контрольный пример» – таблицы с начальными данными, которые будут в базе сразу после её создания. Начальные данные для одной из сущностей представлены в таблице 2.2.

Таблица 2.2 – Перечень видов событий

Код вида	Название
1	Начало открытия дверей
2	Двери открыты
3	Начало закрытия дверей
4	Двери закрыты
5	Начало движения
6	Остановка

Команда составила таблицы и для остальных сущностей БД.

2.1.5 Проектирование симулятора контроллера лифта и алгоритма поведения

В то время, как контроллер управляет железом лифта, команды ему отдает программа, реализующая алгоритм поведения лифта.

Алгоритм поведения определяет, какое следующие действие будет выполнять лифт. Он направляет лифт на этажи, на которые он был вызван или отправлен из кабины, определяет, будут ли остановки по пути и т.п.

Контроллер, в свою очередь, отдает команды железу лифта, и отвечает за последовательное и безопасное выполнение примитивных команд, например: открыть/закрыть двери, привести в движение или остановить двигатель. Он также следит, чтобы не возникали внештатные ситуации, вроде открытия дверей во время движения.

Команда составила подробную диаграмму поведения лифта в простых ситуациях, иллюстрирующую также взаимодействие компонентов системы в процессе работы. Эта диаграмма очень полная и объемная. Небольшая её часть, иллюстрирующая работу лифта при вызове на этаж, приведена на рисунке 2.3.

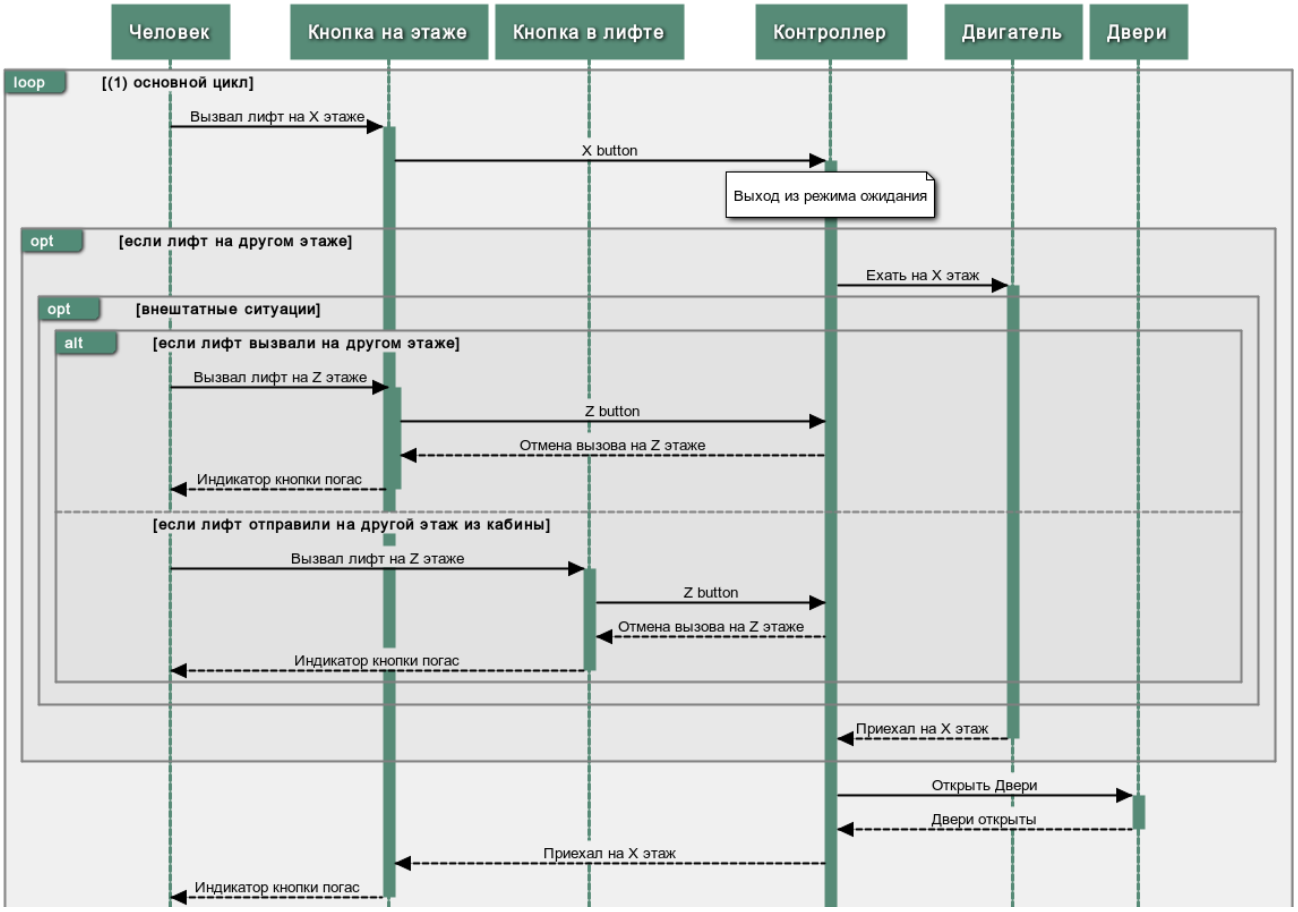

Рисунок 2.3 – Диаграмма поведения лифта при вызове на этаже

2.1.6 Окончание разработки и подведение итогов

Закончив разработку программы, команда подвела итоги прошедших дней, составив список реализованного функционала. Итак, первый релиз включает в себя:

1. Перемещение лифта по этажам, открытие и закрытие дверей кабины лифта.
2. Вызов лифта на заданный этаж с помощью кнопок на этажах.
3. Отправка лифта на заданный этаж с помощью кнопок внутри кабины лифта.
4. Эмуляция виртуального времени работы системы.
5. Сохранение событий системы в базе данных (логирование).
6. Эмуляция действий двух человек, взаимодействующих с лифтом, реализованная посредством пользовательского интерфейса.

Стоит отметить, что релиз не включает в себя модуль симуляции людей. Весь остальной запланированный к реализации функционал присутствует. Окно программы изображено на рис. 2.4.

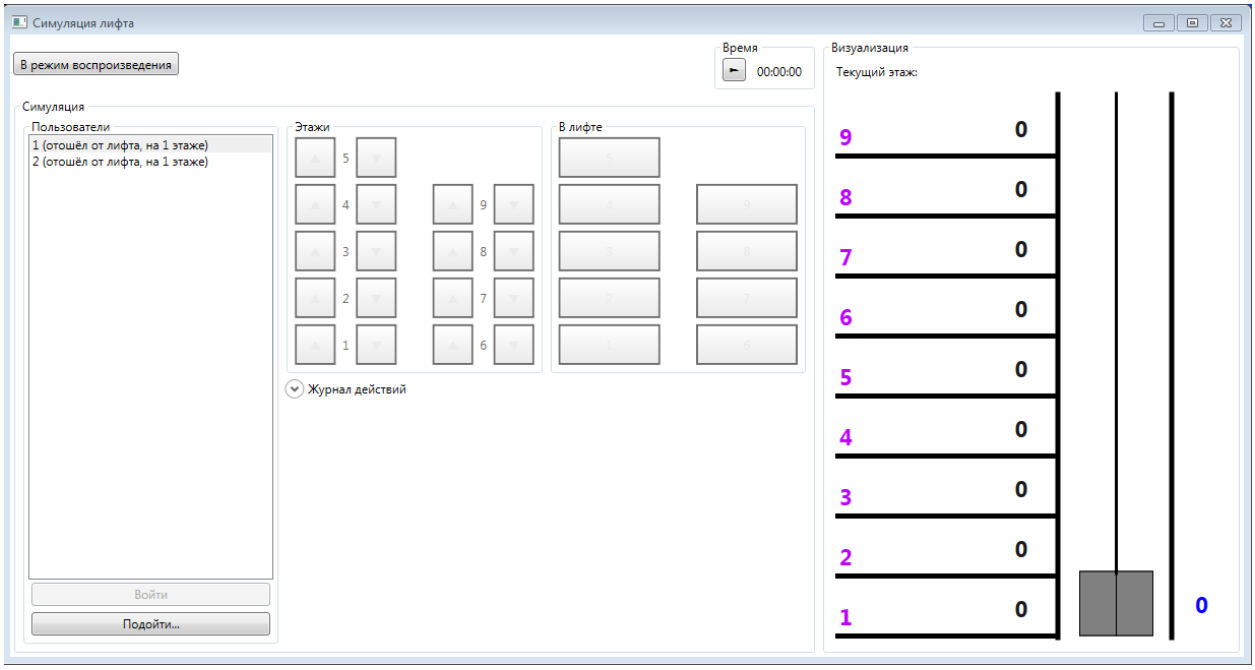

Рисунок 2.4 – Окно программы

2.2 Определение задач по улучшению программы

Итак, ранняя версия программы позволила запустить и протестировать основные механизмы системы, которые, хоть и разрабатывались разными людьми, теперь работают как единое целое. Тем не менее, систему необходимо существенно доработать.

Основной идеей является совершенствование алгоритма поведения лифта, а также иллюстрация преимуществ улучшенного алгоритма перед текущим. Для выполнения этих задач требуется:

- создать улучшенную версию алгоритма поведения лифта;
- реализовать «проигрыватель», который сможет воспроизводить события, ранее происходившие с лифтом, читая их из базы данных (удобен для тестирования симулятора людей);
- реализовать модуль симуляции людей, который сможет воспроизвести поведение людей при взаимодействии с лифтом в течение определенного периода времени (например, за сутки);
- реализовать модуль статистики для сбора различного рода статистических данных на основе событий лифта, взятых из БД;
- воспроизвести одну и ту же ситуацию (например, сутки времени в жилом доме) с применением различных версий алгоритма, собрать статистику об эффективности работы лифтов в каждом случае, сравнить полученные данные, сделать выводы.

Глава 3 Разработка нового функционала программы

3.1 Изменения в базе данных

Для реализации нового функционала программы требуются дополнительные сущности в базе данных, а также новые поля у существующих сущностей (табл. 3.1).

Таблица 3.1 – Сущности обновленной базы данных

№П/П	Название сущности	Описание сущности	Название реквизита	Тип реквизита	Признак ключа
1	**Сессия**	Перечень сеансов работы с приложением	**Код сессии**	**Целочисленный, счетчик**	**Первичный, простой**
			Дата начала сессии	Дата/время	
2	**Вид события**	Перечень видов событий	**Код вида**	**Целочисленный, счетчик**	**Первичный, простой**
			Название вида	Строковый(100)	
3	**Здание**	Перечень обслуживаемых зданий	**Код здания**	**Целочисленный, счетчик**	**Первичный, простой**
			Адрес	Строковый(250)	
			Количество этажей	Целочисленный	
			Высота этажа	Число с плавающей точкой	
4	**Подъезд**	Список подъездов каждого из зданий	**Код подъезда**	**Целочисленный, счетчик**	**Первичный, простой**
			Код здания	***Целочисленный***	***Внешний, простой***
			Номер подъезда	Целочисленный	
5	**Лифт**	Перечень обслуживаемых лифтов	**Код лифта**	**Целочисленный, счетчик**	**Первичный, простой**
			Код подъезда	***Целочисленный***	***Внешний, простой***
			Код конфигурации	***Целочисленный***	***Внешний, простой***
6	**Событие**	Перечень событий, возникающих в процессе функционирования лифта	**Код события**	**Целочисленный, счетчик**	**Первичный, простой**
			Код сессии	***Целочисленный***	***Внешний, простой***
			Код лифта	***Целочисленный***	***Внешний, простой***
			Код вида события	***Целочисленный***	***Внешний, простой***
			Дата/время события	Дата/время	
			Этаж	Целочисленный	

Продолжение таблицы 3.1

№П/П	Название сущности	Описание сущности	Название реквизита	Тип реквизита	Признак ключа
7	**Конфигурация**	Перечень конфигураций лифтов	**Код конфигурации**	**Целочисленный, счетчик**	**Первичный, простой**
			Поддержка мультивызова	Логический	
			Наличие кнопок вверх/вниз	Логический	
			Энергопотребление в режиме ожидания	Число с плавающей точкой	
			Коэффициент энергопотребления при движении	Число с плавающей точкой	
			Коэффициент энергопотребления при работе дверей	Число с плавающей точкой	
			Время движения дверей	Целочисленный	
			Время преодоления одного этажа	Целочисленный	
			Задержка перед открытием дверей	Целочисленный	
			Задержка перед закрытием дверей	Целочисленный	
			Задержка перед отправкой	Целочисленный	
			Грузоподъемность	Целочисленный	

Появились две новые сущности: сессия и конфигурация. Первая является сеансом работы с программой, на его основе будет проводиться группировка событий для передачи их в проигрыватель или сбора статистики. Вторая содержит в себе подробную конфигурацию лифтов. Эти данные пригодятся нам для точной настройки лифтов и подсчета статистических показателей.

Изменились и существующие сущности. Так, «Здание» теперь имеет поле «высота этажа», что пригодится нам при подсчете пробега лифта; «Лифт» обзавелся полем «Код конфигурации», с помощью которого будет

указываться конфигурация этого лифта; а в сущности «Событие» появилось поле «Код сессии» для привязки всех событий к какому-то сеансу использования программы.

Составим схему обновленной базы данных (рис. 3.1):

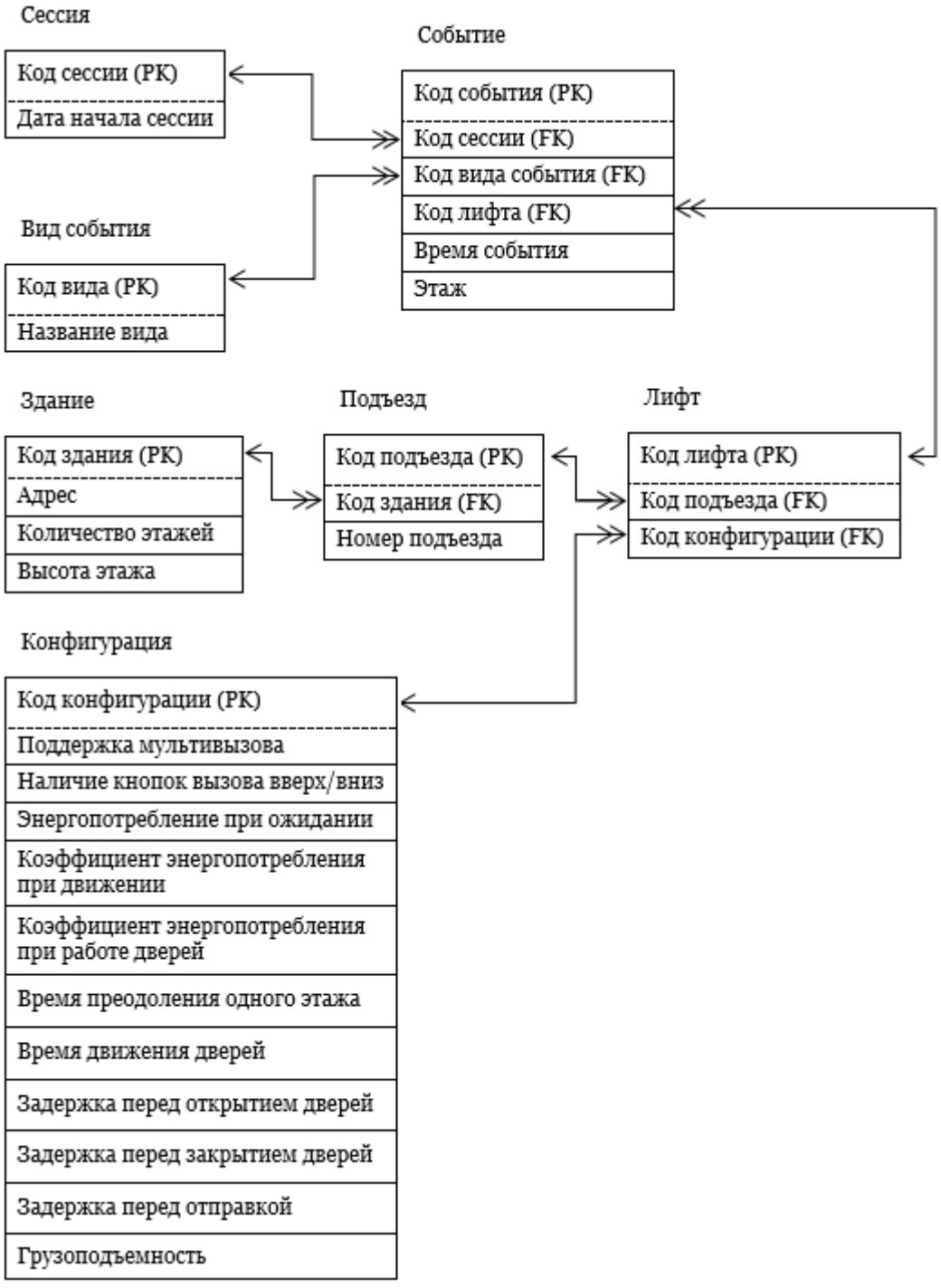

Рисунок 3.1 – Схема обновленной базы данных

Теперь можно приступать к добавлению в программу нового функционала.

3.2 Создание улучшенной версии алгоритма поведения

С текущим алгоритмом поведения лифт способен обрабатывать лишь один вызов единовременно, что, непременно, создаст трудности при перевозке большого кол-ва пассажиров. Следовательно, алгоритм необходимо улучшить.

Текущий алгоритм реализован в классе `MovementLogic`, который, в свою очередь, реализует интерфейс `IMovementLogic`. Переименуем класс `MovementLogic` в `MovementLogic_Simple`, и создадим новый класс `MovementLogic_Queue`, реализующий тот же интерфейс, для программирования улучшенного алгоритма. Он будет характеризоваться поддержкой очереди вызовов, т.е. лифт станет «запоминать» этажи, на которые его вызвали или отправили, и обходить их в порядке очереди. Более того, лифт начнет останавливаться на вызванных этажах при движении вниз, чтобы собирать пассажиров, желающих спуститься на первый этаж, вместо того, чтобы обрабатывать каждый вызов в отдельности. Небольшой фрагмент кода с использованием очередей изображен на рис. 3.2.

```
if (_info.ButtonHandler.CheckFloorButton(floor) && !_callQueue.Contains(floor))
{
    _callQueue.Add(floor); //Если лифт вызвали на этаже, и этажа нет в очереди, то добавляем его туда.
}
if (_info.ButtonHandler.CheckInsideButton(floor) && !_sendQueue.Contains(floor))
{
    _sendQueue.Add(floor); //Если лифт отправили на этаж, и этажа нет в очереди, то добавляем его туда.
}
SetCallingFloor(); // Задаем лифту следующий этаж для движения.

// Если обрабатываемый этаж является следующим по пути движения лифта (и лифт сейчас движется), то просим его остановиться на следующем этаже.
if (Math.Abs(_info.CurrentFloor - floor) == 1 && _movementDirection == (_info.CurrentFloor < floor ? Direction.Up : Direction.Down))
{
    StopOnNextFloor(floor);
}
```

Рисунок 3.2 – Фрагмент кода с использованием очередей вызовов

Поместив функционал двух алгоритмов в разные классы, мы получили возможность использовать тот или иной алгоритм поведения, указывая соответствующую настройку для лифта. Добавим в таблицу конфигурации лифтов в базе данных новое поле – `AllowMultipleCall`, ответственное за поддержку нескольких вызовов лифта единовременно (очередь вызовов).

Теперь, в зависимости от конфигурации лифта, можно «подставить» класс с подходящим алгоритмом поведения в контроллере лифта (рис. 3.3).

```
if (!elevator.Configuration.AllowMultipleCall)
{
    MovementLogic = new MovementLogic_Simple(this);
}
else
{
    MovementLogic = new MovementLogic_Queue(this, resetTimer);
}
```

Рисунок 3.3 – Использование подходящего алгоритма поведения лифта

Проверяем новый алгоритм «вручную», используя режим симуляции в графическом интерфейсе (рис. 3.4):

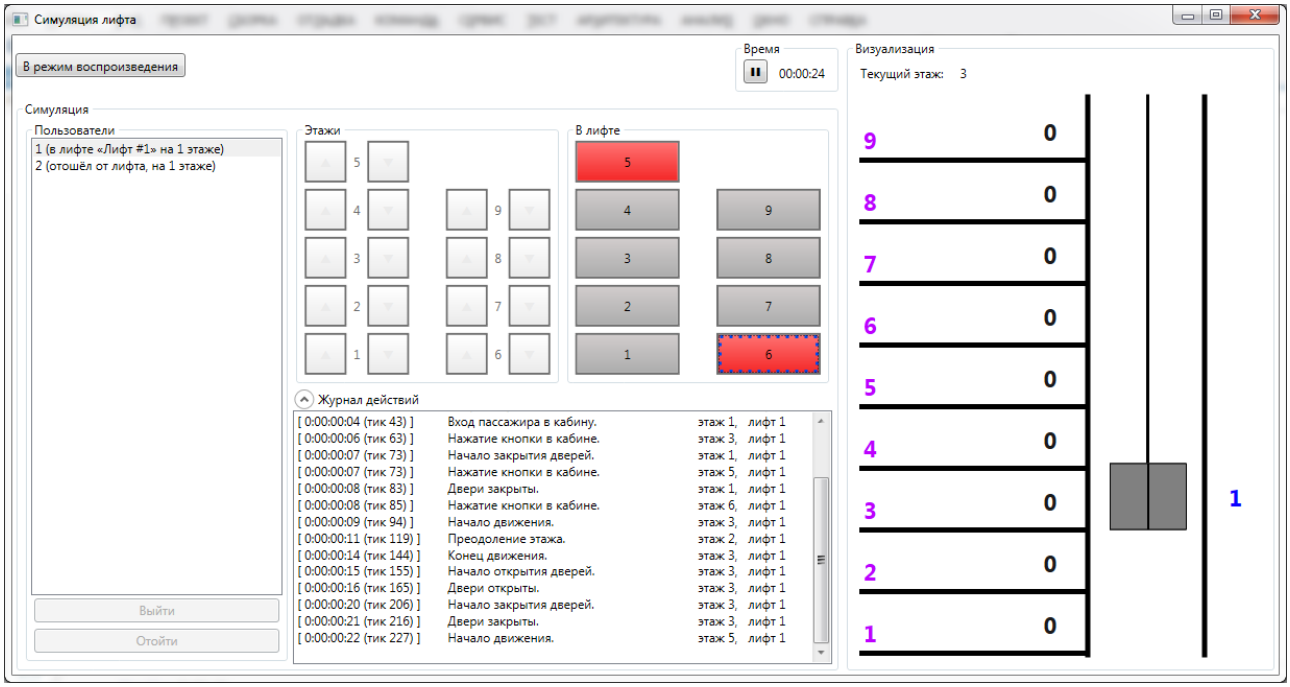

Рисунок 3.4 – Проверка нового алгоритма

Заходим первым пассажиром в лифт на 1 этаже и отправляем его последовательно на 3 этажа: третий, пятый и шестой. В кабине загораются соответствующие кнопки. Добравшись до третьего этажа, лифт останавливается на нём. Кнопка «3» гаснет. Спустя несколько секунд, лифт продолжает движение к оставшимся этажам. Аналогичным образом проверяем «обход» этажей при движении вниз.

3.3 Создание «проигрывателя» событий из базы данных

Для тестирования будущего модуля симуляции людей необходимо иметь возможность просмотреть события, происходившие с лифтом, и записанные в базу данных. Для этого можно немного доработать текущий графический интерфейс. Он уже обладает функционалом отображения событий, приходящих из слоя бизнес-логики (BAL): информация о взаимодействии пассажиров с лифтом (в интерфейсе) уходит в слой BAL, и далее – к контроллеру. Контроллер формирует событие, как реакцию на действия пассажиров, и отправляет его обратно в BAL, откуда оно уходит в двух направлениях: в DAL для записи в БД и в графический интерфейс для визуального отображения.

Дополним интерфейс, добавив режим воспроизведения и кнопку для перехода в него (рис. 3.5).

```
private void SwitchMode_Click(object sender, RoutedEventArgs e)
{

    _UI.BAL.LoopingTicker.Pause = true;

    if (mode == WorkMode.Simulation)
    {
        var result = MessageBox.Show(
            this,
            "Переключиться в режим воспроизведения?\nВНИМАНИЕ: Симуляция будет сброшена.",
            "Переключение режимов",
            MessageBoxButton.YesNo,
            MessageBoxImage.Exclamation,
            MessageBoxResult.No
        );

        if (result == MessageBoxResult.Yes) //переходим в режим Плеера
        {
            if (player == null)
            {
                player = new Player(_UI.BAL.LoopingTicker);
                loggers = new ILogger[2] { _UI.BAL.BroadcastLogger, player };
                player.OnLog += new EventHandler<LoggingEventArgs>(LogEventArrivedPlayer);
            }


            mode = WorkMode.Playback;
            SwitchMode.IsEnabled = false;
            SwitchMode.Content = "В режим симуляции";
            UserGroup.Visibility = Visibility.Hidden;
            ShowStatistics.Visibility = Visibility.Visible;
```

Рисунок 3.5 – Обработка нажатия кнопки перехода в режим воспроизведения

Теперь, необходимо передавать события в интерфейс, как это обычно и происходит, но не новые события (из контроллера), а старые, уже имеющиеся в базе данных. Для этого напишем новый класс `Player` в слое бизнес-логики, подпишем его на «тики» внутри программы, и из него станем передавать события в интерфейс (рис. 3.6).

```
public void Tick()
{
    if(!stopped)
    {
        LocalCurrentTick++;//Прибавляем локальный счетчик тиков(каждый тик)
        Event pendingEvent = _cursor.Current;
        if(pendingEvent != null)
        {
            if (pendingEvent.EventTime <= _ticker.TicksToMilliseconds(LocalCurrentTick) && OnLog != null)
            {
                OnLog(this, new LoggingEventArgs { eventToSave = pendingEvent });
                _cursor.MoveNext();
            }
            //Если разница между временем следующего лога и текущего времени > 5000(мс)
            if ((pendingEvent.EventTime - _ticker.TicksToMilliseconds(LocalCurrentTick)) > 5000)
            {
                if (jump_interval == true) // Если включен пропуск пустых интервалов
                {
                    _ticker.SleepDuration = 1;
                }
            }
            else
            {
                _ticker.SleepDuration = (uint)SpeedSleepDuration;
            }
        }
        else
        {
            stopped = true;
            if (OnPlaybackEnd != null)
                OnPlaybackEnd(this, EventArgs.Empty);
        }
    }
}
```

Рисунок 3.6 – Основной метод класса Player

А вот передачу событий в базу данных выполнять не следует. В проигрывателе мы только отображаем на экране уже имеющиеся события, но не пишем их повторно в БД (рис. 3.7).

Настроив отправку событий в интерфейс, не забываем выключить обработку нажатий кнопок этажей лифта в режиме воспроизведения – мы только просматриваем события, но не управляем лифтом! Реализуем возможность выбора сессии, события которой будут воспроизводиться,

добавляем кнопки для ускорения воспроизведения, и проигрыватель готов. Результат изображен на рис. 3.8.

```
ссылок 9
public EventHandler<LoggingEventArgs> OnLog { get; set; }

ссылок 23
public void Log(int eventTypeId, Elevator elevator, byte currentFloor)
{
    // Так как мы всего лишь проигрываем предзаписанные события, обратная связь
    // не только бесполезна, но и вредна, так как может вызвать рассинхронизацию
    // журнала и отображения.
}
```

Рисунок 3.7 – Перегруженный метод Log класса Player с пустой реализацией

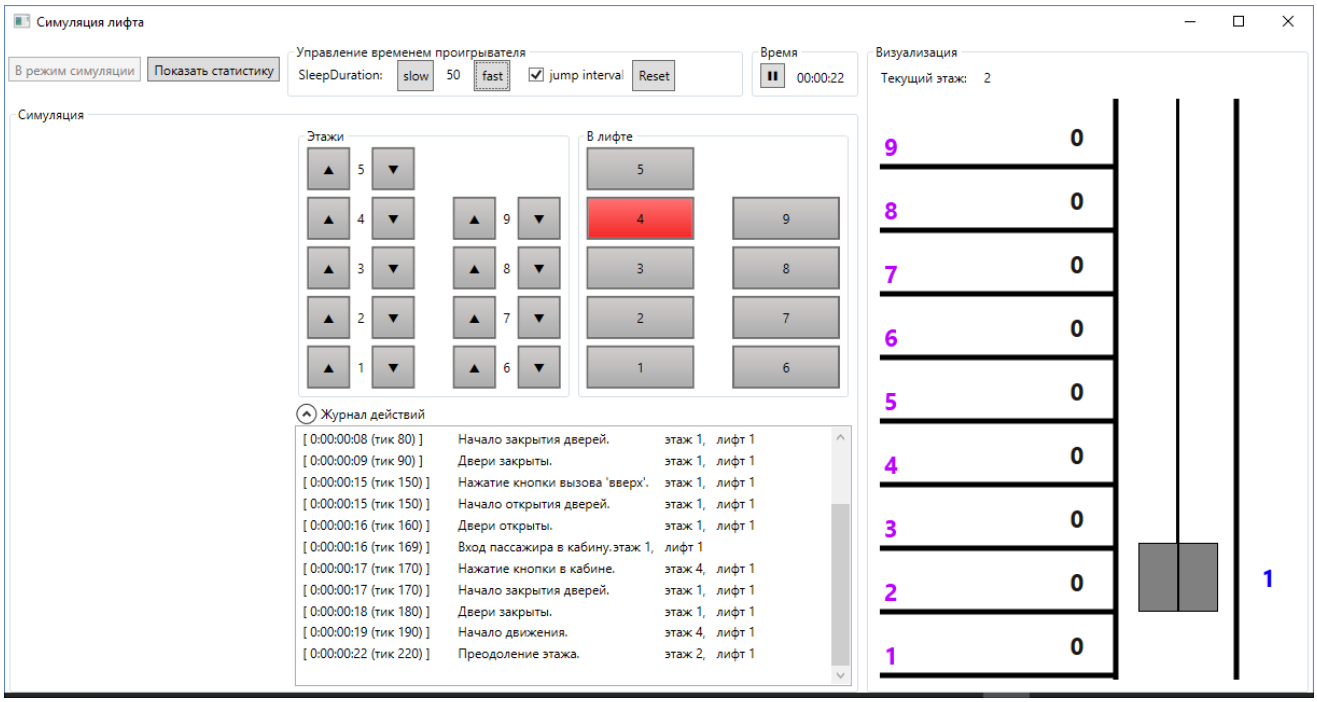

Рисунок 3.8 – Воспроизведение событий из базы данных.

Скорость воспроизведения регулируется кнопками в верхней части окна. По умолчанию включена функция перемотки виртуального времени: если до следующего события больше 5 секунд, программа может мгновенно «прыгнуть» к этому событию, вместо обычного простоя. Очевидно, что визуально режим воспроизведения очень похож на режим симуляции лифта, однако во время воспроизведения вы не можете управлять лифтом или пассажирами.

3.4 Создание модуля симуляции людей

Несмотря на то, что программа предоставляет возможность с помощью графического интерфейса имитировать взаимодействие двух людей с лифтом, таким способом невозможно воспроизвести более масштабные ситуации, например пользование лифтом в жилом доме. Даже если предположить, что количество жителей этого дома невелико, и лифтом не будут пользоваться более двух человек единовременно (графический интерфейс не позволяет управлять более чем двумя пассажирами), тем не менее воспроизведение событий, происходящих с лифтом на протяжении, например, двух часов, потребовало бы от пользователя в течение двух часов реального времени находиться перед монитором и «направлять» наших пассажиров через интерфейс. На помощь придет модуль симуляции людей.

Графический интерфейс хорош для тщательного воспроизведения конкретных ситуаций с хорошей визуализацией происходящих событий, а модуль, имитирующий поведение людей, лучше подойдет для быстрого воспроизведения ситуации, приближенной к реальной, для последующего анализа и сравнения статистики.

Имитация поведения людей при взаимодействии с лифтом будет происходить по той же схеме, что и взаимодействие с лифтом двух пассажиров из графического интерфейса (рис. 3.9). Информация о всех действиях людей будет через бизнес-логику попадать в контроллер лифта. Контроллер будет реагировать на эти действия и отправлять события в бизнес-логику, на которые модуль симуляции людей может подписаться, как это делает графический интерфейс. Получая информацию о событиях, происходящих с лифтом, модуль сможет реагировать на них, имитируя соответствующее поведение пассажиров.

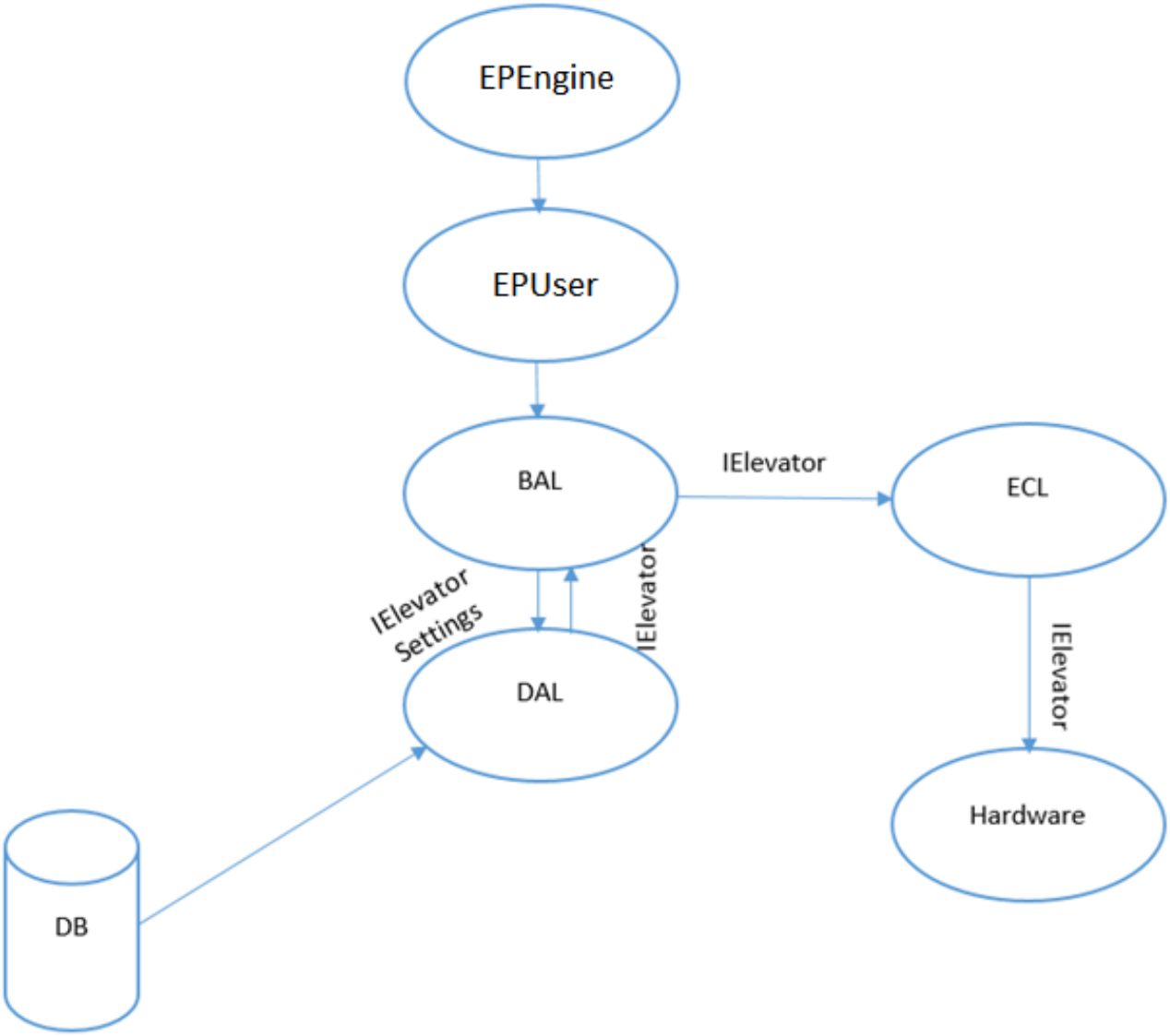

Рисунок 3.9 – Схема взаимодействия модулей программы при запуске симуляции людей

Создадим новый модуль, который заменит графический интерфейс, но с функциональной точки зрения будет работать по его аналогии. Пусть это будет консольный проект *EmulationPeople*. Его основной класс `EPEngine` будет содержать в себе весь алгоритм появления людей и обработки событий, приходящих из бизнес-логики, а вспомогательный класс `EPUser` будет отвечать за пассажиров, пользующихся лифтом.

Пусть будет выполняться имитация поведения людей в жилом доме в будний день. Определим, как часто люди пользуются лифтом в разное время суток (рис 3.10).

```
/// <summary>
/// Кол-во жителей на этаже дома (число от которого мы "отталкиваемся" при генерации людей).
/// </summary>
private const uint INHABITANTS_PER_FLOOR = 12;

/*Зависимость кол-ва людей от времени суток:
0) 00:00 - 03:00 - малое кол-во человек (3% in/1% out), преимущественно приходят;
1) 03:00 - 06:00 - малое кол-во человек (2%/1%), преимущественно приходят;
2) 06:00 - 09:00 - большое кол-во человек (5%/55%), почти все уходят;
3) 09:00 - 12:00 - среднее кол-во человек (7%/20%), преимущественно уходят;
4) 12:00 - 15:00 - среднее кол-во человек (10%/10%), как приходят, так и уходят;
5) 15:00 - 18:00 - среднее кол-во человек (20%/5%), преимущественно приходят;
6) 18:00 - 21:00 - большое кол-во человек (40%/3%), почти все приходят;
7) 21:00 - 00:00 - среднее кол-во человек (13%/6%), преимущественно приходят.
+ 09:00 - 00:00 - случайные перемещения между этажами (гости).
*)  В определенный промежуток времени мы генерируем определённый процент людей.
    Важно заметить, что в сумме столбец in (люди приходят) по всем промежуткам времени даёт 100%
    Аналогично со столбцом out (люди уходят)
    Исходя из такой логики мы получаем, что в среднем, за сутки, у нас 100% людей приходят и 100% уходят. */

/// <summary>
/// Массив с процентами для генерации людей в определённый промежуток времени.
/// </summary>
private readonly uint[,] PROBABILITES =
{
    { 3, 1 },
    { 2, 1 },
    { 5, 55 },
    { 7, 20 },
    { 10, 10 },
    { 20, 5 },
    { 40, 3 },
    { 13, 6 }
};
```

Рисунок 3.10 – Установка вероятностей для генератора людей

Рассмотрим подъезд жилого дома с установленным в нём лифтом. Пусть на каждом этаже имеется 4 квартиры с 3 жителями в каждой, что даёт нам 12 жителей на этаж. В будний день жильцы захотят пойти на работу в утренние часы, и вернуться домой в вечерние. Зададим временные промежутки по 3 часа, начиная с полуночи. Получим 7 промежутков времени с разной активностью пассажиров. В ночные часы активность крайне низкая, но она резко возрастает утром. В это время жильцы обычно уходят на работу, т.е. используют лифт преимущественно для спуска с верхних этажей на первый. Днём активность снижается: небольшое кол-во жителей как приходят домой, так и уходят. К вечеру возникает ситуация, обратная утренней: жильцы подъезда возвращаются с работы домой, используя лифт, чтобы подняться с первого этажа на какой-либо другой. Ближе к ночи активность вновь снижается, т.к. почти все люди уже вернулись домой. Более

того, с утра и до позднего вечера лифтом изредка могут пользоваться люди, не проживающие в подъезде – назовём их «гости».

Определившись с вероятностями, напишем метод для генерации людей на этажах, с учетом, что вызываться он будет каждую виртуальную секунду (рис. 3.11).

```
/// <summary>
/// Метод генерации людей
/// *Срабатывает каждую вертуальную секунду
/// </summary>
ссылка 1
private void GeneratePeople()
{
    //В первую очередь нам нужно решить: генерировать ли человека в текущую виртуальную секунду? именно это и  решает следующее условие
    //В зависимости от периода времени нам нужно генерировать разный процент людей, в функции RANDOM меняем "верхнюю границу" генерации
    if (_rand.Next(1, (int)(10800 / //10800(число секунд в 3 часах(именно такие у нас промежутки)) мы делим на
        //Количество людей на этаже * (на количество этажей в здании - 1(так как люди первого этажа не пользуются лифтом)) и это все мы умножаем на
        (INHABITANTS_PER_FLOOR * (_selectedPorch.Building.FloorsNumber - 1) *
        //Сумму (% людей который приходит в данный период + % людей который уходит) / 100
        (PROBABILITES[(int)_period, 0] + PROBABILITES[(int)_period, 1]) / 100))) == 1)
    //По данной формуле мы получаем вероятность генерации человека в данную секунду
    {
        //Теперь мы должны решить: человек, который сгенерируется он приходит(появляется на 1 этаже) или уходит(появляется на 2-N этаже)
        //Мы сново меняем верхнию границу рандома: в этот раз это сумма (% людей который приходит в данный период + % людей который уходит)
        //И взависимости от того в какой промежуток попадет рандом, мы генерим человека.
        if (_rand.Next(1, (int)(PROBABILITES[(int)_period, 0] + PROBABILITES[(int)_period, 1])) <= PROBABILITES[(int)_period, 0])
        {
            //человек появляется на первом этаже, его targetFloor генерируется рандомно(кроме 1го)
            People.Add(_EP.EPUser(null, 1, _rand.Next(2, _selectedPorch.Building.FloorsNumber+1)));
        }
        else
        {
            //человек появляется на рандомном этаже(кроме 1го), его targetFloor = 1
            People.Add(_EP.EPUser(null, _rand.Next(2, _selectedPorch.Building.FloorsNumber+1), 1));
        }
    }
```

Рисунок 3.11 – Генерация людей на этажах

Не забываем изредка добавлять «гостей» – внеплановых пассажиров с непредсказуемыми маршрутами (рис. 3.12).

```
//Данное условие решает: генерировать ли гостя
if ((int)_period >2) //Гости генерятся с 09:00 до 00:00
{
    if (_rand.Next(1, (int)(54000 / (RANDOM_TRAVELS_PER_HOUR * 15))) == 1) //54000 - количество секунд за 15 часов
    {
        int cFloor = _rand.Next(1, _selectedPorch.Building.FloorsNumber+1), //генерим currentFloor
            tFloor = _rand.Next(1, _selectedPorch.Building.FloorsNumber+1); //генерим targetFloor
        while (cFloor == tFloor) //Если этажи сгенерились одинаковые, генерим новый targetFloor
        {
            tFloor = _rand.Next(1, _selectedPorch.Building.FloorsNumber+1);
        }
        People.Add(_EP.EPUser(null, cFloor, tFloor)); //добавляем гостя
        Console.WriteLine("GUEST!!");
    }
}
```

Рисунок 3.12 – Появление случайных гостей

Вспомогательный класс `EPUser` (рис. 3.13) отвечает за самих пассажиров: их свойства, действия и прочее.

```
public EPUser(EP_Layer EP, int _currentFloor, int _targetFloor)
{
    ButtonNumber = 0;
    _EP = EP; //Сохраняем ссылку связующий класс в приватное поле
    CurrentElevator = null; //Сохраняем ссылку на лифт
    CurrentFloor = _currentFloor; //Сохраняем текущий этаж
    TargetFloor = _targetFloor;  //Сохраняем этаж-цель
    IsInElevator = false; // Человек не в лифте
    _rand = new Random(this.GetHashCode()); // Инициализируем рандом

    Weight = _rand.Next(50,150); //Генерим вес человека

    _aggressionMaxLevel = _rand.Next(100, 200) + ((CurrentFloor > TargetFloor ? 10 : 15) * Math.Abs(CurrentFloor - TargetFloor));
    /*генерим максимальный уровень агрессии в зависимости от направления(человеку нужно подняться/спуститься)
      и в зависимости от разницы этажей */


    Console.WriteLine("Generated man: on {0} floor, want to {1} floor. Weight: {2}. MaxAggro: {3}.",
    CurrentFloor, TargetFloor, Weight, _aggressionMaxLevel); //выводим в консоль сообщение о генерации человека


    var buildings = _EP.BAL.DefaultArchitecture.GetBuildings();
    Elevator _selectedElevator;
    _selectedElevator = buildings.First<Building>().Porches.First<Porch>().Elevators.First<Elevator>();
    _EP.BAL.BroadcastLogger.Log(EventTypes.UserStepUp, _selectedElevator, (byte)CurrentFloor);
}
```

Рисунок 3.13 – Конструктор класса EPUser

Для оценки уровня терпения пассажира вводится условная величина «агрессии». Она повышается, если пассажир сталкивается с неблагоприятными ситуациями (лифт проехал мимо этажа, на котором стоит пассажир; слишком длительное ожидание и т.п.). С повышением уровня агрессии (рис. 3.14) увеличивается шанс, что человек решит не дожидаться лифта, и воспользуется лестницей.

```
#region Constants

private const int AGGRO_IF_ELEV_OVERLOADED = 40;
/// <summary>
/// На сколько увеличивать агрессивность каждые 5 секунд во время ожидания лифта.
/// </summary>
private const int AGGRO_PER_FIVE_SECONDS = 2;

/// <summary>
/// На сколько увеличивать агрессивность во время поездки в лифте, когда лифт остановился на не нужном этаже.
/// </summary>
private const int AGGRO_IF_WRONG_FLOOR = 15;

/// <summary>
/// На сколько увеличивать агрессивность когда лифт остановился на этаже, но человек в него не поместился (лифт полон).
/// </summary>
private const int AGGRO_IF_ELEV_IS_FULL = 40;

/// <summary>
/// На сколько увеличивать агрессивность когда лифт проехал мимо этажа, на котором его ждет человек, без остановки.
/// </summary>
private const int AGGRO_IF_ELEV_DIDNT_STOP = 30;
#endregion
```

Рисунок 3.14 – Повышение уровня агрессии в различных ситуациях

Модуль симуляции людей быстро «прокручивает» виртуальное время, взаимодействуя с лифтом от лица жильцов подъезда. Таким образом, мы быстро получаем данные для анализа и сбора статистики (рис 3.15).

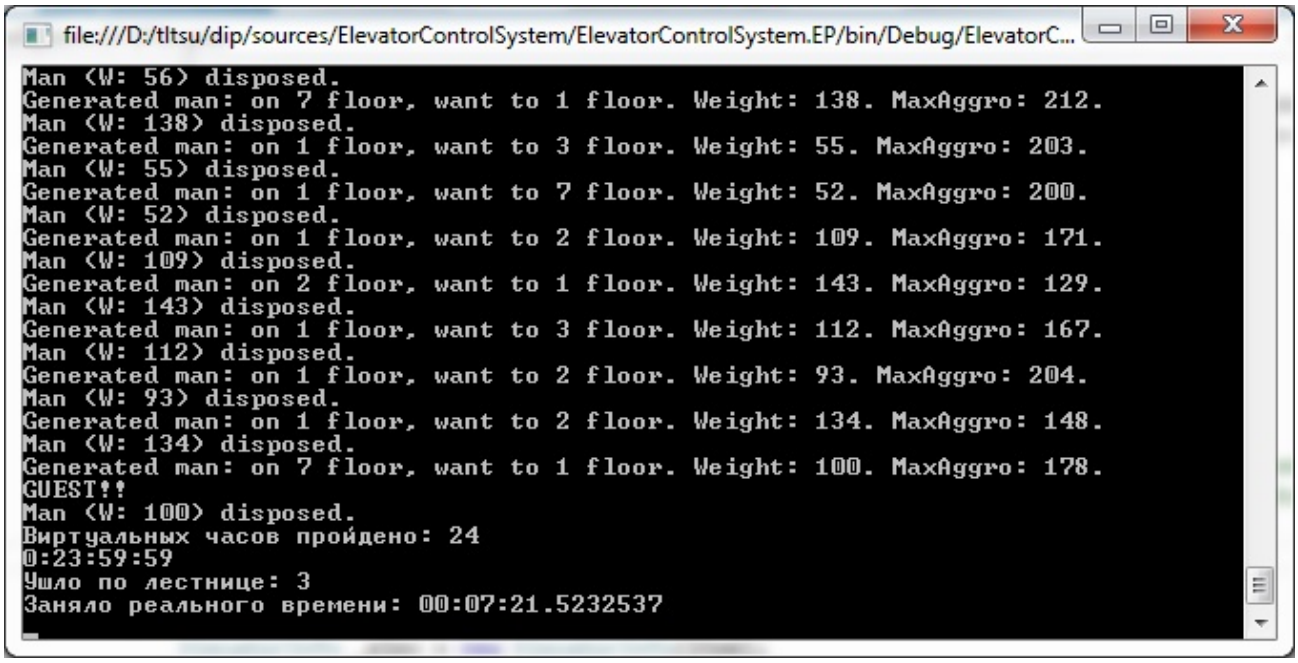

Рисунок 3.15 – Окончание работы симулятора людей спустя сутки виртуального времени (чуть больше 7 минут реального времени)

3.5 Создание функционала для сбора статистики

Теперь мы можем быстро «прокручивать» виртуальное время и записывать события лифта в базу данных. Мы даже можем просмотреть эти события в графическом интерфейсе с помощью проигрывателя, однако анализировать такой объем информации вручную – долгий процесс. Выгоднее написать программный код, который сделает эту работу за нас.

Итак, функционал для сбора статистики можно встроить в интерфейс, наряду с проигрывателем. Просто добавим кнопку, открывающую новое окно с выбором сессии и выводом статистики по ней. Но какую конкретно информацию будет выводить это окно?

Будем считать следующие показатели:

- пробег лифта в метрах;
- затраты электроэнергии в ваттах;

- время простоя в миллисекундах;
- время пути с пассажирами в кабине, в миллисекундах;
- время пути с пустой кабиной, в миллисекундах;
- кол-во перевезенных человек;
- кол-во человек, ушедших пешком по лестнице;
- число остановок на каждом этаже.

Все параметры лифта, необходимые для подсчета статистики, возьмем из таблицы конфигурации лифта в базе данных. Далее, необходимо перебрать все события, относящиеся к выбранной сессии, и правильно посчитать все показатели (рис. 3.16).

```
foreach (Event evnt in events)
{
    // Вычисление продолжительности события (в милисекундах).
    uint eventDuration = (uint)(evnt.EventTime - previousEventTime);

    switch (evnt.EventType.Id)
    {
        case EventTypes.DoorOpened:
        case EventTypes.DoorClosed:
            report.EnergyConsumption += doorsOperationsEnergyConsumptionPerMillisecond * doorsOperationsExecutingTime;
            isIdle = true;
            idleStartTime = evnt.EventTime;
            break;

        case EventTypes.DoorStartedClosing:
        case EventTypes.DoorStartedOpening:
        case EventTypes.MovingBegan:
            if (isIdle)
            {
                isIdle = false;
                idleTime = evnt.EventTime - idleStartTime;
                report.IdleTime += (uint)idleTime;
            }
            break;
```

Рисунок 3.16 – Часть программного кода, выполняющая подсчет статистических показателей на основе событий из БД

Теперь выведем эти показатели в предназначенном для этого окне (рисунки 3.17 и 3.18).

```
private void UpdateShownStats(Session session)
{
    if (!readyReports.ContainsKey(session))
    {
        KeyValuePair<Session, IReport> reportPair = Statistics.OrderReport(SessionChooser.SelectedItem as Session);
        readyReports.Add(reportPair);
    }
    IReport report = readyReports.First(x => x.Key == session).Value;

    Mileage.Content = Math.Round(report.Mileage, 2) + " м";
    EnergyConsumption.Content = Math.Round(report.EnergyConsumption, 2) + " ватт";
    IdleTime.Content = report.IdleTime + " с";
    EmplyTravelTime.Content = report.TravelBeingEmptyTime + " мс";
    PayloadTravelTime.Content = report.TravelBeingLoadedTime + " мс";
    TransportedPeopleNumber.Content = report.TransportedPeopleNumber + " чел";
    LeavedPeopleNumber.Content = report.LeavedPeopleNumber + " чел";

    StringBuilder visits = new StringBuilder();
    foreach(uint count in report.NumberOfVisits)
    {
        visits.Append(count);
        visits.Append(" ");
    }
    NumberOfVisits.Content = visits;
}
```

Рисунок 3.17 – Код для вывода статистики

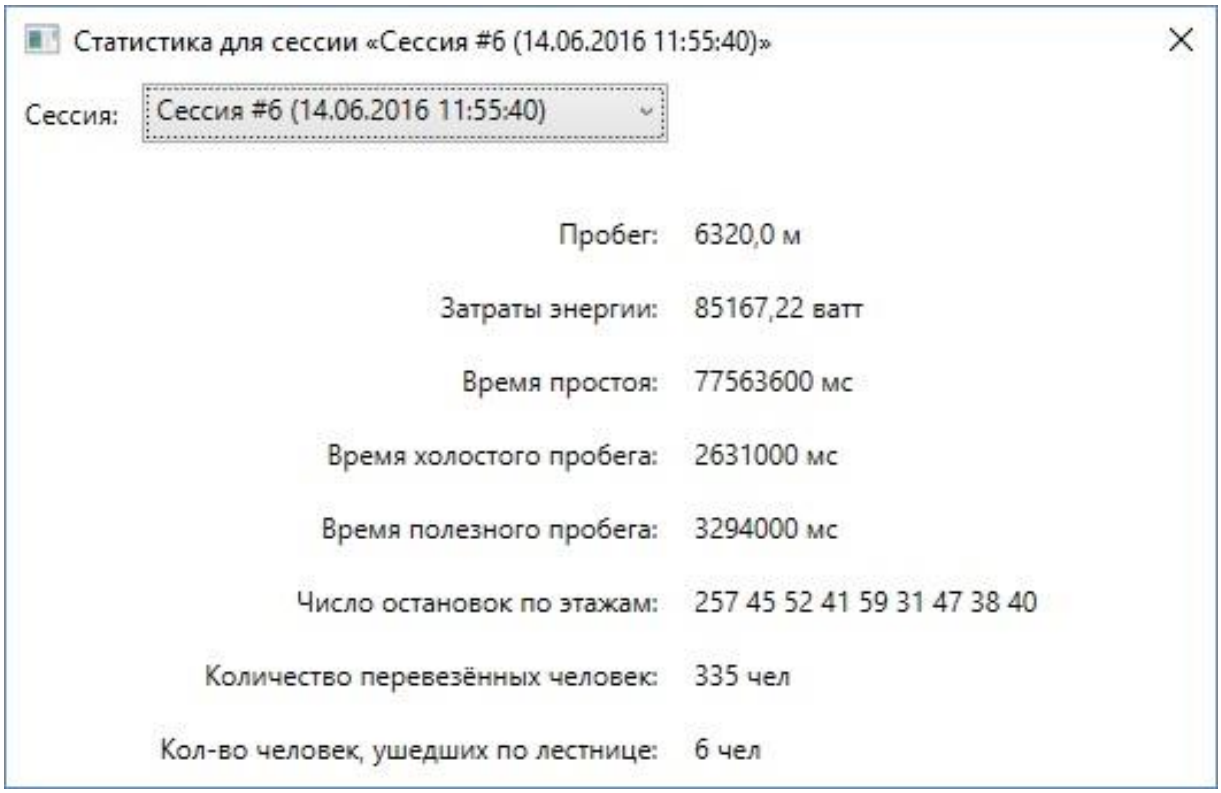

Рисунок 3.18 – Вывод статистики на экран

3.6 Сравнение нового и старого алгоритмов поведения лифта

Теперь мы можем посмотреть, насколько эффективнее работает улучшенный алгоритм поведения лифта в сравнении с его первоначальной версией.

Запустим модуль симуляции людей дважды: первый раз для лифта с простым алгоритмом поведения, второй раз для лифта с улучшенным алгоритмом. В обоих случаях настройки симулятора одинаковы: сутки виртуального времени в одном и том же жилом доме. Откроем окно статистики, выберем соответствующие сессии и сравним показатели.

На рис. 3.19 показана статистика для лифта с новым алгоритмом поведения:

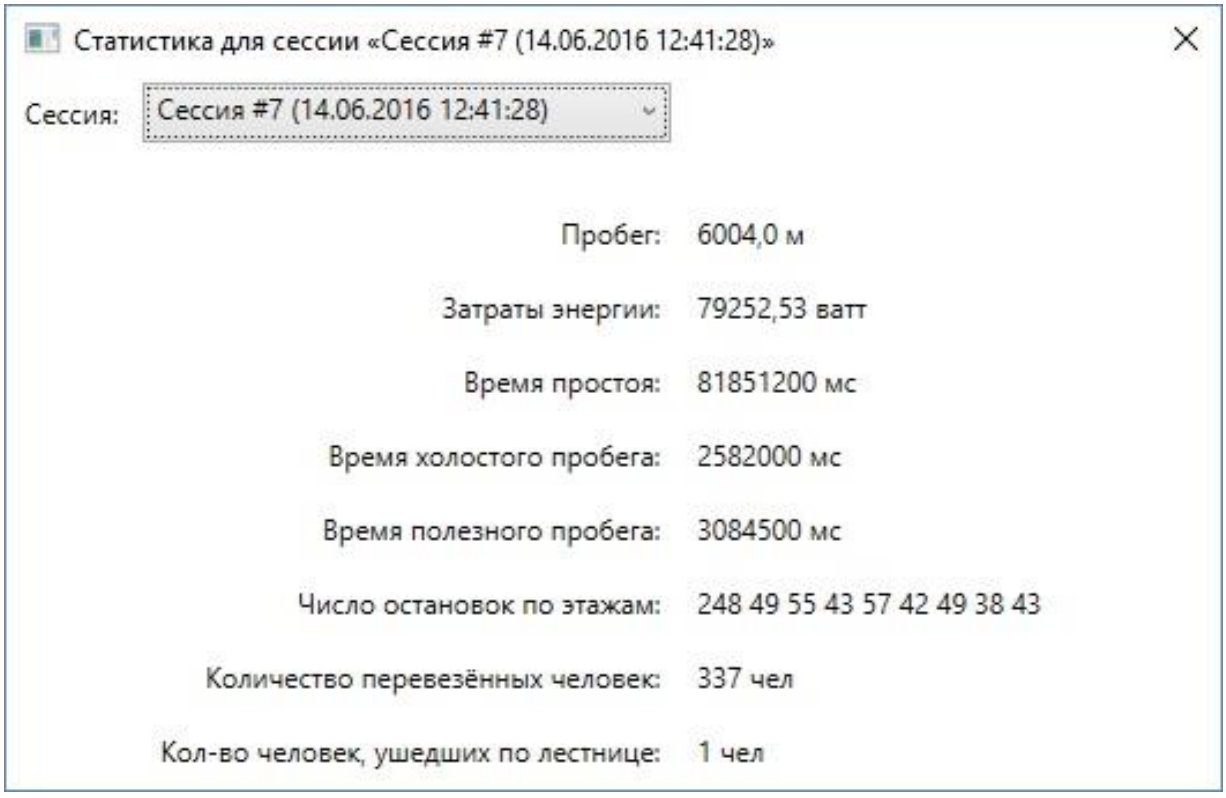

Рисунок 3.19 – Статистика лифта с новым алгоритмом

Можно заметить лишь одно – всего 1 человек предпочел лестницу лифту. Теперь посмотрим на показатели для лифта со старым алгоритмом поведения (рис. 3.20):

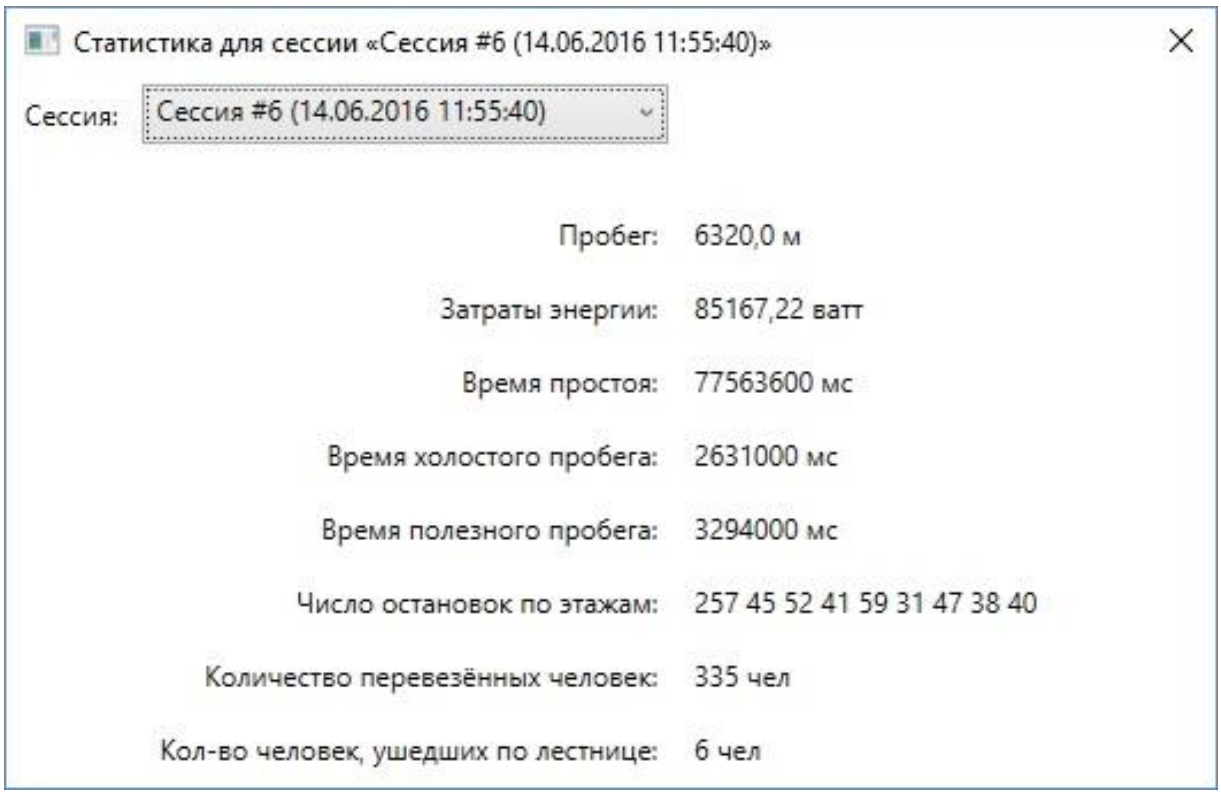

Рисунок 3.20 – Статистика лифта со старым алгоритмом

Сразу бросается в глаза пробег лифта: он несколько больше, чем у первого лифта. Время простоя: оно меньше, а вот время холостого пробега и время полезного пробега больше. Всё это говорит об одном: *лифт со старым алгоритмом поведения тратит больше времени и энергии на перевозку пассажиров*. Это подтверждается и кол-вом человек, которые предпочли пойти по лестнице – они устали ждать лифт.

Сравним показатели статистики более детально (табл. 3.2):

Таблица 3.2 – Сравнение показателей эффективности лифтов

Показатель	Новый алгоритм	Старый алгоритм	**Выигрыш**
Пробег (м)	6004,00	6320,00	**5,00%**
Затраты энергии (ватт)	79252,53	85167,22	**6,94%**
Время простоя (с)	81851,200	77563,600	**5,24%**
Время холостого пробега (с)	2582,000	2631,000	**1,86%**
Время полезного пробега (с)	3084,500	3294,000	**6,36%**
Число остановок по этажам	248 49 55 43 57 42 49 38 43	257 45 52 41 59 31 47 38 40	
Кол-во перевезенных человек	337	335	**-0,60%**
Кол-во человек, ушедших по лестнице	1	6	**83,33%**

По данным в таблице видно, что показатели лифта с новым алгоритмом в среднем на 5-6% лучше, чем у лифта со старым алгоритмом, и это учитывая, что в ходе работы симулятора людей первый лифт перевез на 2 пассажира больше.

Итак, преимущество лифта с усовершенствованным алгоритмом поведения было наглядно показано путем сравнения показателей статистики.

Заключение

В ходе работы была достигнута поставленная цель, для чего были выполнены следующие задачи:

1. Проанализировано текущее состояние исходной программы. Анализ показал, что она работает исправно и реализовывает почти весь запланированный функционал. Тем не менее, для достижения поставленной цели программу необходимо доработать.
2. Проанализирован алгоритм поведения, использующийся в программе: с текущим алгоритмом лифт способен обрабатывать только один вызов единовременно.
3. Руководствуясь сформулированной целью работы и поставленными задачами, был разработан новый функционал программы:
   - 3.1. Создана улучшенная версия алгоритма поведения: лифт теперь «запоминает» вызовы и обрабатывает их в порядке очереди, а также совершает обход этажей при движении вниз.
   - 3.2. Создан режим «проигрывателя» для воспроизведения событий лифта, взятых из базы данных.
   - 3.3. Создан модуль симуляции людей, способный быстро воспроизвести поведение людей жилого дома при взаимодействии с лифтом в течение суток виртуального времени.
   - 3.4. Создан функционал для сбора статистики, позволяющий просмотреть статистические показатели для определенной сессии использования лифта.
4. С помощью нового функционала, было произведено сравнение эффективности лифтов с новым и старым алгоритмами поведения путем сравнения статистических показателей, таких как пробег,

затраты энергии, время простоя, время холостого и полезного пробега, количество перевезенных человек, количество человек, ушедших по лестнице. Было показано преимущество лифта с новым алгоритмом.

Список используемой литературы

1. Кузин, А.В. Базы данных / С.В. Левонисова. – 3-е изд.; Гриф УМО. – М.: Академия, 2008. – 315 с.: ил.
2. Агальцов, В.П. Базы данных. Кн.1. Локальные базы данных. – 2-е изд., перераб.; Гриф УМО. – М.: ФОРУМ - ИНФРА-М, 2009. – 349 с.: ил.
3. Агальцов, В.П. Базы данных. Кн. 2. Распределенные и удаленные базы данных. – Гриф УМО. – М.: ФОРУМ - ИНФРА-М, 2009. – 270 с.: ил.
4. Малыхина, М.П. Базы данных: основы, проектирование, использование. – 2-е изд.; Гриф УМО. – СПб.: БХВ-Петербург, 2007. – 517 с.: ил.
5. Симан, М. Внедрение зависимостей в .NET. — СПб.: «Питер», 2014. — 464 с.: ил.
6. Мельников, В.П. Информационные технологии. – Гриф УМО. – М.: Академия, 2008. – 425 с.
7. Черников, Б.В. Информационные технологии управления. – Гриф УМО. – М.: ФОРУМ: ИНФРА-М, 2008. – 351 с.: ил.
8. Кнут, Д.Э. Искусство программирования: в 4 ч. – 2-е изд. - М.: «Вильямс», 2007. – 824 с.
9. Гвоздева, Т.В. Проектирование информационных систем / Б.А. Баллод. – Гриф УМО. – Ростов н/Д : Феникс, 2009. – 508 с.: ил.
10. Грекул, В. И. Проектирование информационных систем / Г.Н. Денищенко, Н.Л. Коровкина. – 2-е изд. – М.: Интернет-Ун-т Информ. Технологий: БИНОМ. Лаб. знаний, 2008. – 299 с.: ил.
11. Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий. – 2-е изд. – М.: МГИУ, 2008. – 278 с.: ил.

12. Гагарина, Л.Г. Технология разработки программного обеспечения / Е.В. Кокорева, Б.Д. Виснадул. – Гриф УМО. – М.: ФОРУМ - ИНФРА-М, 2009. – 399 с.: ил.
13. Манухин, С.Б. Устройство, техническое обслуживание и ремонт лифтов / И.К. Нелидов – М.: ИЦ «Академия», 2004. – 336 с.
14. Строй-Техника.Ру – информационная система по строительной технике [Электронный ресурс]. – Электрон. дан. – [М.], 2007-2013. – сайт «Строй-Техника.Ру»: http://stroy-technics.ru/
15. «Лифт-Пресс.Ru» - лента лифтовых новостей [Электронный ресурс]. – Электрон. дан. – [М.], 2009-2016. – сайт «Лифт-Пресс.Ru»: http://www.lift-press.ru/
16. MSDN – сеть разработчиков Microsoft [Электронный ресурс]. – Электрон. дан. – [М.], 2005-2016. – сайт MSDN: http://msdn.microsoft.com
17. naladchik2006.ru [Электронный ресурс]: Форум свободного общения о лифтах – Электрон. дан. – [М.], 2000-2007. – сайт «naladchik2006»: http://naladchik2006.ru/
18. Stiefel, M. Application Development Using C# and .NET / R. Oberg – Prentice Hall Professional, 2002.
19. Powell, G. Beginning Database Design. – John Wiley & Sons, 2006.
20. Welch, L. Dependency Injection With Unity: Microsoft Patterns & Practices. – CreateSpace Independent Publishing Platform, 2015.
21. Hejlsberg, A. The C# Programming Language (Covering C# 4.0) / M. Torgersen, S. Wiltamuth, P. Golde – Addison-Wesley Professional, 2010.
22. Liberty, J. Programming C#: Building .NET Applications with C#. – O'Reilly Media, Inc., 2005.
23. Lerman, J. Programming Entity Framework: Dbcontext / R. Miller – O'Reilly Media, Inc., 2012.
24. Bosch, J. Software Architecture: System Design, Development and Maintenance / M. Gentleman, C. Hofmeister, J. Kuusela – Springer, 2013.