

Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
федерального государственного бюджетного образовательного учреждения высшего образования
«Владимирский государственный университет
Имени Александра Григорьевича и Николая Григорьевича Столетовых»
(МИВлГУ)

Факультет _____ ИТ
Кафедра _____ ПИН

КУРСОВАЯ РАБОТА

по _____ Разработка кроссплатформенных приложений

Тема Автоматизированная информационная система Страхового агентства

(оценка)

Руководитель

Кульков Я.Ю.
(фамилия, инициалы)

(подпись) (дата)

Члены комиссии

Студент ПИН-119
(группа)

(подпись) (Ф.И.О.)

Лямина И.А.
(фамилия, инициалы)

(подпись) (Ф.И.О.)

(подпись) (дата)

Задание

Задание

В данной курсовой работе была разработана БД и АИС Страхового агентства. В ходе выполнения курсовой работы выявлены требования к программе, разработаны модели данных и диаграммы. На основе разработанных моделей создана БД в СУБД MySQL и разработано приложение на языке программирования Java в среде разработки IntelliJ IDEA. На заключительном этапе работы произведено тестирование разработанного продукта.

In this course work, a database and an automated information system of the Insurance Agency were developed. In the course of the course work, the requirements for the program were identified, data models and diagrams were developed. Based on the developed models, a database was created in the MySQL DBMS and an application was developed in the Java programming language in the IntelliJ IDEA development environment. At the final stage of the work, the developed product was tested.

Содержание

Введение	6
1. Разработка алгоритмов	8
1.1 Действующие документы	8
1.2 Обоснование выбора средств реализации	9
1.3 Функциональные возможности	10
2. Разработка алгоритмов	12
3. Руководство программиста	39
4. Руководство пользователя	43
5. Тестирование АИС	44
Заключение	60
Список используемой литературы	61
Приложение 1. Текст программы.....	62
Приложение 2. Снимки окон программы (скриншоты программы).....	63

					МИВУ 09.03.04-8.000 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Лямина И.А.			АИС Страхового агентства	Лит.	Лист
Провер.		Кульков Я.Ю.					5
Реценз.						МИ ВлГУ ПИН-119	
Н. Контр.							
Утверд.							

Введение

Сегодня, большинство пользовательских приложений рассчитаны на работу с базами данных, в которых может храниться не только текстовая информация, но и графика, музыка или любой другой тип данных. Так как потребность в использовании БД растет, появилось большое количество СУБД.

Эффективное управление предприятием в современных условиях невозможно без использования компьютерных технологий. Разработка автоматизированной информационной системы (АИС) страхового агентства важна, так как будет разработано приложение для автоматизации процесса страхования, которое поможет страховым агентам сократить время на работу с документацией.

Внедрение информационных технологий в процесс планирования и управления деятельностью страховых компаний предусматривает не только обработку больших и взаимосвязанных массивов данных, но может использоваться также для их анализа и обоснований вариантов управленческих решений.

Объемы информации, высокие требования к точности и достоверности, необходимость эффективного анализа финансового состояния клиентуры и страховой фирмы — вот основные причины, предопределяющие автоматизацию страхового бизнеса.

Функции, которые будут автоматизированы:

– если приходил страхователь, то сотруднику приходится искать вручную его бумажные договора, возможно спрашивая самого клиента. В АИС достаточно написать номер телефона или паспорт страхователя и сотрудник получит всю информацию как о самом клиенте, так и о всех заключенных договорах;

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

– чтобы продлить договор приходилось переписывать и перепроверять все данные. В АИС можно просто нажать на кнопку продлить и выбрать срок, на который нужно продлить;

– автоматически составляются отчёты. Сотруднику достаточно выбрать вид страхования и период времени, и он получит следующие данные: количество и сумма заключенных договоров, сумма страховых выплат.

Целью курсовой работы является проектирование и разработка автоматизированной информационной системы страхового агентства.

Для достижения поставленной цели были поставлены следующие задачи:

- проанализировать предметную область;
- разработать модели данных;
- реализовать базу данных;
- разработать клиентское приложение;
- протестировать программный продукт.

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

1. Анализ технического задания

1.1 Действующие документы

Для наиболее точного анализа предоставленного технического задания были найдены действующие полисы ОСАГО и КАСКО. Примеры данных документов приведены на рисунках 1 и 2.

1 — **ЭЛЕКТРОННЫЙ СТРАХОВОЙ ПОЛИС** серия XXX № 000000000 — **2**

ОБЯЗАТЕЛЬНОГО СТРАХОВАНИЯ ГРАЖДАНСКОЙ ОТВЕТСТВЕННОСТИ
ВЛАДЕЛЬЦЕВ ТРАНСПОРТНЫХ СРЕДСТВ

РОСГОВЕСТРАХ
ООО «Росгосстрах»
Т: 0530 (моб.) или 8-800-200-0-900
www.RGS.ru

Срок страхования с **17** ч. **30** мин. **01** **05** . 20 **16** г. — **3**
по 24 ч. 00 мин. **30** **04** . 20 **17** г.

4 — Страхование распространяется на страховые случаи, произошедшие в период использования транспортного средства в течение срока страхования:
с **01** **05** . 20 **16** г. по **31** **08** . 20 **16** г., с **01** **10** . 20 **16** г. по **30** **04** . 20 **17** г.,
с . 20 г. по . 20 г.

5 — 1. Страхователь (полное наименование юридического лица или фамилия, имя, отчество* гражданина)
Николаев Николай Николаевич

6 — Собственник транспортного средства (полное наименование юридического лица или фамилия, имя, отчество* гражданина)
Николаева Мария Сергеевна

2. Транспортное средство используется с прицепом: да, нет.

Марка, модель транспортного средства **Toyota Corolla** Идентификационный номер транспортного средства **JTHBA00E490144371** Государственный регистрационный знак транспортного средства **T110 EA 63**

7 — Вид документа **паспорт транспортного средства** серия **63 УТ** номер **348543**

Цель использования транспортного средства (отметить нужное): ☒ личная, учебная езда, такси, перевозка опасных и легковоспламеняющихся грузов, прокат/краткосрочная аренда, регулярные пассажирские перевозки/перевозки пассажиров по заказам, дорожные и специальные транспортные средства, экстренные и коммунальные службы, прочие.

3. Договор заключен в отношении: неограниченного количества лиц, допущенных к управлению транспортным средством, лиц, допущенных к управлению транспортным средством

8 — № п/п Лица, допущенные к управлению транспортным средством (фамилия, имя, отчество*)
1 Николаев Николай Николаевич
2 Николаев Степан Николаевич
Водительское удостоверение (серия, номер)
63 01 112113
63 02 673749

4. Страховая сумма, в пределах которой страховщик при наступлении каждого страхового случая (независимо от количества страховых случаев в течение срока страхования по договору обязательного страхования) обязуется возместить потерпевшим причиненный вред, установлена Федеральным законом от 25 апреля 2002 года №40-ФЗ «Об обязательном страховании гражданской ответственности владельцев транспортных средств» в редакции, действующей на дату заключения (изменения*) настоящего договора.

5. Страховой случай - наступление гражданской ответственности владельца транспортного средства за причинение вреда жизни, здоровью или имуществу потерпевших при использовании транспортного средства, влекущее за собой в соответствии с договором обязательного страхования обязанность страховщика осуществить страховую выплату.

6. Страховой полис действует на территории Российской Федерации.

9 — 7. Страховая премия **8 345 руб. 00 копеек** рублей.

10 — 8. Особые отметки
Данный полис был приобретен на сайте ООО «Росгосстрах» по адресу: www.RGS.ru (заказ № 1241538)

11 — Дата заключения договора **27** апреля 20 **16** г.

12 — Страхователь: **27** апреля 20 **16** г. (дата выдачи полиса)

13 — Страховщик / представитель страховщика: **Маркоров Д.Э.** (подпись) (фамилия, имя, отчество)

Рисунок 1 - пример страхового полиса ОСАГО

Изм.	Лист	№ докум.	Подпись	Дата

Чтобы разработать БД Страхового агентства выбрана СУБД MySQL. MySQL является надежной базой данных для любых целей, может продолжать расширяться по мере наполнения информацией, без заметного уменьшения быстродействия операций с записями в многопользовательском режиме. Обеспечивается максимальная безопасность. Техническое обслуживание MySQL очень простое и не требует больших знаний.

В качестве среды для разработки прикладной программы для работы с созданной в MySQL базой данных, была выбрана среда программирования IntelliJ IDEA, язык программирования Java. IntelliJ IDEA предоставляет массу возможностей для быстрой и эффективной разработки: умное автодополнение, анализ кода в реальном времени и надежные рефакторинги.

Визуальная часть приложения разработана на JavaFX. JavaFX - это платформа клиентских приложений нового поколения с открытым исходным кодом для настольных, мобильных и встраиваемых систем, построенных на Java.

Разрабатываемое программное средство должна обеспечивать получение из базы данных всей необходимой информации в полном объеме, а также возможность её редактирования и удаления. Также программа должна иметь визуальный интерфейс для работы с базой данных.

1.3 Функциональные возможности

Пользоваться системой страхового агентства будут сотрудники, которые будут иметь возможность просмотра, добавления, изменения и удаления данных.

Программа должна содержать следующие функциональные возможности:

- добавление данных о страхователях, сотрудниках, автомобилях, лицах, допущенных к управлению;
- добавление полисов;

					МИВУ 09.03.04-8.000 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

- возможность работы со страховыми случаями;
- возможность изменения информации;
- возможность удаления выбранной информации;
- возможность анализировать количество и суммы заключенных договоров по каждому из видов, а также оценивать риски, подсчитывая суммы страховых выплат по каждому виду договоров, а также составлять финансовый отчет деятельности компании за заданный период времени.

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

2. Разработка алгоритмов

Модели базы данных разрабатываются, чтобы определить логическую структуру базы данных и каким образом данные могут храниться, организовываться и обрабатываться. Разработка моделей данных является очень важным этапом в разработке АИС, на котором выделяются сущности, атрибуты сущностей и связи между ними. Необходимо разработать концептуальную, логическую и физическую модели данных.

Но прежде, чем разрабатывать модели данных нужно выявить ограничения предметной области. В данной предметной области существуют следующие ограничения:

- страхователь может оформлять несколько полисов;
- полис оформляется одним страхователем;
- полис оформляется на один автомобиль;
- на каждый автомобиль можно оформить несколько полисов;
- одновременно на автомобиль могут быть оформлены полисы ОСАГО и КАСКО. Пока не пройдет срок действия ОСАГО нельзя оформлять ещё один полис ОСАГО. Аналогично с КАСКО;
- у автомобиля должна быть хотя бы одна фотография, но есть возможность закреплять за каждым автомобилем сразу несколько фотографий;
- каждая фотография может закрепляться только за одним автомобилем;
- полис не перестаёт действовать после наступления страхового случая, поэтому за каждым полисом могут быть закреплены несколько страховых случаев;
- страховой случай закрепляется за одним конкретным полисом;
- при оформлении полиса должен указываться сотрудник, который его оформил;
- сотрудник может оформить несколько полисов;

					МИВУ 09.03.04-8.000 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

– в полисе должно указываться хотя бы одно лицо, допущенное к управлению, но есть возможность закреплять за каждым полисом сразу несколько лиц, допущенных к управлению;

– лицо, допущенное к управлению может быть закреплено за несколькими полисами;

– если страхователь оформляет полис, то он не является лицом, допущенным к управлению по умолчанию, поэтому при необходимости его тоже нужно добавлять в данный список.

2.1 Концептуальная модель данных

Концептуальная модель — это отражение предметной области, для которой разрабатывается база данных. Так, все сущности обозначаются в виде прямоугольника. Атрибуты, характеризующие объект - в виде овала, а связи между объектами - ромбами. Мощность связи обозначаются стрелками (в направлении, где мощность равна многим - двойная стрелка, а со стороны, где она равна единице - одинарная).

Концептуальная модель, разработанная для БД Страхового агентства представлена на рисунке 3. Объектами на разработанной модели являются Страхователи, Автомобили, Изображения, Полисы, Лица, допущенные к управлению, Страховые случаи, Сотрудники. Страхователь может оформить полис, который включает Лица, допущенные к управлению, Страховые случаи и Автомобили, у которых есть изображения. Полис регистрирует Сотрудник.

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

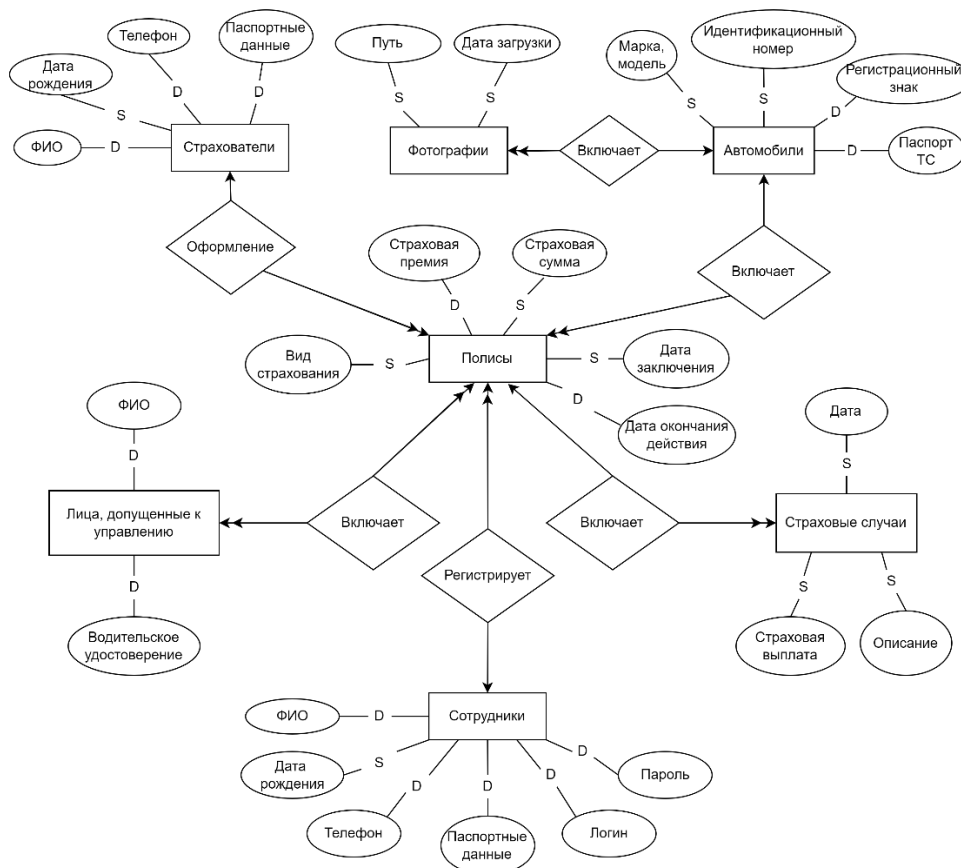


Рисунок 3 - Концептуальная модель данных

Страхователи

Dom (ФИО) = {строка символов длиной не более 64}

Dom (Дата рождения) = {строка символов длиной 10, символами являются быть цифры или «-»}

Dom (Телефон) = {строка символов длиной 15, символами являются цифры}

Dom (Паспортные данные) = {строка символов длиной 10, символами являются цифры}

Автомобили

Dom (Марка, модель) = {строка символов длиной не более 50}

Dom (Идентификационный номер) = {строка символов длиной 17, символами являются цифры и буквы}

Dom (Регистрационный знак) = {строка символов длиной не более 25, символами являются цифры и буквы}

Dom (Паспорт ТС) = {строка символов длиной 10, символами являются цифры и буквы}

Изображения

Dom (Путь) = {строка символов длиной не более 100}

Dom (Дата загрузки) = {строка символов длиной 10, символами являются быть цифры или «-»}

Полисы

Dom (Вид страхования) = {строка символов длиной 5, символами являются буквы}

Dom (Страховая сумма) = {строка символов длиной не более 10, символами являются цифры}

Dom (Страховая премия) = {строка символов длиной не более 10, символами являются цифры}

Dom (Дата заключения) = {строка символов длиной 10, символами могут быть цифры или «-»}

Dom (Дата окончания действия) = {строка символов длиной 10, символами могут быть цифры или «-»}

Лица, допущенные к управлению

Dom (ФИО) = {строка символов длиной не более 64}

Dom (Водительское удостоверение) = {строка символов длиной не более 10, символами являются цифры}

Страховые случаи

Dom (Дата) = {строка символов длиной 10, символами являются цифры или «.»}

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		15

Dom (Страховая выплата) = {строка символов длиной не более 50, символами являются цифры}

Dom (Описание) = {строка символов длиной 1000}

Сотрудники

Dom (ФИО) = {строка символов длиной не более 64}

Dom (Дата рождения) = {строка символов длиной 10, символами являются цифры или «.»}

Dom (Телефон) = {строка символов длиной 15, символами являются цифры}

Dom (Паспортные данные) = {строка символов длиной 10, символами являются цифры}

Dom (Логин) = {строка символов длиной не более 32}

Dom (Пароль) = {строка символов длиной не более 32}

2.3 Логическая модель данных

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области. Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

Логическая модель должна читаться по схеме: <Сущность 1> — <отношение / влияние> — <Сущность 2>. Чтение логической модели, представленной на рисунке 4: Страхователь оформляет полис. Полис включает страховые случаи и автомобиль. Автомобиль включает фотографии. Полис регистрирует сотрудник. Полис включает лица, допущенные к управлению (соединение через дополнительную таблицу, так как связь “многие ко многим”).

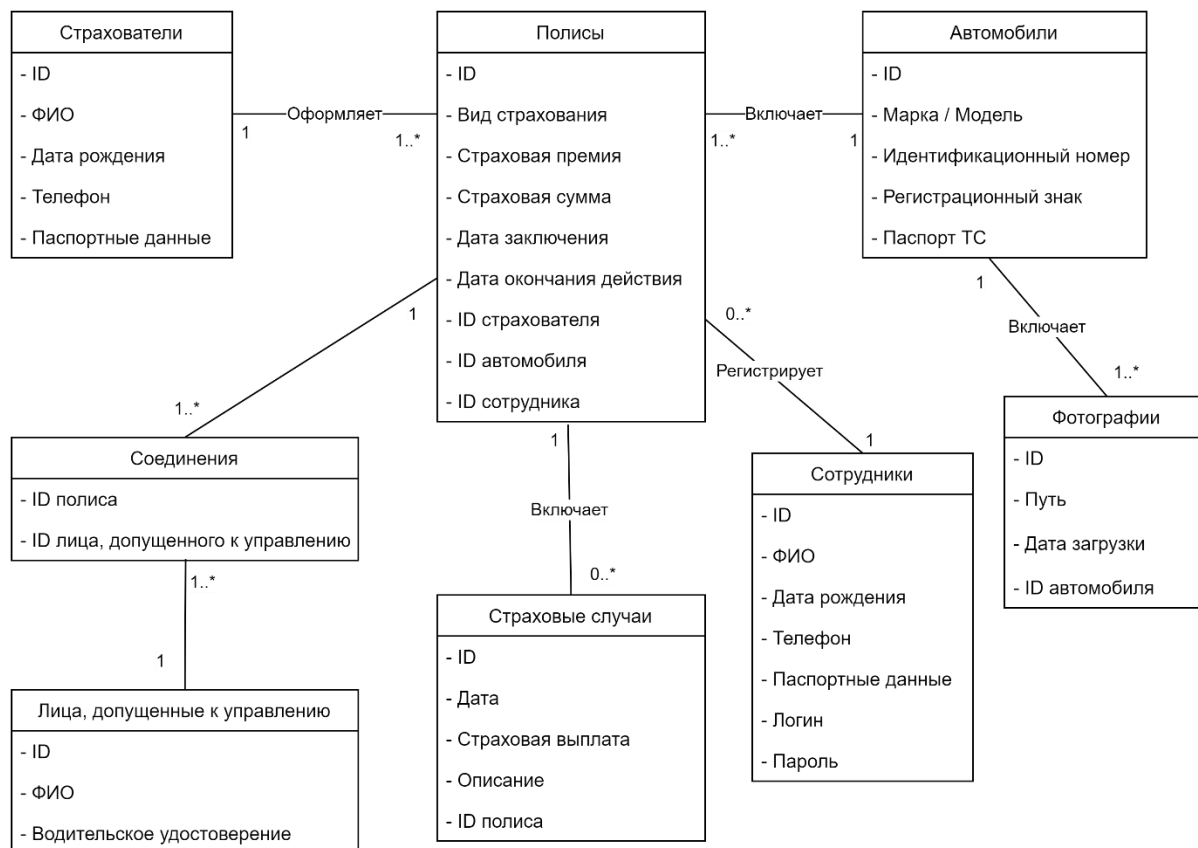


Рисунок 4 - Логическая модель данных

2.4 Соответствие сущностей логической и физической моделей

Построение физической модели БД производится на основе логической модели. В таблицах 1-8 описано соответствие сущностей логической модели и таблиц физической.

Таблица 1 - Таблица Страхователи(Policyholders)

Сущность	Таблица
ID	ID
ФИО	FullName
Дата рождения	Birthday
Телефон	Telephone
Паспортные данные	Passport

Таблица 2 - Таблица Автомобили(Cars)

Сущность	Таблица
ID	ID
Марка, модель	Model
Идентификационный номер	VIN
Регистрационный знак	RegistrationPlate
Паспорт ТС	VehiclePassport

Таблица 3 - Таблица Фотографии(Photos)

Сущность	Таблица
ID	ID
Путь	Path
Дата загрузки	UploadDate
ID автомобиля	Car_ID

Таблица 4 - Таблица Полисы(Policies)

Сущность	Таблица
ID	ID
Вид страхования	InsuranceType
Страховая премия	InsurancePremium
Страховая сумма	InsuranceAmount
Дата заключения	DateOfConclusion
Дата окончания действия	ExpirationDate
ID страхователя	Policyholder_ID
ID автомобиля	Car_ID
ID сотрудника	Employee_ID

Таблица 5 - Таблица Соединения(Connections)

Сущность	Таблица
ID полиса	Policy_ID
ID лица, допущенного к управлению	PersonAllowedToDrive_ID

Таблица 6 - Таблица Лица, допущенные к управлению
(PersonsAllowedToDrive)

Сущность	Таблица
ID	ID
ФИО	FullName
Водительское удостоверение	DrivingLicence

Таблица 7 - Таблица Страховые случаи (InsuranceEvents)

Сущность	Таблица
ID	ID
Дата	Date
Страховая выплата	InsurancePayment
Описание	Discription
ID полиса	Policy_ID

Таблица 8 - Таблица Сотрудники(Employees)

Сущность	Таблица
ID	ID
ФИО	FullName
Дата рождения	Birthday
Телефон	Telephone
Паспортные данные	Passport
Логин	Login
Пароль	Password

Подробное описание каждой сущности приведено в таблицах 9-16. Также в данных таблица приведён расчет памяти, необходимой для хранения одной записи.

Таблица 9 - Таблица Policyholders

Имя столбца	Тип	Размер(байт)
ID	INT	4
FullName	VARCHAR(64)	64
Birthday	DATE	3
Telephone	VARCHAR(15)	15
Passport	VARCHAR(10)	10
Итого		96

Таблица 10 - Таблица Connections

Имя столбца	Тип	Размер(байт)
Policy_ID	INT	4
PersonAllowedToDrive_ID	INT	4
Итого		8

Таблица 11 - Таблица Cars

Имя столбца	Тип	Размер(байт)
ID	INT	4
Model	VARCHAR(50)	50
VIN	VARCHAR(17)	17
RegistrationPlate	VARCHAR(25)	25
VehiclePassport	VARCHAR(10)	10
Итого		106

Таблица 12 - Таблица Photos

Имя столбца	Тип	Размер(байт)
ID	INT	4
Path	VARCHAR(20)	20
UploadDate	DATE	3
Car_ID	INT	4
Итого		31

Таблица 13 - Таблица Policies

Имя столбца	Тип	Размер(байт)
ID	INT	4
InsuranceType	VARCHAR(5)	5
InsurancePremium	INT	4
InsuranceAmount	INT	4
DateOfConclusion	DATE	3
ExpirationDate	DATE	3
Policyholder_ID	INT	4
Car_ID	INT	4
Employee_ID	INT	4
Итого		35

Таблица 14 - Таблица PersonsAllowedToDrive

Имя столбца	Тип	Размер(байт)
ID	INT	4
FullName	VARCHAR(64)	64
DrivingLicence	VARCHAR(10)	10
Итого		78

Таблица 15 - Таблица InsuranceEvents

Имя столбца	Тип	Размер(байт)
Имя столбца	Тип	Размер(байт)
ID	INT	4
Date	DATE	3
InsurancePayment	INT	4
Description	VARCHAR(1000)	1000
Policy_ID	INT	4
Итого		1015

Таблица 16 - Таблица Employees

Имя столбца	Тип	Размер(байт)
ID	INT	4
FullName	VARCHAR(64)	64
Birthday	DATE	3
Telephone	VARCHAR(15)	15
Passport	VARCHAR(10)	10
Login	VARCHAR(32)	32
Password	VARCHAR(32)	32
Итого		160

Определим объем внешней памяти, необходимой для размещения данных за год использования. Для того чтобы оценить объем, занимаемый таблицами базы данных, необходимо оценить объем каждой таблицы. Примерный расчет необходимо объема памяти приведен в таблице 17.

Таким образом, при максимальном заполнении БД объем таблиц составит: $V_{\text{данных}} = 462.598 \text{ Кбайт/год}$.

Таблица 17 - Расчет объема ПЗУ для хранения данных

Таблица	Размер записи, байт	Максимальное (оценочное) количество записей	Всего, Кбайт
Policyholders	96	700	65,625
Cars	106	700	72,461
Photos	31	2000	60.547
Connections	8	1700	13,281
Policies	35	1000	34,180
PersonsAllowedToDrive	78	1500	114,258
InsuranceEvents	1015	100	99.121
Employees	160	20	3.125
Итого			462.598

2.5 Физическая модель данных

Физическая модель данных, зависит от конкретной СУБД, фактически являясь отображением системного каталога. Физическая модель БД определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне.

Физическая модель, разработанная для БД Страхового агентства представлена на рисунке 5.

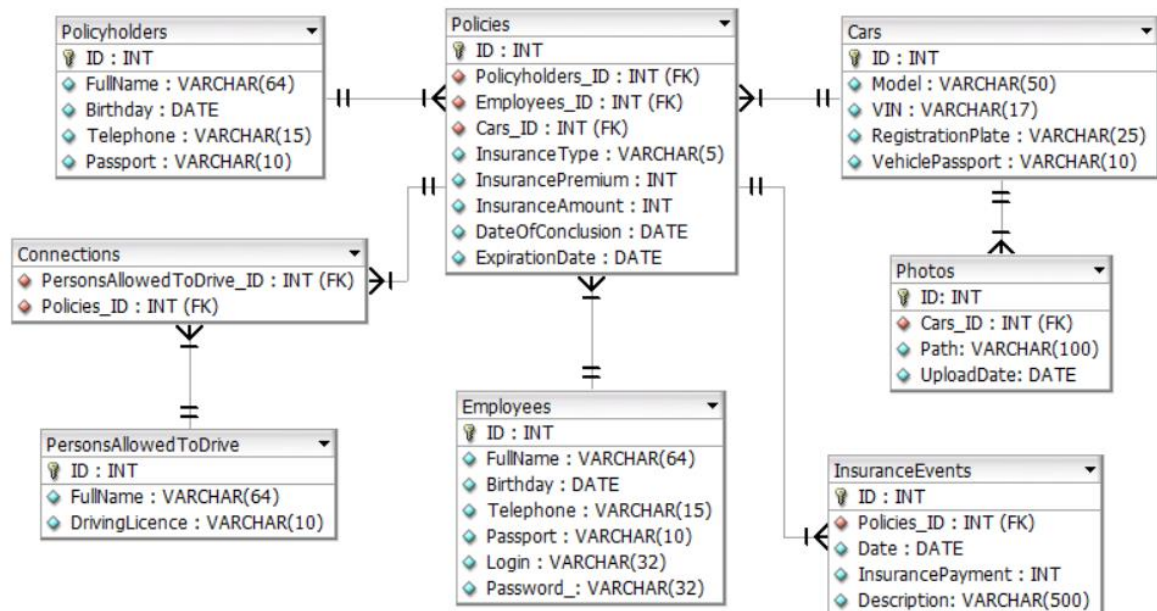


Рисунок 5 - Физическая модель данных

2.6. Диаграмма вариантов использования

Создав диаграмму вариантов использования, которая представлена на рисунке 6, были сформулированы общие требования к функциональному поведению проектируемой системы и определены общие границы и контекст моделируемой предметной области.

Актером в диаграмме является Пользователь. В самой диаграмме отображен план работы с ИС Страхового агентства. Пользователь – внешняя часть программы. Основной частью программы является: авторизация, работа с полисом, работа с автомобилем, работа с лицом, допущенным к управлению, работа со страхователем, работа с сотрудником, работа с отчётами.

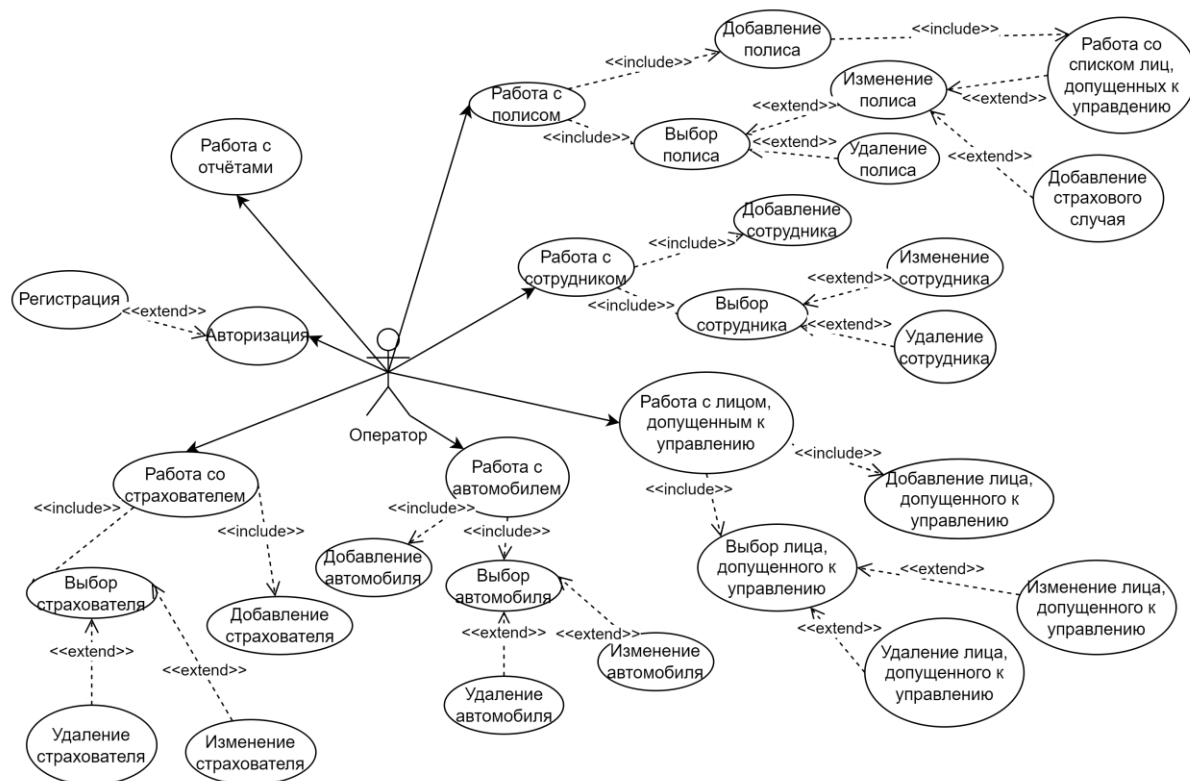


Рисунок 6 - Диаграмма вариантов использования

2.7 Добавление

Рассмотрим метод добавления Страхователя. Примерно такой же синтаксис имеют методы добавления Сотрудника, Автомобиля, Страхового случая и Лица, допущенного к управлению.

Сначала происходит считывание данных с формы и проверка введённых данных.

Листинг метода:

```
/**
 * Считывание данных с формы и проверка правильности введённых данных
 * @return Сформированный страхователь
 */
private Policyholder readDate() throws Exception {
    String fullName = tfFullName.getText().trim();
    if (fullName.isEmpty()) {
        throw new Exception("Заполните поле ФИО");
    }
}
```

```

        LocalDate birthday = dpBirthday.getValue();

        String telephone = tfTelephone.getText().trim();
        if (telephone.isEmpty()) {
            throw new Exception("Заполните поле Номер телефона");
        }
        if (telephone.length() > 15) {
            throw new Exception("Номер телефона не может быть больше 15 символов");
        }
        for (var i = 0; i < telephone.length(); i++) {
            if (!Character.isDigit(telephone.charAt(i))) {
                throw new Exception("Номер телефона должен содержать только цифры");
            }
        }

        String passport = tfPassport.getText().trim();
        if (passport.isEmpty()) {
            throw new Exception("Заполните поле Паспорт");
        }
        if (passport.length() != 10) {
            throw new Exception("Паспорт должен содержать 10 цифр");
        }
        for (var i = 0; i < passport.length(); i++) {
            if (!Character.isDigit(passport.charAt(i))) {
                throw new Exception("Паспорт должен содержать только цифры");
            }
        }

        return new Policyholder(fullName, birthday, telephone, passport);
    }

```

Затем происходит вызов метода добавления страхователя в БД и при успешном добавлении выводится сообщение об успешном добавление, а иначе сообщение об ошибке.

Листинг метода:

```

/**
 * Вызов метода добавления страхователя в БД по нажатию на кнопку "Добавить"
 */
public void onAdd(ActionEvent actionEvent) {
    try
    {
        Policyholder policyholder = readDate();
        DBPolicyholder.addPolicyholder(policyholder);

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Information");
        alert.setHeaderText(null);
        alert.setContentText("Страхователь успешно добавлен");
        alert.showAndWait();

        if(dialogStage != null ){
            policyholderForPolicy =
            DBPolicyholder.searchPolicyholderTelephoneOrPassport(policyholder.getPassport());

```

					МИВУ 09.03.04-8.000 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

```

        dialogStage.close();
    }
    else clear();
} catch (Exception exc) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText(null);
    alert.setContentText(exc.getMessage());
    alert.showAndWait();
}
}
}

```

Метод добавления страхователя в БД. В данном методе сначала отправляются два запроса, с помощью которых происходит проверка на наличие в БД, уже существующих Телефона и Паспорта. При наличии таковых выводится ошибка.

Если таковых нет, то происходит добавление Страхователя со значениями, которые ввёл/выбрал пользователь.

Листинг метода:

```

/**
 * Добавление страхователя в БД
 * @param policyholder Страхователь
 */
public static void addPolicyholder(@NotNull Policyholder policyholder) throws
Exception {
    if (policyholder == null) throw new Exception("Страхователь не выбран");

    String query1 = String.format("SELECT id FROM policyholders WHERE telephone =
's'", policyholder.getTelephone());
    String query2 = String.format("SELECT id FROM policyholders WHERE passport =
's'", policyholder.getPassport());

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    String query = String.format("INSERT INTO policyholders (fullName, birthday,
telephone, passport) VALUES ('%s', '%s', '%s', '%s')",
        policyholder.getFullName(),
        policyholder.getBirthday().format(formatter),
        policyholder.getTelephone(),
        policyholder.getPassport());

    boolean flag = false;

    try (Connection connection = DriverManager.getConnection(Database.DB_URL,
Database.LOGIN, Database.PASSWORD)) {
        Statement statement = connection.createStatement();

        ResultSet resultSet = statement.executeQuery(query1);
        int countRow = 0;
        while (resultSet.next()) countRow++;
    }
}

```

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		27

```

    if (countRow != 0) {
        flag = true;
        throw new Exception("Данный телефон уже используется");
    }

    resultSet = statement.executeQuery(query2);
    countRow = 0;
    while (resultSet.next()) countRow++;
    if (countRow != 0) {
        flag = true;
        throw new Exception("Данный паспорт уже используется");
    }

    statement.executeUpdate(query);
} catch (Exception exp) {
    if (flag) throw exp;
    else throw new Exception("Ошибка в работе БД");
}
}

```

2.8 Добавление с транзакцией

Рассмотрим метод добавления Полиса.

В целом метод похож на обычное Добавление, но так как при добавление полиса необходимо, чтобы добавилась как информация о полисе, так и лица, допущенные к управлению (а друг без друга они не должны добавляться), то используется транзакция.

То есть сначала мы начинаем транзакцию, затем добавляем полис и получаем последний добавленный ID (то есть ID полиса, который добавляем), и добавляем лица, допущенные к управлению, которые в качестве внешнего ключа используют ID добавленного автомобиля, после чего завершаем транзакцию, тем самым фиксируя изменения.

Листинг метода:

```

/**
 * Получение всего списка полисов
 * @return Список полисов
 */
public static ArrayList<Policy> allPolicies() throws Exception {
    var resultList = new ArrayList<Policy>();

    String query = "SELECT * FROM policies ORDER BY dateOfConclusion DESC";

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

```

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		28

```

    try (Connection connection = DriverManager.getConnection(Database.DB_URL,
Database.LOGIN, Database.PASSWORD)) {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query);

        while (resultSet.next()){
            int id = resultSet.getInt("id");
            String insuranceType = resultSet.getString("insuranceType");
            int insurancePremium = resultSet.getInt("insurancePremium");
            int insuranceAmount = resultSet.getInt("insuranceAmount");

            String dateOfConclusionTemp = resultSet.getString("dateOfConclusion");
            LocalDate dateOfConclusion = LocalDate.parse(dateOfConclusionTemp,
formatter);

            String expirationDateTemp = resultSet.getString("expirationDate");
            LocalDate expirationDate = LocalDate.parse(expirationDateTemp,
formatter);

            int policyholderId = resultSet.getInt("policyholderId");
            int carId = resultSet.getInt("carId");
            int employeeId = resultSet.getInt("employeeId");

            var policy = new Policy(id, insuranceType, insurancePremium,
insuranceAmount, dateOfConclusion, expirationDate, policyholderId, carId,
employeeId);
            policy.searchName();
            resultList.add(policy);
        }

        return resultList;
    } catch (Exception exp) {
        throw new Exception("Ошибка в работе БД");
    }
}

```

2.9 Изменение

Рассмотрим метод изменения Страхователя. Примерно такой же синтаксис имеют методы изменения Сотрудника, Автомобиля и Лица, допущенного к управлению.

Сначала происходит считывание данных с формы и проверка введённых данных, как и при добавлении.

Затем происходит вызов метода изменения страхователя в БД и при успешном изменении выводится сообщение об успешном добавление, а иначе сообщение об ошибке.

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		29

Листинг метода:

```
/**
 * Вызов метода изменения страхователя в БД по нажатию на кнопку "Изменить"
 */
public void onChange(ActionEvent actionEvent) {
    try {
        if (policyholder == null) throw new Exception("Страхователь не выбран");

        readDate();
        DBPolicyholder.changePolicyholder(policyholder);

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Information");
        alert.setHeaderText(null);
        alert.setContentText("Страхователь успешно изменён");
        alert.showAndWait();

        if(dialogStage != null ) dialogStage.close();
        else clear();
    } catch (Exception exp) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(null);
        alert.setContentText(exp.getMessage());
        alert.showAndWait();
    }
}
```

Метод изменения страхователя в БД. В данном методе сначала отправляются два запроса, с помощью которых происходит проверка на наличие в БД, уже существующих Телефона и Паспорта. При наличии таковых выводится ошибка.

Если таковых нет, то происходит изменение Страхователя со значениями, которые ввёл/выбрал пользователь. При этом невозможно изменить дату рождения Страхователя. Аналогично у Сотрудника невозможно изменить дату рождения, у автомобиля VIN номер.

Листинг метода:

```
/**
 * Изменение страхователя в БД
 * @param policyholder Страхователь
 */
public static void changePolicyholder(@NotNull Policyholder policyholder) throws
Exception {
    if (policyholder == null) throw new Exception("Страхователь не выбран");
}
```

					МИВУ 09.03.04-8.000 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

```

        String query1 = String.format("SELECT id FROM policyholders WHERE telephone =
        '%s' AND id <> %d",
                                     policyholder.getTelephone(),
                                     policyholder.getId());
        String query2 = String.format("SELECT id FROM policyholders WHERE passport = '%s'
        AND id <> %d",
                                     policyholder.getPassport(),
                                     policyholder.getId());

        String query = String.format("UPDATE policyholders SET fullName = '%s', telephone
        = '%s', passport = '%s' WHERE id = %d",
                                     policyholder.getFullName(),
                                     policyholder.getTelephone(),
                                     policyholder.getPassport(),
                                     policyholder.getId());

        boolean flag = false;

        try (Connection connection = DriverManager.getConnection(Database.DB_URL,
        Database.LOGIN, Database.PASSWORD)) {
            Statement statement = connection.createStatement();

            ResultSet resultSet = statement.executeQuery(query1);
            int countRow = 0;
            while (resultSet.next()) countRow++;
            if (countRow != 0) {
                flag = true;
                throw new Exception("Данный телефон уже используется");
            }

            resultSet = statement.executeQuery(query2);
            countRow = 0;
            while (resultSet.next()) countRow++;
            if (countRow != 0) {
                flag = true;
                throw new Exception("Данный паспорт уже используется");
            }

            statement.executeUpdate(query);
        } catch (Exception exp) {
            if (flag) throw exp;
            else throw new Exception("Ошибка в работе БД");
        }
    }
}

```

2.10 Удаление

Рассмотрим метод удаления Страхователя. Примерно такой же синтаксис имеют методы удаления Сотрудника, Автомобиля и Лица, допущенного к управлению.

В данном методе сначала отправляются запроса, с помощью которого происходит проверка оформлял ли Страхователь полис. Если да, то его невозможно удалить и выдаётся ошибка.

Аналогично с Сотрудником. У автомобиля проверяется оформление на него полиса, а у Лица, допущенного к управлению - наличие привязки к какому-либо полису.

Если таких связей нет, то можно произвести удаление. Удаление происходит по выбранному Id.

Листинг метода:

```
/**
 * Удаление страхователя из БД
 * @param id Id страхователя
 */
public static void deletePolicyholder(int id) throws Exception {
    String query1 = String.format("SELECT id FROM policies WHERE policyholderId = %d", id);

    String query = String.format("DELETE FROM policyholders WHERE id = %d", id);

    boolean flag = false;

    try (Connection connection = DriverManager.getConnection(Database.DB_URL,
        Database.LOGIN, Database.PASSWORD)) {
        Statement statement = connection.createStatement();

        ResultSet resultSet = statement.executeQuery(query1);
        int countRow = 0;
        while (resultSet.next()) countRow++;
        if (countRow != 0) {
            flag = true;
            throw new Exception("Вы не можете удалить данного страхователя, так как на него оформлен полис");
        }

        statement.executeUpdate(query);
    } catch (Exception exp) {
        if (flag) throw exp;
        else throw new Exception("Ошибка в работе БД");
    }
}
```

2.11 Поиск

Рассмотри поиск страхователя. При вводе в поле номера телефона или страхователя и нажатии на кнопку “Поиск” вызывается метод поиска в БД. При удачном поиске заполняются поля информацией о найденном страхователе или выводится ошибка, что такого страхователя нет.

Листинг метода:

```
/**
 * Поиск страхователя в БД по номеру телефона или паспорту при нажатии на кнопку
 * "Поиск"
 */
public void onSearch(ActionEvent actionEvent) {
    try {
        String search = tfSearch.getText();
        policyholder = DBPolicyholder.searchPolicyholderTelephoneOrPassport(search);
        fillInfo();
        tfSearch.clear();
    } catch (Exception exc) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(null);
        alert.setContentText(exc.getMessage());
        alert.showAndWait();
    }
}

/**
 * Заполняет поля информацией о выбранном страхователе
 */
private void fillInfo(){
    tfFullName.setText(policyholder.getFullName());

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
    tfBirthDay.setText(policyholder.getBirthDay().format(formatter));

    tfTelephone.setText(policyholder.getTelephone());
    tfPassport.setText(policyholder.getPassport());
}
```

Рассмотрим метод поиска Страхователя. Данный метод отправляет запрос к БД, который осуществляет выборку всех полей, той строки в которой совпадает номер телефона или паспорта с выбранным для поиска. Если такого Страхователя не существует, то выводится ошибка. Иначе возвращается информация о найденном Страхователе.

					МИВУ 09.03.04-8.000 ПЗ	Лист
						33
Изм.	Лист	№ докум.	Подпись	Дата		

Листинг метода:

```

/**
 * Поиск страхователя по номеру телефона или паспорту
 * @param telephoneOrPassport Номер телефона или паспорт
 * @return Найденный страхователь
 */
public static Policyholder searchPolicyholderTelephoneOrPassport(@NotNull String
telephoneOrPassport) throws Exception{
    if(telephoneOrPassport == null || telephoneOrPassport.isEmpty()) throw new
Exception("Телефон или паспорт страхователя не выбран");

    String query = String.format("SELECT * FROM policyholders WHERE telephone = '%s'
OR passport = '%s'", telephoneOrPassport, telephoneOrPassport);
    return searchPolicyholder(query);
}

/**
 * Поиск страхователя по определённому запросу
 * @param query Запрос
 * @return Найденный страхователь
 */
private static Policyholder searchPolicyholder(@NotNull String query) throws
Exception{
    if (query == null || query.isEmpty()) throw new Exception("Запрос не выбран");

    boolean flag = false;
    try (Connection connection = DriverManager.getConnection(Database.DB_URL,
Database.LOGIN, Database.PASSWORD)) {
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(query);

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        while (resultSet.next()) {

            int id = resultSet.getInt("id");
            String fullName = resultSet.getString("fullName");

            String birthdayTemp = resultSet.getString("birthday");
            LocalDate birthday = LocalDate.parse(birthdayTemp, formatter);

            String telephone = resultSet.getString("telephone");
            String passport = resultSet.getString("passport");

            return new Policyholder(id, fullName, birthday, telephone, passport);
        }

        flag = true;
        throw new Exception("Данный страхователь не существует");
    }catch (Exception exp) {
        if (flag) throw exp;
        else throw new Exception("Ошибка в работе БД");
    }
}

```

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		34

2.12 Отчёты

Чтобы сформировать финансовый отчёт нужно, чтобы пользователь выбрал вид страхования: ОСАГО, КАСКО или ОСАГО и КАСКО, а также дату начала и дату окончания - период за который будет сформирован отчёт.

Листинг метода:

```
/**
 * Формирует финансовый отчёт при нажатии на кнопку "Сформировать"
 */
public void onCreateReport(ActionEvent actionEvent){
    try {
        String insuranceType = cbInsuranceType.getValue();
        if (insuranceType == null)
        {
            throw new Exception("Заполните поле Вид страхования");
        }

        LocalDate dateStart = dpDateStart.getValue();
        LocalDate dateEnd = dpDateEnd.getValue();

        if (dateStart.isAfter(dateEnd))
        {
            throw new Exception("Дата начала не может быть больше даты окончания");
        }

        ArrayList<Integer> list = Database.createReport(insuranceType, dateStart,
dateEnd);

        tfCountContracts.setText(list.get(0).toString());
        tfSumContracts.setText(list.get(1).toString());
        tfSumInsuranceEvents.setText(list.get(2).toString());
    }
    catch (Exception exp) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(null);
        alert.setContentText(exp.getMessage());
        alert.showAndWait();
    }
}
```

Данный метод отправляет запрос к БД, который подсчитывает сумму страховых выплат, количество и сумму заключенных договоров за определённый период времени и по определённому виду страхования или по обоим сразу.

					МИВУ 09.03.04-8.000 ПЗ	Лист
						35
Изм.	Лист	№ докум.	Подпись	Дата		

Листинг метода:

```

/**
 * Формирует финансовый отчёт деятельности компании за заданный период времени
 * @param insuranceType Вид страхования
 * @param dateStart Дата начала
 * @param dateEnd Дата окончания
 * @return Сформированный отчёт
 */
public static ArrayList<Integer> createReport(String insuranceType, LocalDate
dateStart, LocalDate dateEnd) throws Exception{
    var resultList =new ArrayList<Integer>();

    String query;
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    if (insuranceType.equals("ОСАГО и КАСКО"))
    {
        query = String.format("SELECT " +
            "(SELECT SUM(insurancePayment) FROM insuranceEvents " +
            "WHERE date >= '%s' AND date <= '%s'), " +

            "COUNT(id), SUM(insurancePremium) FROM policies " +
            "WHERE dateOfConclusion >= '%s' AND DateOfConclusion <=
            '%s'",

            dateStart.format(formatter),
            dateEnd.format(formatter),
            dateStart.format(formatter),
            dateEnd.format(formatter));
    }
    else
    {
        query = String.format("SELECT " +
            "(SELECT SUM(insurancePayment) " +
            "FROM policies as p LEFT JOIN insuranceEvents as ie ON
            p.id = ie.policyId " +
            "WHERE p.insuranceType = '%s' AND date >= '%s' AND Date
            <= '%s'), " +

            "COUNT(id), SUM(insurancePremium) FROM policies " +
            "WHERE insuranceType = '%s' AND dateOfConclusion >=
            '%s' AND dateOfConclusion <= '%s'",

            insuranceType, dateStart.format(formatter),
            dateEnd.format(formatter),
            insuranceType, dateStart.format(formatter),
            dateEnd.format(formatter));
    }

    try (Connection connection = DriverManager.getConnection(Database.DB_URL,
Database.LOGIN, Database.PASSWORD)) {
        Statement statement = connection.createStatement();

        ResultSet resultSet = statement.executeQuery(query);
        resultSet.next();
        resultList.add(resultSet.getInt(2));
        resultList.add(resultSet.getInt(3));
        resultList.add(resultSet.getInt(1));
    }
}

```

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		36

```

        return resultList;
    } catch (Exception exp) {
        throw new Exception("Ошибка в работе БД");
    }
}

```

2.13 Хранение изображений

В базе данных хранятся только пути к изображениям, поэтому при добавлении фотографии автомобиля они хранятся по специальному пути.

Листинг:

```

final static String pathDirectoryForAllPhotos =
"src/main/resources/com/insuranceagency/photos/";

/**
 * Загрузка фотографии в папку
 * @param carId Название папки (Id автомобиля)
 * @param file Изображение для загрузки
 */
public static void loadPhotoInDirectory(int carId, File file){
    Path path = Paths.get(pathDirectoryForAllPhotos + "/" + (carId) + "/" +
file.getName());
    try {
        Files.copy(file.toPath(), path, StandardCopyOption.REPLACE_EXISTING);
    } catch (Exception exp) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(null);
        alert.setContentText(exp.getMessage());
        alert.showAndWait();
    }
}

/**
 * Добавление фотографий автомобиля в БД
 * @param listPhotos Список фотографий для добавления
 */
public static void addPhotos(ArrayList<Photo> listPhotos) throws Exception{
    if (listPhotos == null || listPhotos.size() == 0) throw new Exception("Список
фотографий пуст");

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    String query = "INSERT INTO photos(path, uploadDate, carId) VALUES ";
    for (var i = 0; i < listPhotos.size() - 1; i++)
    {
        query += String.format("('%s', '%s', %d), ",
            listPhotos.get(i).getPath(),
            listPhotos.get(i).getUploadDate().format(formatter),
            listPhotos.get(i).getCarId());
    }
    int index = listPhotos.size() - 1;

```

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		37

```

        query += String.format("('%s', '%s', %d); ",
                                listPhotos.get(index).getPath(),
                                listPhotos.get(index).getUploadDate().format(formatter),
                                listPhotos.get(index).getCarId());

        try (Connection connection = DriverManager.getConnection(Database.DB_URL,
                                                                    Database.LOGIN, Database.PASSWORD)) {
            Statement statement = connection.createStatement();
            statement.executeUpdate(query);
        } catch (Exception exp) {
            throw new Exception("Ошибка в работе БД");
        }
    }
}

```

3. Руководство программиста

Полную документацию по классам можно найти по ссылке, указанной в приложении 1. Там описаны все классы и методы.

3.1 Начало работы

При запуске приложения сначала появляется форма авторизации с полями для ввода логина и пароля, за работу которой отвечает `AuthorizationController`. После успешной авторизации появляется главное окно приложения с меню, за работу которого отвечает `MainController`. В `MainController` обрабатываются все нажатия на кнопки меню и загрузка определённых страниц.

3.2 Модели

В программе реализован ряд моделей, отражающих соответствующие таблицы в базе данных – классы `Car` (автомобиль), `Employee` (сотрудник), `InsuranceEvent` (страховой случай), `PersonAllowedToDrive` (лицо, допущенное к управлению), `Photo` (фотография), `Policy` (полис), `Policyholder` (страхователь).

3.3 База данных

За взаимодействия с базой данных отвечают классы: `Database` (класс с общей информацией, а также с методами авторизации и формированием отчёта) и классы взаимодействия с определёнными таблицами: `DBCar`, `DBEmployee`, `DBInsuranceEvent`, `DBPersonAllowedToDrive`, `DBPhoto`, `DBPolicy`, `DBPolicyholder`.

3.4 Автомобили, Страхователи, Сотрудники и Лица, допущенные к управлению

Для добавления автомобиля используется представление `addCar.fxml`, которое обрабатывается контроллером `AddCarController`. На данной `fxml` основные поля для ввода это: Модель, VIN номер, Регистрационный знак, Паспорт ТС. Также на данной форме есть кнопка для загрузки фотографий и загруженные фотографии отображаются в мини-слайдере.

Аналогично представление для изменения `changeCar.fxml`, которое обрабатывается контроллером `ChangeCarController`. Исключение в этом представлении составляет лишь то, что нельзя изменять поля Модель, VIN номер.

Для добавления страхователя используется представление `addPolicyholder.fxml`, которое обрабатывается контроллером `AddPolicyholderController`. На данной `fxml` основные поля для ввода это: ФИО, Дата рождения, Номер телефона и Паспорт.

Аналогично представление для изменения `changePolicyholder.fxml`, которое обрабатывается контроллером `ChangePolicyholderController`. Исключение в этом представлении составляет лишь то, что нельзя изменять поля дату рождения.

Для добавления сотрудника используется представление `addEmployee.fxml`, которое обрабатывается контроллером `AddEmployeeController`. На данной `fxml` основные поля для ввода это: ФИО, Дата рождения, Номер телефона, Паспорт, Логин и Пароль.

Аналогично представление для изменения `changeEmployee.fxml`, которое обрабатывается контроллером `ChangeEmployeeController`. Исключение в этом представлении составляет лишь то, что нельзя изменять

поля дату рождения. Также если не вводить пароль, то останется старый, а при вводе значения в поле пароль обновляется.

Для добавления лица, допущенного к управлению, используется представление `addPersonAllowedToDrive.fxml`, которое обрабатывается контроллером `AddPersonAllowedToDriveController`. На данной `fxml` основные поля для ввода это: ФИО и Водительское удостоверение.

Аналогично представление для изменения `changePersonAllowedToDrive.fxml`, которое обрабатывается контроллером `ChangePersonAllowedToDriveController`.

Вывод списков. Колонки таблиц в представлениях `allCars.fxml`, `allEmployees.fxml`, `allPersonsAllowedToDrive.fxml`, `allPolicyholders.fxml`, которые обрабатываются контроллерами `AllCarsController`, `AllEmployeesController`, `AllPersonsAllowedToDriveController`, `AllPolicyholdersController`, разработаны с учётом всех полей представленных моделей.

3.5 Полис

`AddPolicyController` - контроллер для представления `addPolicy.fxml`. На данной `fxml` основные поля для ввода это: Вид страхования, Страховая премия, Страховая сумма, Дата заключения, и Срок действия. Также можно добавить или выбрать страхователя и автомобиль, сотрудник добавляется автоматически (тот, кто сейчас авторизован в системе). Для добавления полиса нужно закрепить хотя бы одно лицо, допущенное к управлению (можно добавить новое или выбрать из тех, кто уже добавлен в системе благодаря представлению `choosePersonsAllowedToDrive.fxml`, которое обрабатывается контроллером `ChoosePersonsAllowedToDriveController`)

`ChangePolicyController` - контроллер для представления `changePolicy.fxml`. На этом представлении можно изменять только поля

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		41

Страховая премия и Дата окончания. Также можно изменить список лиц, допущенных к управлению.

Из представления изменения полиса можно перейти в представление добавления страхового случая - addInsuranceEvent.fxml, которое обрабатывается контроллером AddInsuranceEventController. На данной fxml основные поля для ввода это: Дата, Страховая выплата и Описание.

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		42

4. Руководство пользователя

При запуске приложения появляется форма авторизации, представленная на рисунке 20 в приложении 2. После введения правильного логина и пароля откроется основная форма приложения.

В меню есть пункт компоненты. Там можно выбрать Автомобиля, Лицо, допущенное к управлению, Страхователя или Сотрудника. По этим пунктам можно провести добавление (рисунки 21, 24, 27, 30 в приложении 2), изменение (рисунки 22, 25, 28, 31 в приложении 2) или посмотреть весь список (рисунки 23, 26, 29, 32 в приложении 2). Изменение происходит после поиска сущности по одному из полей, также такой поиск можно провести при просмотре всего списка. Также если выбрать одну строку в таблице и нажать на кнопку “Изменить”, то появится форма изменения.

Также в меню есть пункт полис. Там можно выбрать добавление полиса (рисунок 33 в приложении 2), изменение (рисунок 34 в приложении 2) или просмотр всего списка (рисунок 37 в приложении 2). При изменении полиса можно добавить страховой случай, что показано на рисунке 35 в приложении 2. После изменения полиса появляется окно с подробной информацией (рисунок 36 в приложении 2). Также это окно можно вывести при выборе одной из строк в таблице всех полисов.

Последний пункт меню — это отчёт. После выбора вида страхования и периода формируется финансовый отчёт деятельности компании, что представлено на рисунке 38 в приложении 2.

5. Тестирование АИС

Одним из важнейших этапов создания приложения является его тестирование и отладка. Тестирование позволяет выявить скрытые и явные недостатки программы, либо убедиться в ее пригодности для применения. Обнаруженные недостатки устраняются в ходе отладки.

Целью тестирования является проверка работоспособности программы, правильности выполнения всех функций, а также правильности обработки всех исключений, возникающих в ходе работы программы.

5.1 Авторизация

В классе Database обрабатываются следующие ошибки для данной страницы:

- неправильно указан логин и/или пароль;
- произошла ошибка в работе БД.

При отсутствии ошибок пользователь переходит в саму программу.

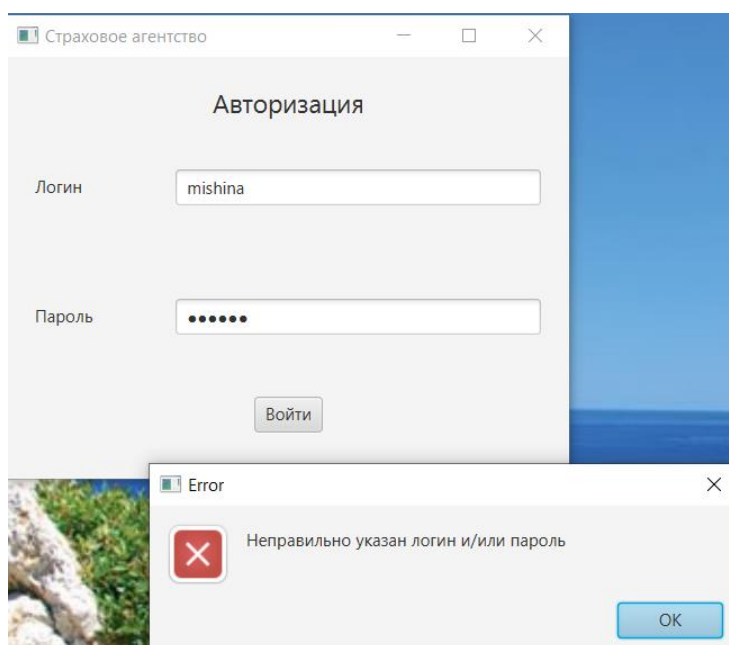


Рисунок 7 - скриншот вывода ошибок

5.2 Добавление

Ошибки при добавлении сотрудника

На данной странице обрабатываются следующие ошибки:

- незаполненные поля;
- введён номер телефона, который больше 15 символов;
- введена серия паспорта, которая не содержит 4 цифры;
- введён номер паспорта, который не содержит 6 цифр;
- введены серия и номер паспорта, которые содержат не только цифры;
- введён логин, длина которого меньше 4 или больше 32 символов;
- введён логин, который содержит пробелы;
- введён пароль, длина которого меньше 4 или больше 32 символов;
- введён пароль, который содержит пробелы.

В классе Database обрабатываются следующие ошибки для данной страницы:

- введен телефон, который уже используется;
- введен паспорт, который уже используется;
- введен логин, который уже занят;
- произошла ошибка в работе БД.

При отсутствии ошибок страхователь успешно добавляется и выводится сообщение "Сотрудник успешно добавлен".

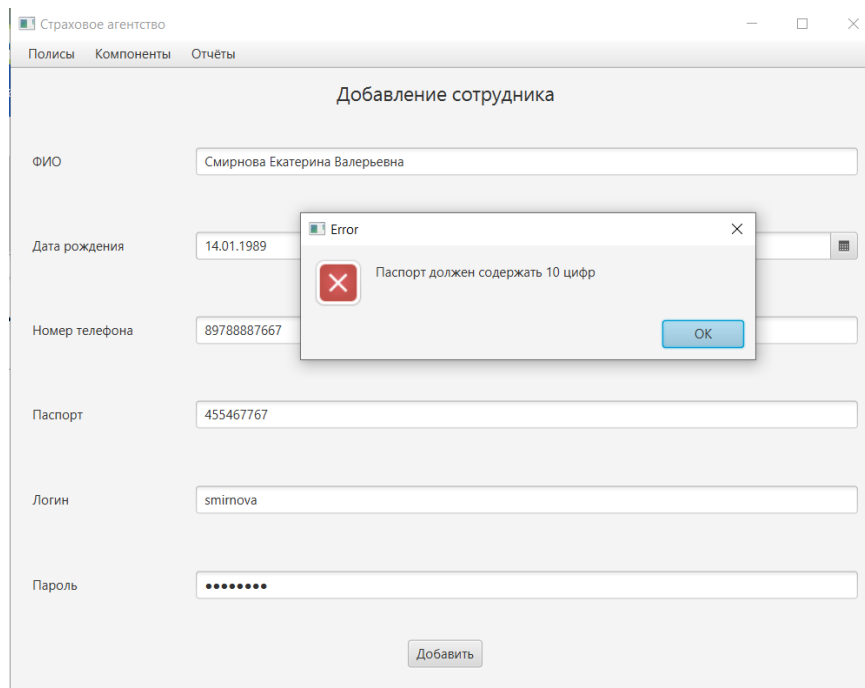


Рисунок 8 - скриншот вывода ошибок

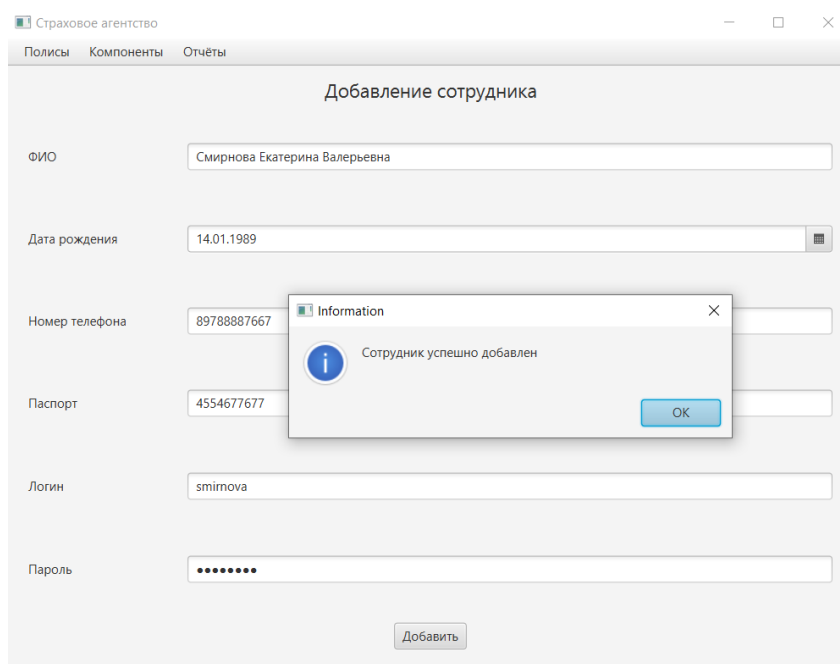


Рисунок 9 - скриншот успешного добавления

Аналогично выводятся ошибки или сообщение об успешном добавлении при добавлении страхователя, лица допущенного к управлению.

Ошибки при добавлении страхователя

На данной странице обрабатываются следующие ошибки:

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		46

- незаполненное поле ФИО;
- незаполненное поле Номер телефона;
- введен номер телефона, который содержит больше 15 символов;
- введена серия паспорта, которая не содержит 4 цифры;
- введен номер паспорта, который не содержит 6 цифр;
- введены серия и номер паспорта, которые содержат не только цифры.

В классе Database обрабатываются следующие ошибки для данной страницы:

- введен телефон, который уже используется;
- введен паспорт, который уже используется;
- произошла ошибка в работе БД.

Ошибки при добавлении лица, допущенного к управлению

На данной странице обрабатываются следующие ошибки:

- незаполненное поле ФИО;
- введена серия водительского удостоверения, которая не содержит 4 цифры;
- введен номер водительского удостоверения, который не содержит 6 цифр;
- введены серия и номер водительского удостоверения, которые содержат не только цифры.

В классе Database обрабатываются следующие ошибки для данной страницы:

- введено водительское удостоверение, которое уже используется;
- произошла ошибка в работе БД.

Ошибки при добавлении автомобиля

На данной странице обрабатываются следующие ошибки:

					МИВУ 09.03.04-8.000 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

- незаполненное поле Модель;
- введенный VIN номер не содержит 17 знаков;
- введенный VIN номер состоит не только из цифр и заглавных латинских букв;
- незаполненное поле Регистрационный знак;
- введенная серия паспорта не содержит 4 знака;
- введенные первые два символа серии паспорта ТС не цифры;
- введенные последние два символа серии паспорта ТС не буквы;
- введенный номер паспорта ТС не содержит 6 цифр;
- введенный номер паспорта ТС содержит не только цифры;
- не была добавлена ни одна фотография автомобиля.

В классе Database обрабатываются следующие ошибки для данной страницы:

- введен VIN номер, который уже используется;
- произошла ошибка в работе БД.

5.3 Изменение/удаление

Ошибки при изменении/удалении сотрудника

На данной странице обрабатываются следующие ошибки:

- незаполненная строка поиска;
- не выбран сотрудник;
- незаполненные поля;
- введен номер телефона, который содержит больше 15 символов;
- введена серия паспорта, которая не содержит 4 цифры;
- введен номер паспорта, который не содержит 6 цифр;
- введены серия и номер паспорта, которые содержат не только цифры;
- введен логин, длина которого меньше 4 или больше 32 символов;

- введён логин, который содержит пробелы;
- введён пароль, длина которого меньше 4 или больше 32 символов;
- введён пароль, который содержит пробелы.

В классе Database обрабатываются следующие ошибки для данной страницы:

- введён телефон или паспорт сотрудника, который не существует (для строки поиска);
- введён телефон, который уже используется;
- введён паспорт, который уже используется;
- введён логин, который уже занят;
- попытка удаления сотрудника, который оформил полис;
- произошла ошибка в работе БД.

Также в форме нельзя изменить дату рождения.

При отсутствии ошибок сотрудник успешно изменяется и выводится сообщение "Сотрудник успешно изменён" или удаляется с сообщением "Сотрудник успешно удалён".

Страхование агентство

Полисы Компоненты Отчёты

Изменение/удаление сотрудника

Введите номер телефона или паспорт Поиск

ФИО

Дата рождения

Номер телефона

Паспорт

Логин

Пароль

Изменить Удалить

Error

Данный логин уже используется

OK

Рисунок 10 - скриншот вывода ошибок

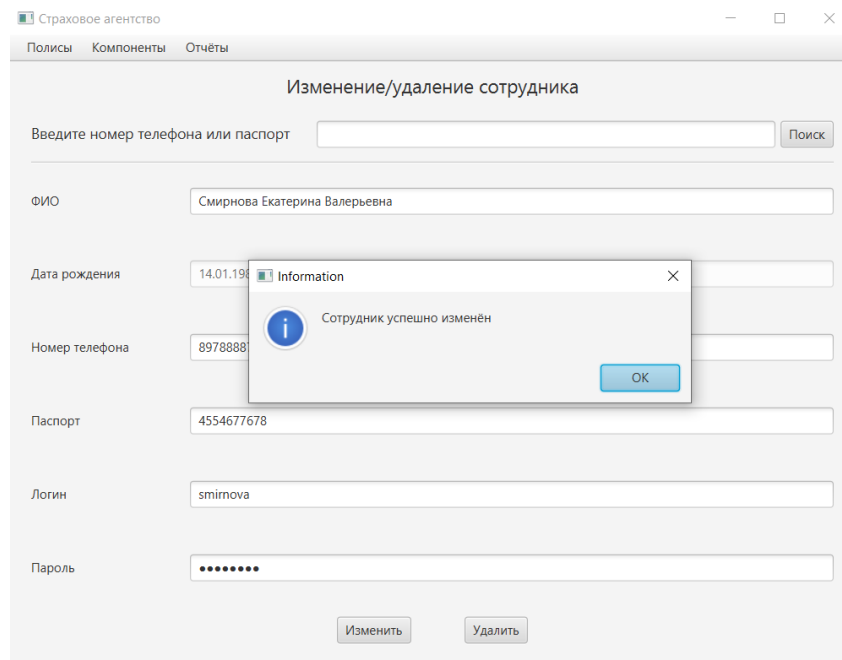


Рисунок 11 - скриншот успешного изменения

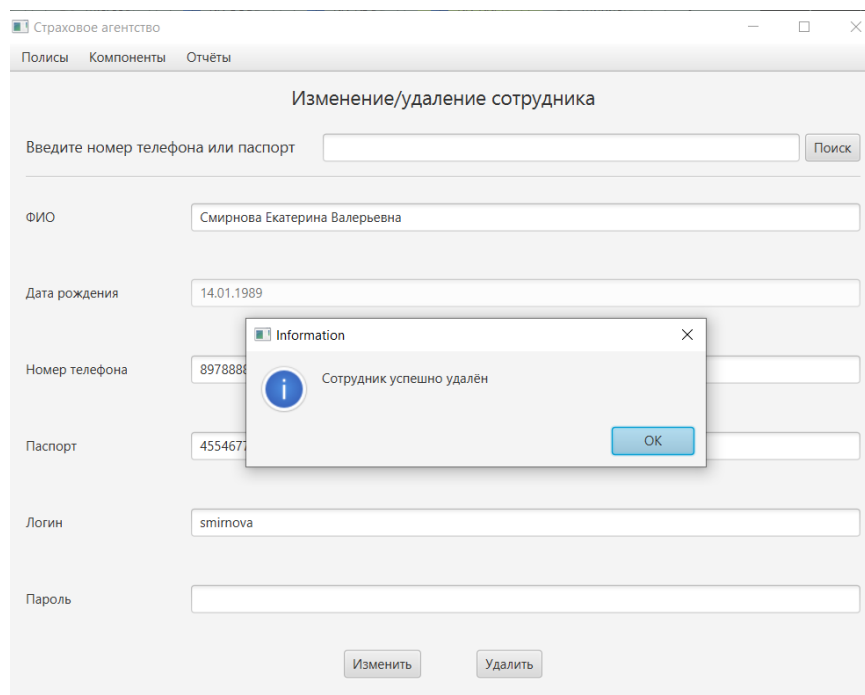


Рисунок 12 - скриншот успешного удаления

Ошибки при изменении/удалении страхователя

На данной странице обрабатываются следующие ошибки:

- незаполненная строка поиска;

- не выбран страхователь (если пользователь нажал кнопку изменить, но до этого в строке поиска не нашёл нужную информацию);
- незаполненное поле ФИО;
- незаполненное поле Номер телефона;
- введён номер телефона, который содержит больше 15 символов;
- введена серия паспорта, которая не содержит 4 цифры;
- введён номер паспорта, который не содержит 6 цифр;
- введены серия и номер паспорта, которые содержат не только цифры.

В классе Database обрабатываются следующие ошибки для данной страницы:

- введён телефон или паспорт страхователя, который не существует (для строки поиска);
- введён телефон, который уже используется;
- введён паспорт, который уже используется;
- попытка удаления страхователя, который оформил полис;
- произошла ошибка в работе БД.

Также в форме нельзя изменить дату рождения.

Ошибки при изменении/удалении лица, допущенного к управлению

На данной странице обрабатываются следующие ошибки:

- незаполненная строка поиска;
- не выбран водитель (если пользователь нажал кнопку изменить, но до этого в строке поиска не нашёл нужную информацию);
- незаполненное поле ФИО;
- введена серия водительского удостоверения, которая не содержит 4 цифры;
- введён номер водительского удостоверения, который не содержит 6 цифр;

– введены серия и номер водительского удостоверения, которые содержат не только цифры.

В классе Database обрабатываются следующие ошибки для данной страницы:

- данный водитель не существует (для строки поиска);
- введено водительское удостоверение, который уже используется;
- попытка удаления лица, допущенного к управлению, на которое оформлен полис;
- произошла ошибка в работе БД.

Ошибки при изменении/удалении автомобиля

На данной странице обрабатываются следующие ошибки:

- незаполненная строка поиска;
- не выбран автомобиль (если пользователь нажал кнопку изменить, но до этого в строке поиска не нашёл нужную информацию);
- незаполненное поле Регистрационный знак;
- введённая серия паспорта не содержит 4 знака;
- введённые первые два символа серии паспорта ТС не цифры;
- введённые последние два символа серии паспорта ТС не буквы;
- введённый номер паспорта ТС не содержит 6 цифр;
- введённый номер паспорта ТС содержит не только цифры;
- в список фотографий не добавлена ни одна фотография.

В классе Database обрабатываются следующие ошибки для данной страницы:

- введён VIN номер автомобиля, которого не существует (для строки поиска);
- попытка удаления автомобиля, на который оформлен полис;
- произошла ошибка в работе БД.

[illegible]

Рисунок 14 - скриншот поиска сотрудника

5.5 Добавление полиса

На данной странице обрабатываются следующие ошибки:

- незаполненное поле Вид страхования;
- незаполненное поле Страховая сумма;
- введена страховая сумма, которая не является целым числом;
- незаполненное поле Страховая премия;
- введена страховая премия, которая не является целым числом;
- незаполненное поле Срок действия;
- введён VIN номер, который не содержит 17 знаков;
- введён VIN номер, который состоит не только из цифр и заглавных латинских букв;
- список лиц, допущенных к управлению пуст;
- введён водитель, который уже добавлен (при добавление нового водителя в список лиц, допущенных к управлению);

– выбран водитель, который не существует в списке добавленных водителей (при удалении водителя из списка лиц, допущенных к управлению).

В классе Database обрабатываются следующие ошибки для данной страницы:

– введён автомобиль, который не существует (добавление автомобиля к полису);

– введён сотрудник, который не существует (добавление сотрудника к полису);

– введён водитель, который не существует (добавление нового водителя в список лиц, допущенных к управлению);

– введён страхователь, которого нет (добавление полиса);

– введён автомобиль, которого нет (добавление полиса);

– введён сотрудник, которого нет (добавление полиса);

– произошла ошибка в работе БД.

При отсутствии ошибок полис успешно добавляется и выводится сообщение "Полис успешно добавлен".

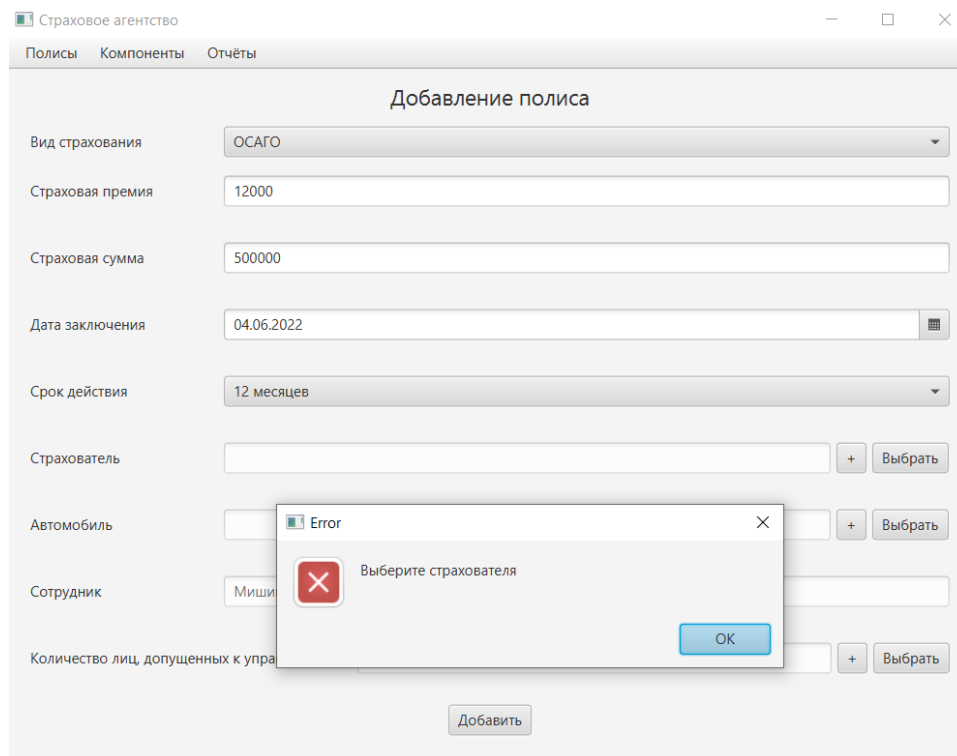


Рисунок 15 - скриншот вывода ошибок

5.6 Изменение полиса

На данной странице обрабатываются следующие ошибки:

- незаполненное поле Страховая премия;
- введена страховая премия, которая не является целым числом;
- введена дата окончания действия, которая меньше даты заключения;
- введена дата окончания действия, которая меньше даты последнего страхового случая;
- список лиц, допущенных к управлению пуст;
- введён водитель, который уже добавлен (при добавление нового водителя в список лиц, допущенных к управлению);
- выбран водитель, который не существует в списке добавленных водителей (при удалении водителя из списка лиц, допущенных к управлению).

В классе Database обрабатываются следующие ошибки для данной страницы:

- введён водитель, который не существует (проверка при добавление нового водителя в список лиц, допущенных к управлению);
- произошла ошибка в работе БД.

Также в форме нельзя изменить вид страхования, страховую сумму, дату заключения, закреплённого страхователя, автомобиль и сотрудника.

При отсутствии ошибок полис успешно изменяется и выводится подробная информация об изменённом полисе.

Страхование

Полисы Компоненты Отчёты

Изменение полиса

Вид страхования: КАСКО

Страховая премия: 4000

Страховая сумма: 500000

Дата заключения: 03.06.2022

Дата окончания: 03.12.2022

Страхователь: Гаврилов Борис Дмитриевич (89017654595)

Автомобиль: KIA Seed (2G) (UN1CZ24A8LX094080)

Сотрудник: Александров Филипп Сергеевич (89665556734)

Лица, допущенные к управлению

ФИО	Водительское удостоверение
Гаврилов Борис Дмитриевич	2378382438
Глухов Максим Александрович	7342367288

Страховые случаи

Дата	Страховая выплата	Описание
No content in table		

Рисунок 16 - скриншот вывода ошибок

5.7 Добавление страхового случая

На данной странице обрабатываются следующие ошибки:

- выбрана дата, которая меньше даты заключения полиса;
- выбрана дата, которая больше даты окончания действия полиса;
- незаполненное поле Страховая выплата;
- введена страховая выплата, которая не является целым числом;
- введена страховая выплата, которая больше Страховой суммы.

В классе Database обрабатываются следующие ошибки для данной страницы:

- произошла ошибка в работе БД.

При отсутствии ошибок страховой случай успешно добавляется и выводится сообщение "Страховой случай успешно добавлен".

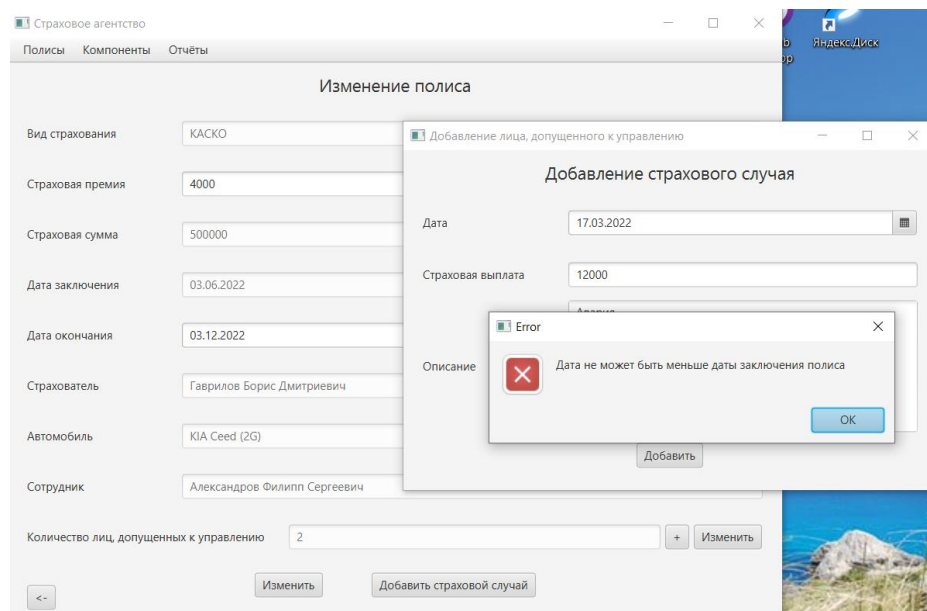


Рисунок 17 - скриншот вывода ошибок

5.8 Отчёты

На данной странице обрабатываются следующие ошибки:

- незаполненное поле Вид страхования;
- выбрана дата начала, которая не больше даты окончания.

В классе Database обрабатываются следующие ошибки для данной страницы:

– произошла ошибка в работе БД.

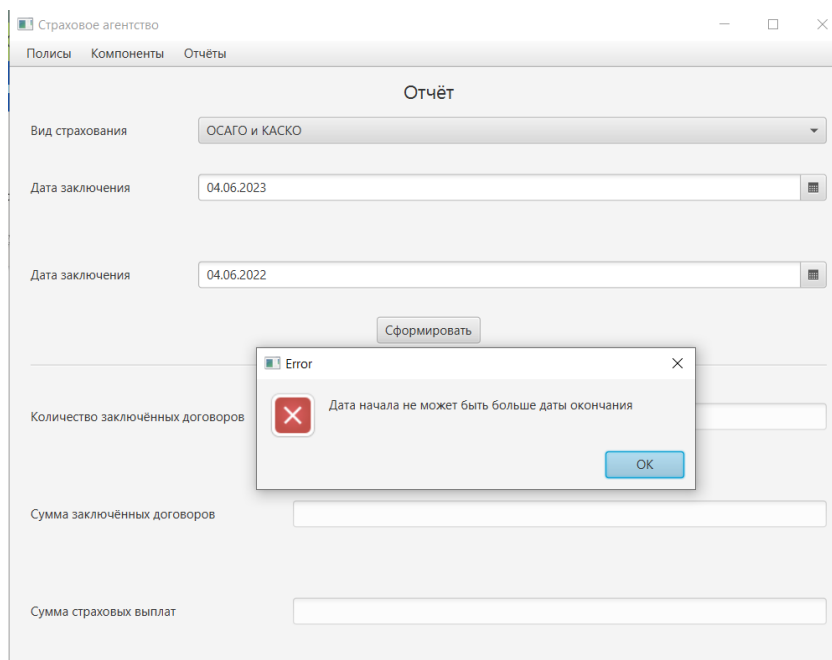


Рисунок 18 - скриншот вывода ошибок

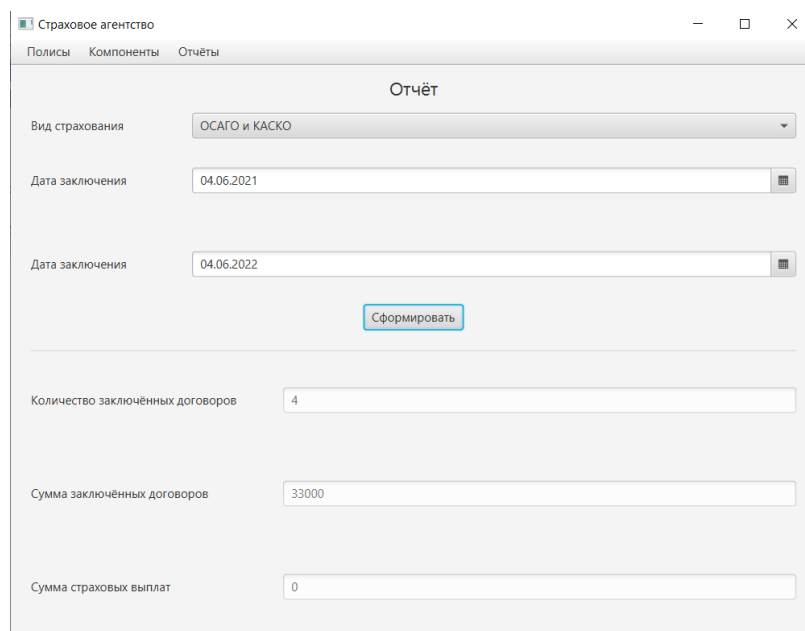


Рисунок 19 - скриншот отчёта

Заключение

В данной курсовой работе в соответствии с заданием была разработана АИС Страхового агентства.

В ходе выполнения курсовой работы были выполнены следующие задачи:

- выявлены требования к программе;
- разработаны модели данных;
- создана база данных;
- разработана программа;
- осуществлено ее тестирование.

Разработанная программа обеспечивает осуществление следующих функций:

- добавление данных о страхователях, сотрудниках, автомобилях, лицах, допущенных к управлению;
- добавление полисов;
- возможность работы со страховыми случаями;
- возможность изменения информации;
- возможность удаления выбранной информации;
- предоставление информации на форме в табличном виде;
- возможность анализировать количество и суммы заключенных договоров по каждому из видов, а также оценивать риски, подсчитывая суммы страховых выплат по каждому виду договоров, а также составлять финансовый отчет деятельности компании за заданный период времени.

Список используемой литературы

1. Аналог Полисы ОСАГО 1.0.9: [Электронный ресурс] // URL: <https://araxgroup.ru/index.php/products/53-other-programs/271-polisi-osago> (Дата обращения – 1.03.2022)

2. Аналог Учёт автострахования ОСАГО: [Электронный ресурс] // URL: http://bestsoft.moy.su/load/ofis_programmy_dlja_kompjutera_skachat_besplatno/ofis_programmy_dlja_kompjutera_skachat_besplatno/skachat_besplatno_programmu_uchjot_i_zapolnenie_polisov_avtostrakhovanija_osago_5_6_bez_registracii_i_sms_licenzionnyj_kljuch_aktivacii/11-1-0-108 (Дата обращения – 1.03.2022)

3. Руководство по языку программирования Java: [Электронный ресурс] // URL: <https://metanit.com/java/tutorial/> (Дата обращения – 11.05.2022)

4. Вязовик, Н. А. Программирование на Java : учебное пособие / Н. А. Вязовик. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 601 с. — ISBN 978-5-4497-0852-6. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/102048.html> (Дата обращения – 30.04.2022)

Приложение 1. Текст программы

Полный код проекта приведён на сайте GitHub по ссылке:
<https://github.com/vivir-para-volar/JavaAppInsuranceAgency>

Там же приведена документация JavaDoc по проекту в папке
documentation.

					МИВУ 09.03.04-8.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

Приложение 2. Снимки окон программы (скриншоты программы)

Страховое агентство

Авторизация

Логин mishina

Пароль

Войти

Рисунок 20 - окно авторизации

Страховое агентство

Полисы Компоненты Отчёты

Добавление автомобиля

Модель Лада 2101

VIN номер JW2346TR567823YU12

Регистрационный знак H112345

Паспорт ТС 12GH121212

Фото автомобиля

Загрузить

04.06.2022

Добавить

Рисунок 21 - окно добавления автомобиля

Страховое агентство

ПолисыКомпонентыОтчёты

Изменение/удаление автомобиля

Введите VIN номер

Поиск

Модель

Opel Astra H

VIN номер

1GCESBFE2C8192977

Регистрационный знак

E100EE22


Паспорт ТС

45TE823924

Фото автомобиля

Загрузить

<



>

04.06.2022

Изменить

Удалить

Рисунок 22 - окно изменения автомобиля

Страховое агентство

Полисы Компоненты Отчёты

Список автомобилей

Введите VIN номер

Модель	VIN номер	Регистрационный знак	Паспорт ТС
Hyundai Santa Fe (3G)	3GNDA23D08S544387	ОP897Ш09	67Тб907867
Kia Ceed (2G)	JN1CZ24A8LX094080	AЛ090PB12	25TЧ082337
Renault Fluence	3N1CC1AP8AL392058	E222XC22	89TC348280
Land Rover Discovery II	2G1125S33F9179984	ПН923А33	78ТА373434
Rover 45	1N6BF0KL5CN138975	A343PP90	34TB734342
Лада 2101	JN8AF5MV4DT293416	B0378P79	38TE489348
Opel Astra H	1GCESBFE2C8192977	E100EE22	45TE823924

Рисунок 23 - окно списка автомобилей

Страховое агентство

Полисы Компоненты Отчёты

Добавление лица, допущенного к управлению

ФИО

Водительское удостоверение

Рисунок 24 - окно добавления лица, допущенного к управлению

Страховое агентство

Полисы Компоненты Отчёты

Изменение/удаление лица, допущенного к управлению

Введите водительское удостоверение

ФИО

Водительское удостоверение

Рисунок 25 - окно изменения лица, допущенного к управлению

Страховое агентство

Полисы Компоненты Отчёты

Список лиц, допущенных к управлению

Введите водительское удостоверение

ФИО	Водительское удостоверение
Бессонов Иван Савельевич	7345901132
Васильева Мария Ильинична	2929222999
Воробьева Вера Матвеевна	1907540089
Гаврилов Борис Дмитриевич	2378382438
Глухов Максим Александрович	7342367288
Колесова Эмилия Тимофеевна	8734284883
Маркова Таисия Серафимовна	3489344688
Морозова Софья Яновна	2255505432
Попов Иван Александрович	1731634821
Сидоров Максим Максимович	7292742382
Степанов Роман Николаевич	5278982365
Шестаков Даниил Данилович	4397562113

Рисунок 26 - окно списка лиц, допущенных к управлению

Страховое агентство

Полисы Компоненты Отчёты

Добавление страхователя

ФИО

Дата рождения

Номер телефона

Паспорт

Рисунок 27 - окно добавления страхователя

Страховое агентство

ПолисыКомпонентыОтчёты

Изменение/удаление страхователя

Введите номер телефона или паспорт

Поиск

ФИО

Корнилов Ярослав Максимович

Дата рождения

12.01.1976

Номер телефона

89200755299

Паспорт

1220729563

Изменить

Удалить

Рисунок 28 - окно изменения страхователя

[illegible]

Рисунок 29 - окно списка всех страхователей

Страховое агентство

Полисы Компоненты Отчёты

Добавление сотрудника

ФИО

Дата рождения

Номер телефона

Паспорт

Логин

Пароль

Рисунок 30 - окно добавления сотрудника

Страховое агентство

Полисы Компоненты Отчёты

Изменение/удаление сотрудника

Введите номер телефона или паспорт

ФИО

Дата рождения

Номер телефона

Паспорт

Логин

Пароль

Рисунок 31 - окно изменения сотрудника

Страховое агентство

Полисы Компоненты Отчёты

Список сотрудников

Введите номер телефона или паспорт

ФИО	Дата рождения	Номер телефона	Паспорт	Логин
Александров Филипп Се...	05.09.1999	89665556734	1799545433	aleksandrov
Мишина Екатерина Фед...	19.05.1992	89346566645	1298454567	mishina
Морозова Александра ...	15.03.1998	89340043434	1765767676	morozova
Новиков Владимир Свят...	21.10.1987	89209430243	2387006767	novikov

Рисунок 32 - окно списка сотрудников

Страховое агентство

Полисы Компоненты Отчёты

Добавление полиса

Вид страхования: ОСАГО

Страховая премия: 8000

Страховая сумма: 500000

Дата заключения: 04.06.2022

Срок действия: 12 месяцев

Страхователь: Корнилов Ярослав Максимович

Автомобиль: Rover 45

Сотрудник: Мишина Екатерина Федоровна

Количество лиц, допущенных к управлению: 2

Рисунок 33 - окно добавления полиса

Страхование агентство

Полисы Компоненты Отчёты

Изменение полиса

Вид страхования: КАСКО

Страховая премия: 9000

Страховая сумма: 600000

Дата заключения: 03.06.2022

Дата окончания: 03.06.2023

Страхователь: Гаврилов Борис Дмитриевич

Автомобиль: Hyundai Santa Fe (3G)

Сотрудник: Мишина Екатерина Федоровна

Количество лиц, допущенных к управлению: 1

Изм. Изменить Добавить страховой случай

Рисунок 34 - окно изменения полиса

Добавление лица, допущенного к управлению

Добавление страхового случая

Дата: 04.06.2022

Страховая выплата: 40000

Описание: Авария

Добавить

Рисунок 35 - окно добавления страхового случая

Страховое агентство

Полисы Компоненты Отчёты

Отчёт

Вид страхования: ОСАГО и КАСКО

Дата заключения: 04.06.2021

Дата заключения: 04.06.2022

[Сформировать](#)

Количество заключённых договоров: 4

Сумма заключённых договоров: 33000

Сумма страховых выплат: 42000

Рисунок 38 - окно отчётов

Изм.	Лист	№ докум.	Подпись	Дата

МИВУ 09.03.04-8.000 ПЗ

Лист

72