Одесский Национальный Политехнический Университет

Кафедра информационных систем

Расчетно-графическая работа

по дисциплине: «Технологии компьютерного проектирования»

Тема: «Проектирование информационной системы страховой компании»

Выполнил:

Ст. группы АИ-166

Дидух Э. Г.

Проверил:

Галченков О. Н.

Одесса, 2018

Цель работы: закрепление навыков проектирования структуры программного обеспечения в виде UML – модели с помощью case средства Star UML

Задание на РГР

1. Создать проект в Star UML

2. Разработать диаграмму прецедентов (общий вариант, 3-5 прецедентов).

3. Разработать поток событий и присоединить его к диаграмме прецедентов.

4. Разработать детальные дополнительные диаграммы прецедентов для каждого из прецедентов основной диаграммы и присоединить их к основной диаграмме прецедентов.

5. Произвести документирование элементов на основной и дополнительных диаграммах прецедентов.

6. Для каждого из сценариев, отображённых в потоке событий, построить диаграмму активностей и присоединить к общей или детальной диаграмме прецедентов.

7. Разработать диаграмму классов с использование паттерна MVC и других.

8. Разработать диаграмму компонентов с разбиением по слоям.

9. Разработать диаграммы последовательностей для основного сценария .

10. Разработать диаграммы последовательностей для альтернативных и ошибочных сценариев и прикрепить их к диаграмме последовательностей основного сценария.

11. Разработать диаграммы кооперации, соответствующие построенным диаграммам последовательностей.

12. Привести в соответствие сообщения на диаграммах последовательностей и диаграммах кооперации с операциями классов.

13. Разработать диаграмму развёртывания.

14. При необходимости, дополнительно разработать другие виды диаграмм.

15. Оформить пояснительную записку к РГР.

В рамках данной расчетно-графической работы будет представлен процесс проектирования информационной системы страховой компании. Для проектирования используется паттерн MVC.

Первый этап проектирования системы – построение основной диаграммы прецедентов.

Диаграмма вариантов использования (диаграмма прецедентов, use case diagram) — это диаграмма, на которой изображаются отношения между экторами и вариантами использования.

С помощью этой диаграммы можно:

* Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* Сформулировать общие требования к функциональному поведению проектируемой системы;
* Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования (прецедентов) представляет собой граф, в вершинах которого расположены актеры или прецеденты, связи между вершинами – это разного вида отношения.

В связи со спецификой проекта и ориентированием на веб-разработку определим таких экторов для дальнейшего использования на диаграмме прецедентов:

* Пользователь – клиент страхового агентства
* Агент – работник страхового агентства, привязанный к конкретному клиенту
* Электронная почта – электронный почтовый ящик пользователя/агента, зарегистрированного в системе
* Кредитная система – электронная служба денежных транзакций для обеспечения оплаты страховых услуг

Информационная система страховой компании подразумевает использование базы данных для хранения всей необходимой информации: данных пользователей, каталогов страховых полисов, историю работы агентов и так далее. Также в функциональные возможности системы должны входить такие функции как:

* Регистрация пользователя
* Авторизация
* Редактирование личного профиля
* Просмотр каталога страховых полисов
* Оформление страховки
* Социальное взаимодействие с агентом

Для описания указанных возможностей системы был выбран Rational Approach и создан новый проект StarUML. Далее была создана общая диаграмма прецедентов. В нашем случае в функциональные возможности системы входят такие обобщенные прецеденты:

* Работа с профилем пользователя
* Работа со страховыми полисами
* Чат

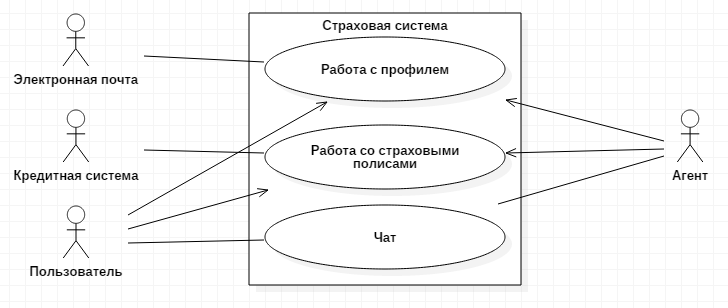


Рисунок 1 - Общая диаграмма прецедентов

Из представленных на диаграмме экторов активное участие в деятельности системы принимают Пользователь и Агент. Электронная почта и Кредитная система используются одноразово или в отдельных прецедентах для подтверждения операций, выполняемых Пользователем/Агентом.

Приведенная выше диаграмма включает в себя всех экторов, взаимодействующих с системой и обобщенное описание ее функциональных возможностей. Для более детального рассмотрения функциональных возможностей программного продукта приведем диаграммы прецедентов более низкого уровня:

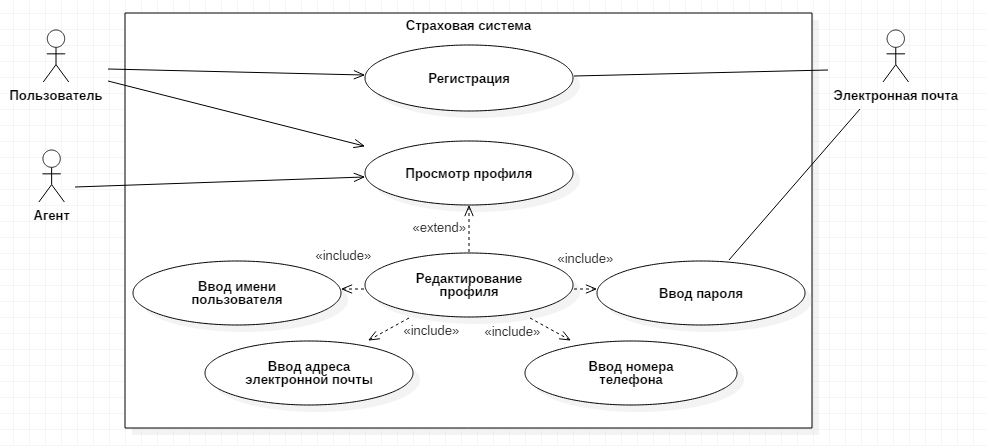


Рисунок 2 – Дополнительная диаграмма прецедента работы с профилем

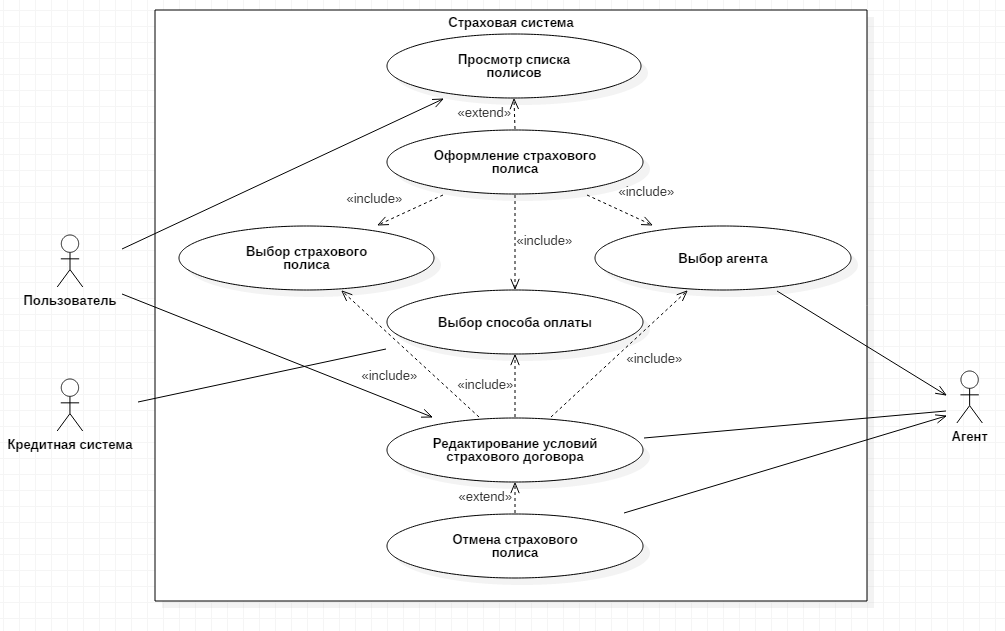


Рисунок 3 – Дополнительная диаграмма прецедента работы со страховыми полисами



Рисунок 4 – Дополнительная диаграмма прецедента чата

Для лучшего понимания сути и функциональности прецедентов/экторов не лишним будет проводить документацию объектов диаграмм. Документирование элементов модели в StarUML производится с помощью поля Documentation. Документация должна в полной мере описывать свойства и возможности объекта и его роль в системе.

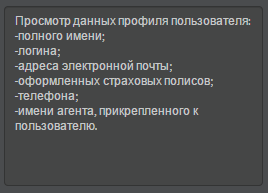


Рисунок 5 – Пример документирования объекта в StarUML

Одним из требований языка UML является самодостаточность диаграмм для представления информации о моделях проектируемых систем. Однако, как уже отмечалось выше, диаграммы вариантов использования описывают то, что делает система, без уточнения того, как она это делает.

Для реального описания системы потребуются более специфические данные, которые отражены в потоке событий. Потоки событий уточняют или детализируют последовательность действий, совершаемых системой при выполнении ее вариантов использования, а также описывают логику переходов через варианты использования.

Поток событий – это определенная последовательность действий, которая описывает действия актеров и поведение моделируемой системы в форме обычного текста.

Потоки событий – это текстовые описания пошагового выполнения прецедентов, они понятны не только разработчику, но и стороннему читателю. Их задача – еще больше детализировать описание функциональности системы до того, как разработчики приступят к написанию программного кода, и устранить возможное недопонимание требуемой функциональности, как можно больше сблизить представления разработчика о системе и заказчика.

Потоки событий бывают трех типов: основной, альтернативный и поток ошибок.

Основной (главный) поток описывает наилучший сценарий либо наиболее используемый путь исполнения прецедента.

Альтернативный поток специфицирует отклонения от основного потока, которые не рассматриваются как ошибочные.

Поток ошибок рассматривается как отклонение от альтернативного или основного, которое порождает условия формирования ошибки.

В рамках описания прецедентов созданной ранее диаграммы было составлено основных потока событий:

Основной поток событий работы с профилем:

1. Прецедент начинается с запуска системы пользователем

А1. Пользователь нажимает кнопку "Авторизация"

2. Пользователь нажимает кнопку "Регистрация"

3. Открывается окно регистрации

4. Пользователь вводит необходимые данные: логин, пароль, адрес электронной почты

5. Пользователь подтверждает введенные данные

А2. Пользователь отменяет регистрацию

6. Система отправляет письмо со ссылкой подтверждения регистрации на электронный адрес пользователя

Е1. Ошибка отправки письма

7. Пользователь проходит по предоставленной ссылке

8. Регистрация пользователя подтверждается

А3. Регистрация пользователя не подтверждается

9. Пользователь добавляется в базу данных

10. Система запрашивает авторизацию пользователя

11. Пользователь вводит необходимые данные

12. Проверяется наличие пользователя в базе данных

13. Пользователь подтвержден

14. Пользователь переадресовывается в свой личный кабинет

Основной поток событий работы со страховыми полисами:

1. Пользователь открывает список страховых полисов

2. Пользователь выбирает страховой полис

3. Происходит переадресация пользователя в окно оформления страховки

4. Пользователь выбирает срок страховки, условия контракта, способ оплаты, страхового агента

5. Пользователь подтверждает введенные данные

А1. Пользователь не подьверждает введенные данные

6. Происходит переадресация пользователя на страницу оплаты страхового полиса

7. Пользователь подтверждает оплату

А2. Пользователь отменяет оплату

8. Отправляется запрос на транзакцию кредитной системе

9. Ожидание окончания транзакции

Е1. Ошибка транзакции

10. Подтверждение оплаты полиса

11. Отправка оповещения об оформлении полиса выбраному агенту

12. Агент подтверждает принятие нового клиента

А3. Агент отклоняет нового клиента

13. Отправка подтверждения оформления полиса клиенту(пользователю)

14. Возврат к списку страховых полисов

Основной поток событий чата:

1. Пользователь заходит в окно чата

2. Пользователь отправляет текстовое сообщение агенту

3. Отправка оповещения о новом сообщении агенту

4. Ожидание открытия окна чата агентом

5. Агент входит в окно чата

6. Отправка оповещения пользователю о том, что сообщение прочитано

7. Агент отправляет ответное сообщение

8. Отправка оповещения пользователю о новом сообщении

9. Ожидание открытия окна чата пользователем

Для изображения потоков событий с помощью UML используются диаграммы активности(деятельности). Диаграммы деятельности отражают работу системы в динамике в соответствии с потоком событий, используя нотации. Основными нотациями этих диаграмм являются деятельность и переход.

Деятельностью называется исполнение определенного поведения в потоке управления системы. В UML деятельность изображается в виде скругленного прямоугольника с текстовым описанием внутри.

Переход показывает, как поток управления переходит от одной деятельности к другой. Обычно переход осуществляется по завершении деятельности.

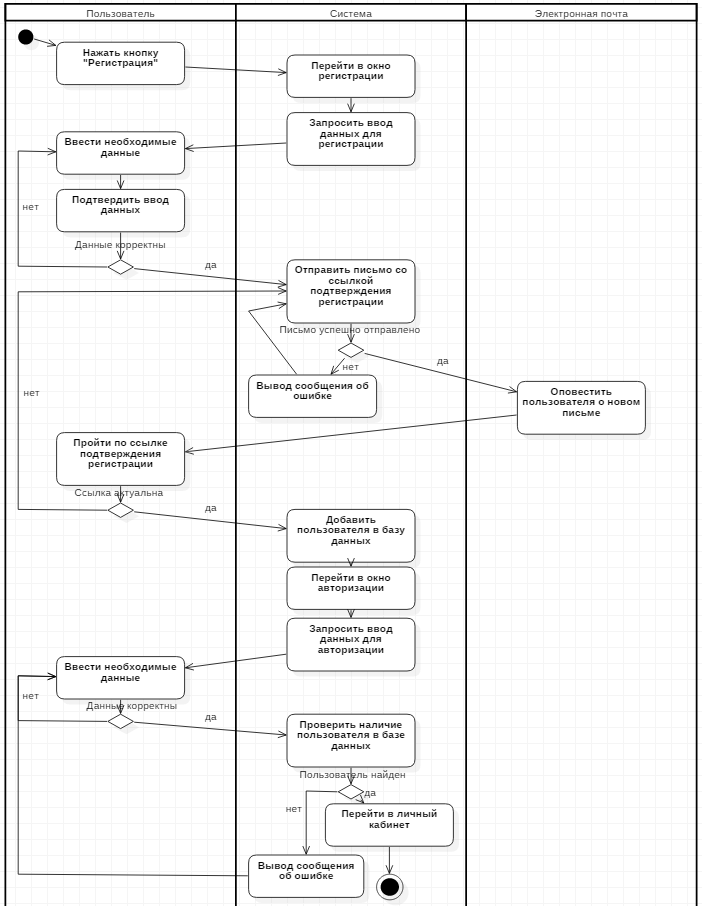


Рисунок 6 – Диаграмма деятельности основного потока событий работы с профилем

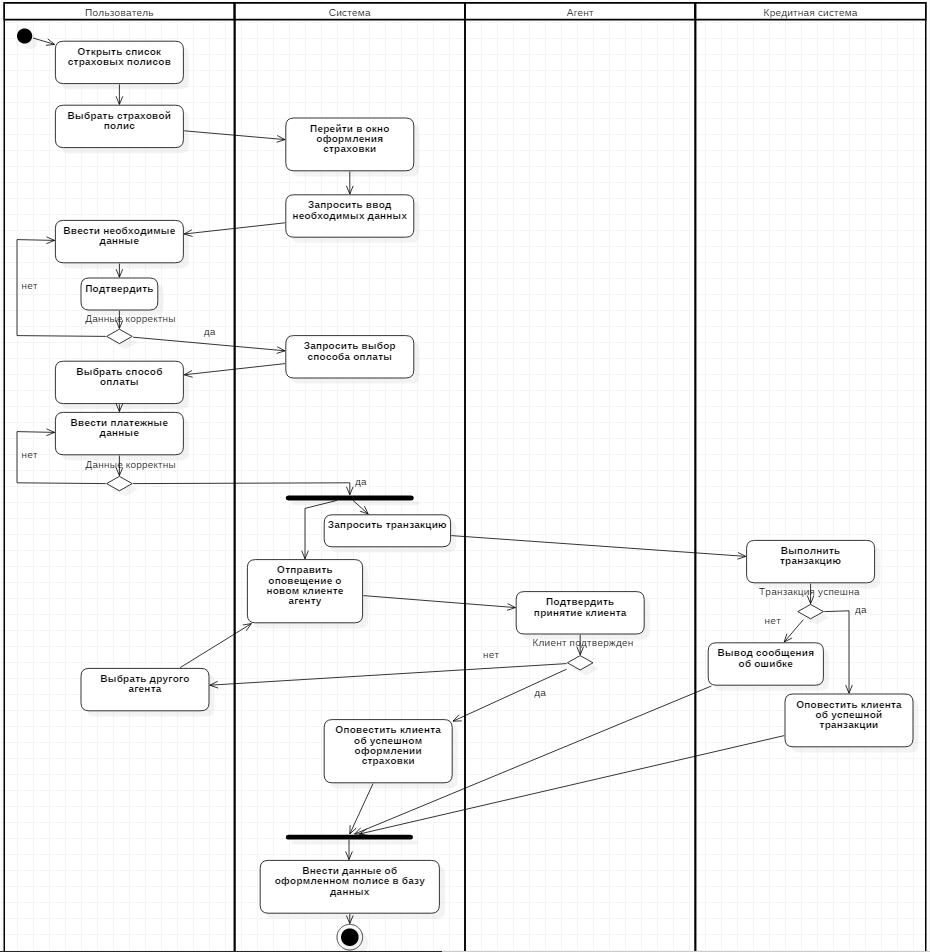


Рисунок 7 – Диаграмма деятельности основного потока событий работы с полисами

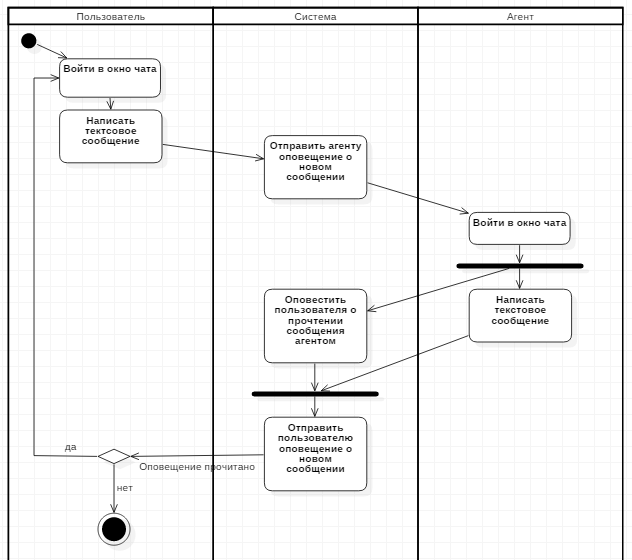
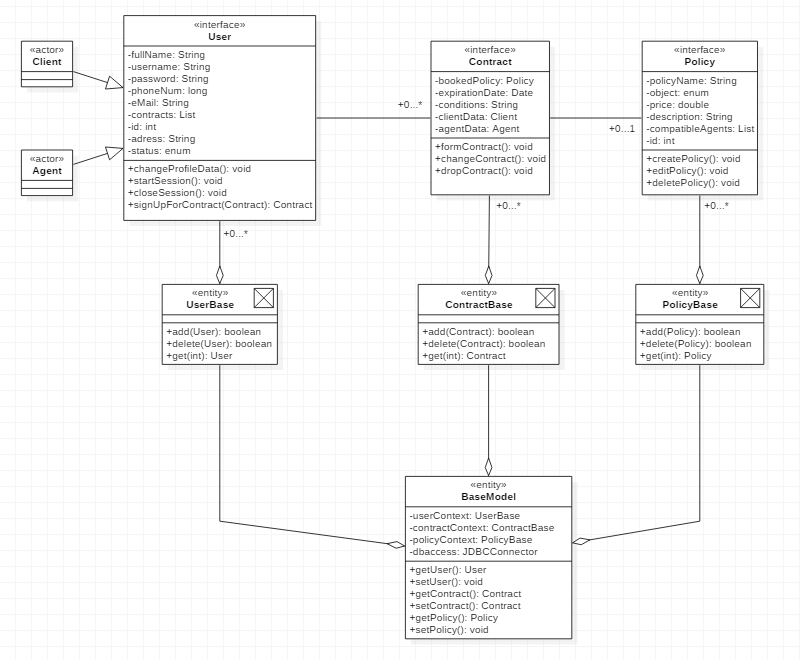


Рисунок 8 – Диаграмма деятельности основного потока событий чата

На данном этапе проектирования мы все ближе подходим к реализации задуманной системы и уровень абстракции снижается все больше. Следующим этапом является проектирование диаграмм классов.

Для реализации системы был выбран паттерн проектирования MVC(Model-View-Controller). Данный паттерн является проверенным временем и имеет хорошую эффективность и расширяемость. Основное преимущество MVC перед другими паттернами – четкое разделения системы на составляющие: Модель, Представление и Контроллер. Такой подход позволяет уменьшить связность кода и приветствует использование интерфейсов, что играет на руку расширяемости программного обеспечения. В заключении, для реализации MVC существует множество фреймворков на различных языках программирования, что еще больше упрощает разарботку.

Рисунок 9 – Диаграмма классов модели

Поскольку проект будет реализован в виде веб-приложения, все окна представления будут представлены HTML-страницами. Форматирование Веб страниц имеет стандарт, который подразумевает наличие Header, MainBody и Footer на странице. Для целей создания однотипных объектов хорошо подходит паттерн Builder. Builder отделяет создание сложного объекта от его представления, позволяя использовать один и тот же процесс разработки для создания различных представлений.

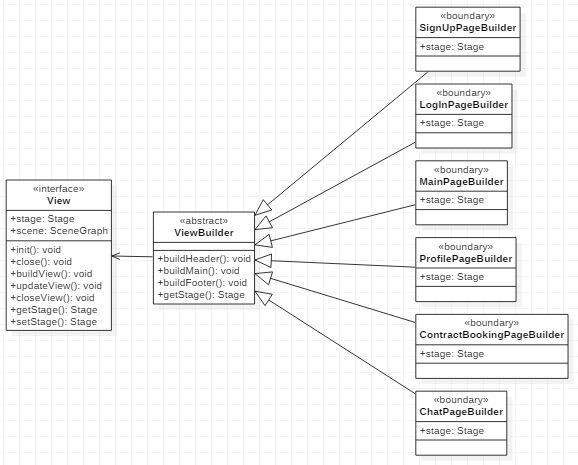


Рисунок 10 – Диаграмма классов представления

Оставшаяся для реализации часть паттерна MVC – контроллер. В проектировании классов-контроллеров не предусмотрено особой иерархии, поскольку зачастую контроллеры используются в качестве связующего звена между моделью и представлением и занимаются обработкой пользовательского взаимодействия с программой.

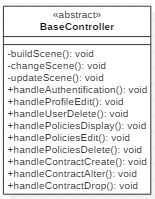


Рисунок 11 – Диаграмма классов контроллера

В соответствии с паттерном и диаграммами классов создана диаграмма компонентов, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами.

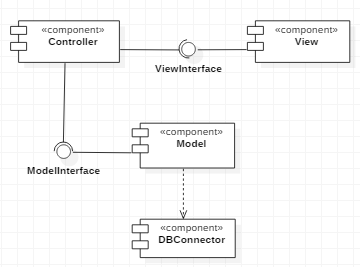


Рисунок 12 – Диаграмма компонентов

Для лучшего представления взаимодействия между компонентами программы во времени существует диаграмма последовательностей. На диаграммах последовательности изображаются объекты, классы и последовательность сообщений, которыми обмениваются объекты в ходе выполнения сценария.

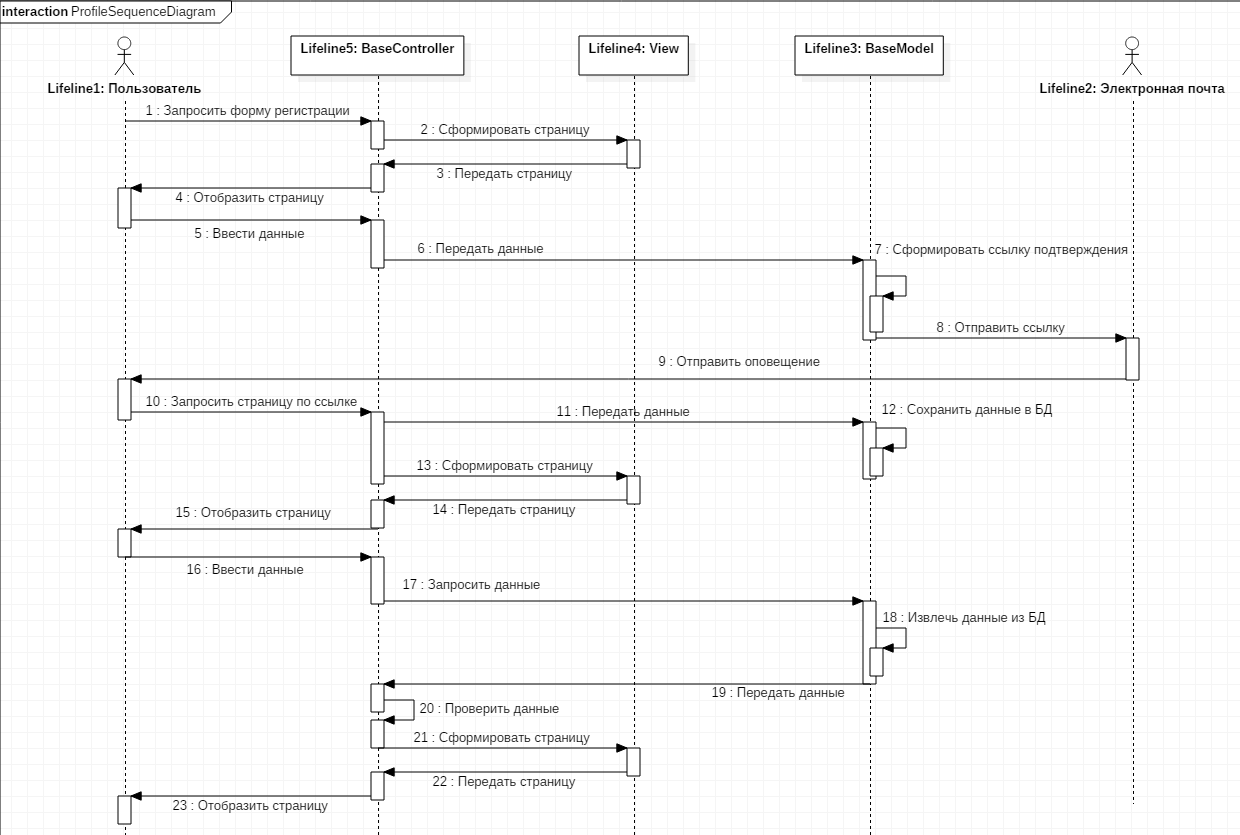


Рисунок 13 – Диаграмма последовательности основного сценария работы с профилем

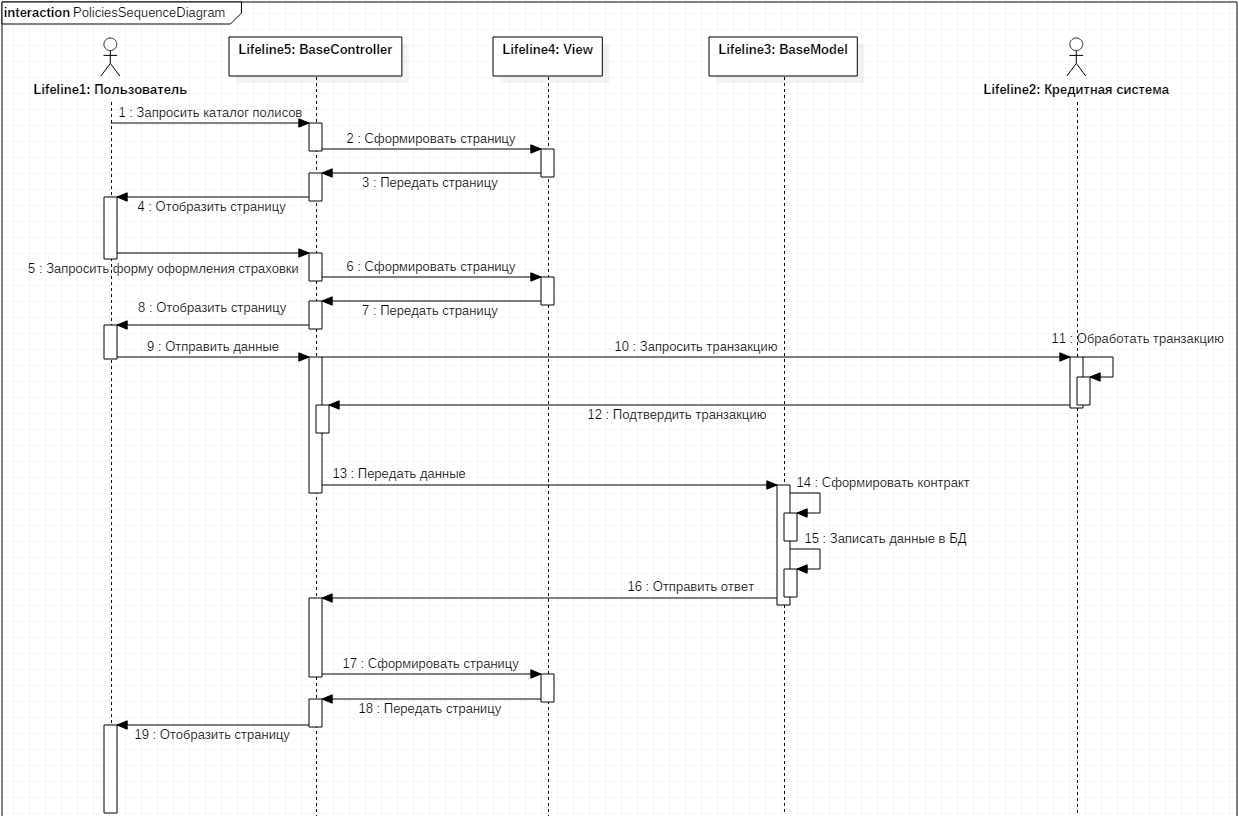


Рисунок 14 – Диаграмма последовательности основного сценария работы с полисами

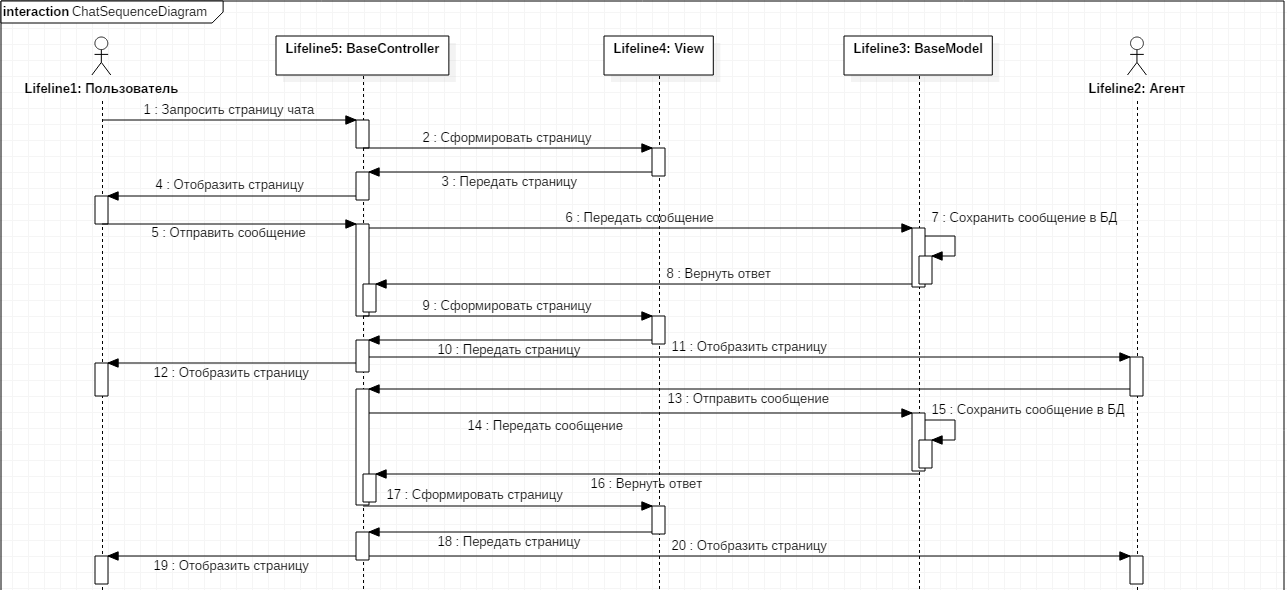


Рисунок 15 – Диаграмма последовательности основного сценария чата

Диаграмма последовательностей – одна из двух диаграмм взаимодействия. Вторая – диаграмма коопераций.

Диаграмма кооперации – это альтернативный способ изображения сценария варианта использования. Этот тип диаграмм заостряет внимание на связях между объектами, отображая обмен данными в системе. А диаграммы последовательности отображают взаимодействие объектов во времени, поэтому ее следует читать сверху вниз и слева направо.

Диаграммы кооперации содержат все те же элементы, что и диаграммы последовательности: объекты, действующие лица, связи между ними и сообщения, которыми они обмениваются, но они уже не упорядочены во времени.

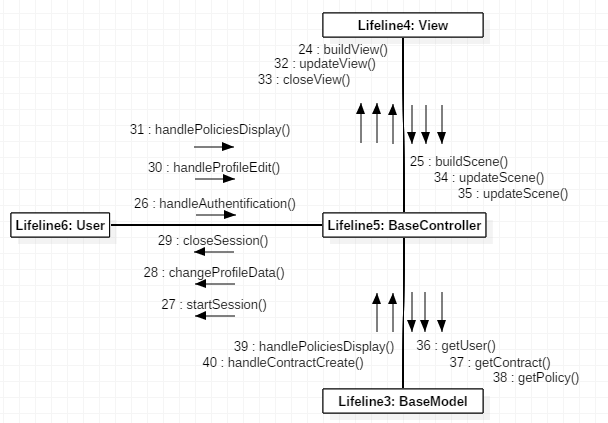


Рисунок 16 – Общая диаграмма коопераций

Финальным этапом проектирования системы является разработка диаграммы развертывания. Диаграмма развертывания в UML моделирует физическое развертывание артефактов на узлах.Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI).

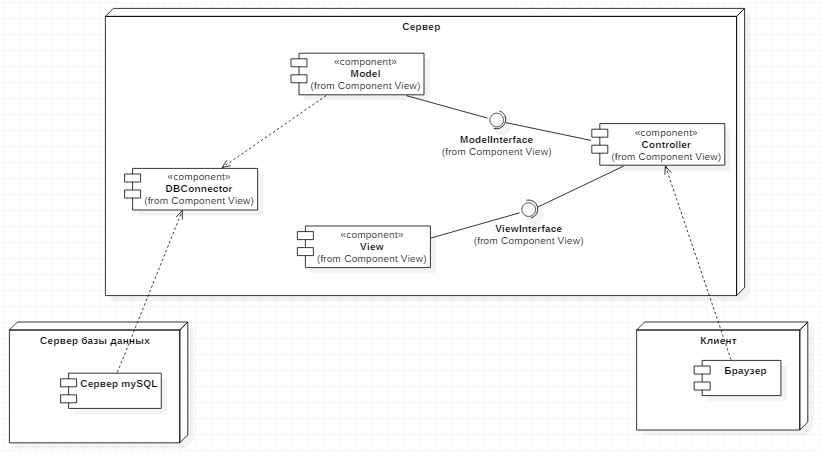


Рисунок 17 – Диаграмма развертывания

Из диаграммы развертывания видно, что для функционирования системы пользователю необходим лишь персональный компьютер и установленный веб-браузер для отправки запросов на сервер. Сервер в свою очередь перенаправляет сообщение на контроллер приложения, который обрабатывает его и в соответствии с полученными инструкциями выполняет работу над Моделью, результаты которой передаются пользователю в виде обновления представления. Основная часть ПО разворачивается серверно в самой страховой компании, где оно будет устанавливаться. Сервер базы данных может находиться как локально, так и отдельно от остальной системы, так как связь с ним поддерживается с помощью JDBC.

**Выводы:** в ходе данной расчетно-графической работы были закреплены и усовершенствованы навыки проектирования ПО с помощью нотации UML. Был разработан начальный концепт веб-приложения для работы страховой компании, составлены диаграммы прецедентов, активности, классов, последовательностей, взаимодействия и развертывания для описания возможностей, структуры, функционала и требований будущей программы. В ходе разработки был использован паттерн MVC, который в перспективе сделает программу легко расширяемой и к тому же поддерживается большинством языков программирования. Данный опыт важен в будущей карьере разработчика, так как тщательное проектирование системы помогает избежать ошибок на этапе реализации, которые могут привести к значительным денежным и временным потерям.