Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет: инженерно – экономический

Кафедра: экономики

Дисциплина: Распределенные системы обработки информации

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЕЯТЕЛЬНОСТИ БАНКА НА ПРИМЕРЕ «ЗАКРЫТОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО «АКЦИОНЕРНЫЙ БАНК РЕКОНВЕРСИИ И РАЗВИТИЯ»

(ЗАО «РРБ-Банк»)»

Студент: гр. 773901 Сидоров В. А.

Руководитель:

Минск 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc7857473)

[1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc7857474)

[1.1 Сущность, основные понятия и элементы банка и банковской системы 4](#_Toc7857475)

[1.2 Банковская система в Республике Беларусь 7](#_Toc7857476)

[1.3 Депозитные вклады 8](#_Toc7857477)

[1.4 Кредитование 9](#_Toc7857478)

[1.5 Рейтинг банков Республики Беларусь за 2018 год по размеру чистой прибыли и рентабельности 11](#_Toc7857479)

[1.6 Краткая характеристика ЗАО «РРБ-Банк» 14](#_Toc7857480)

[1.7 Описание решения проблемы планирования бюджета 16](#_Toc7857481)

[2 Описание основного процесса предметной области 20](#_Toc7857482)

[3 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ 27](#_Toc7857483)

[4 СПЕЦИФИКАЦИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ 29](#_Toc7857484)

[5 ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩУЮ БИЗНЕС-ЛОГИКУ CЕРВЕРНОЙ ЧАСТИ 30](#_Toc7857485)

[5.1 Использование системы контроля версий git 33](#_Toc7857486)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 36](#_Toc7857487)

[ЗАКЛЮЧЕНИЕ 46](#_Toc7857488)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc7857489)

[ПРИЛОЖЕНИЯ 48](#_Toc7857490)

[Приложение А (логическая информационная модель) 48](#_Toc7857491)

[Приложение Б (физическая информационная модель) 48](#_Toc7857492)

[Приложение В (диаграмма компонентов) 49](#_Toc7857493)

[Приложение Г (диаграмма классов серверной части) 50](#_Toc7857494)

[Приложение Д (диаграмма классов клиентской части) 51](#_Toc7857495)

[Приложение Е (диаграмма развертывания) 51](#_Toc7857496)

[Приложение Ж (диаграмма последовательности) 52](#_Toc7857497)

[Приложение З (листинг генерации базы данных) 53](#_Toc7857498)

[Приложение И (листинг) 56](#_Toc7857499)

ВВЕДЕНИЕ

Развитие компьютерной индустрии, в частности сферы разработки программного обеспечения, привело к автоматизации многих процессов нашей повседневной жизни и деятельности. Значительную роль в современном обществе играют разнообразные системы анализа и хранения данных, благодаря которым стала возможной быстрая обработка информации и централизация ее хранения, что обеспечило экономию затрат и времени на получение необходимых данных, а также их обработку.

Использование компьютера в банковских операциях позволяет расширить применение экономико-математических методов в управлении, оптимизировать некоторые процессы (например, распределение и размещение мобилизованных средств). При этом повышается оперативность проведения расчетов и, следовательно, оперативность принимаемых решений. Появляется возможность расширения спектра оказываемых услуг, повышения их качества и расширения географии за счет более полного использования средств телекоммуникаций.

Целью курсовой работы является разработка информационной системы деятельности банка на примере существующего коммерческого банка ЗАО «РРБ-Банк». Для достижения данной цели необходимо решить следующие задачи курсового проекта:

1. Выполнить анализ предметной области.
2. Поставить техническую задачу и определить требования к банковским информационным системам.
3. Построить информационную и функциональную модели информационной системы банка на примере ЗАО «РРБ-Банк».
4. Разработать программное обеспечение решаемой задачи.

Во время анализа предметной области формируется характеристика банковской сферы, базовые аспекты функционирования в мире и на территории РБ, проводится анализ конкурентной среды. Для выполнения второй задачи необходимо сформировать требования к программному обеспечению в соответствии с предметной областью: поставить задачу, определить стек технологий, основные функциональные модули. Во время построения информационной и функциональной моделей необходимо осуществить декомпозицию на примере коммерческого банка ЗАО «РРБ-Банк». В результате необходимо разработать программное обеспечение решаемой задачи, подготовить руководство пользователя и сделать выводы.

1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Сущность, основные понятия и элементы банка и банковской системы

Слово «банк» происходит от итальянского banco и означает «стол». Предшественниками банков были средневековые менялы — представители денежно-торгового капитала. Они принимали денежные вклады у купцов и специализировались на обмене денег разных городов и стран. Со временем менялы стали использовать эти вклады, а также собственные денежные средства для выдачи ссуд и получения процентов, что означало превращение менял в банкиров.

В XVI-XVII вв. купеческие гильдии ряда городов (Венеции, Генуи, Милана, Амстердама, Гамбурга, Нюрнберга) создали специальные банки для проведения безналичных расчетов между своими клиентами-купцами. Банки вели расчеты с клиентами в специальных денежных единицах, выраженных в определенных весовых количествах благородных металлов. Свободные денежные средства жиробанки предоставляли в ссуду государству, городам и привилегированным внешнеторговым компаниям [1].

Банк — денежно-кредитный институт, регулирующий денежный оборот в наличной и безналичной формах. В условиях рынка банки являются ключевым звеном, питающим народное хозяйство дополнительными денежными ресурсами. Современные банки не только торгуют деньгами, одновременно они являются аналитиками рынка. По своему местоположению банки оказываются ближе всего к бизнесу, его потребностям, меняющейся конъюнктуре.

Таким образом, рынок неизбежно выдвигает банк в число основополагающих, ключевых элементов экономического регулирования.

Банковская система – это совокупное функционирование банков и кредитных учреждений в качестве единого денежно-кредитного механизма. На ее развитие влияют законодательство, политика государства и межбанковская конкуренция. Сдерживающие факторы — значительный налоговый пресс на прибыль банков, нехватка ресурсов для успешного обслуживания операций, недостаток квалифицированных кадров [1].

Большинство стран придерживаются двухуровневой банковской системы. Ее смысл состоит в том, что один банк контролирует все иные банки (второго уровня), работающие в государстве. Центральный банк может быть государственным (таковы Банк Англии, Немецкий Федеральный банк), акционерным (Федеральная резервная система США), смешанной формы собственности (Банк Японии — государству принадлежит 55%). Для определения предметной области необходимо оперировать такими понятиями, как бюджет, финансовое планирование.

Под банковской системой любой страны понимается совокупность различных видов национальных банковских институтов, банков и небанковских кредитных организаций, действующих в рамках общего денежно-кредитного механизма и единого законодательного поля.

Традиционными элементами банковской системы являются кредитные организации, банковская инфраструктура и банковское законодательство.

Кредитные организации:

* центральный банк;
* коммерческие банки;
* небанковские кредитные организации.

С экономической точки зрения центральный банк выполняет регулятивную, контролирующую и исследовательскую функцию в государстве. Центральный банк контролирует и регулирует как денежные потоки в стране, так и административно-правовые отношения с коммерческими банками. Основной особенностью центральных банков является то, что в отличии от коммерческих банков, они не вступают в контакт с конечными потребителями. Таким образом можно сделать вывод, что центральный банк выполняет обязанность эмиссионного центра во всей денежно-кредитной системе государства.

Если же говорить про коммерческий банк, то его основные функции это:

* аккумуляция временно свободных денежных средств;
* размещение привлеченных денежных средств;
* расчетно-кассовое обслуживание с физическими и юридическими клиентами.

Исходя из вышесказанного можно сделать вывод, что коммерческий банк – это денежно-кредитное предприятие, находящееся на нижнем уровне двухуровневой банковской схемы государства и занимающееся оказанием на платной (коммерческой) основе банковских услуг физическим и юридическим лицам в условиях рыночной конкуренции с другими коммерческими банками.

Небанковская кредитная организация – это денежно-кредитное предприятие, которое имеет право осуществлять ограниченный спектр услуг в рамках своей деятельности. Основными функциями небанковских кредитных организаций являются:

* открытие и ведение банковских счетов юридических лиц;
* осуществление расчетов по поручению юрлиц, в т. ч. банков-корреспондентов, по их банковским счетам;
* инкассация денежных средств, векселей, платежных и расчетных документов и кассовое обслуживание юридических лиц;
* купля-продажа иностранной валюты в наличной и безналичной форме;
* осуществление деятельности на рынке ценных бумаг.

Небанковская кредитная организация не имеет права привлекать вклады и выдавать кредиты, она обеспечивает систему расчетов и переводов.

Банковская инфраструктура — организации, обеспечивающие эффективное функционирование банковской системы:

* информационное обеспечение (кредитные бюро, консалтинговые службы);
* кадровое обеспечение (учебные заведения, бюро занятости);
* коммуникационное обеспечение (системы связи, провайдеры услуг).

В современном обществе банки выполняют самые разнообразные операции. Они не только организуют денежный оборот и кредитные отношении, через них осуществляются финансирование народного хозяйства, страховые операции, купля-продажа ценных бумаг, а в некоторых случаях посреднические сделки и управление имуществом. Кредитные учреждения выступают в качестве консультантов, участвуют в обсуждении народно-хозяйственных программ, ведут статистику, имеют свои подсобные предприятия [2].

В период централизации управления в государственном аппарате кредитование осуществлялось на базе нормативного, директивного распределения ресурсов сверху, безналичные расчеты между предприятиями носили директивный характер, принцип «один банк для клиента» не давал основы для развития коммерческих отношений в банковской сфере. С переходом к рыночной экономике положение банка существенно изменилось и трактовка его сути как аппарата управления (или части государственного аппарата управления) сама собой отпала.

Еще одна характеристика – посредническая. Банк выступает в качестве кредитора, заемщика, посредника, но одновременно с этим не является ни тем, ни другим, ни третьим, так как это всего лишь его функциональные обязанности. Любое предприятие может выполнять вышеперечисленные функции, но именно банк выполняет данные операции в денежном расчете.

Банки являются участниками биржи. Они самостоятельно выполняют и организовывают биржевые операции, торгуют ценными бумагами. Исторически и логически сложилось так, что банк не является частью биржевой организации. Частные банки появились задолго до создания бирж и проводили операции купли-продаж ценных бумаг.

1.2 Банковская система в Республике Беларусь

Днем рождения первой банковской системы Беларуси можно считать 8 января 1870 года, когда в Гомеле учредили общественный банк. В 1917 году был учрежден Народный банк РСФСР, который был реорганизован в 1923 году в государственный банк СССР. С 1987 года начался переход к новой денежной системе, а непосредственно 1 апреля 1991 года был сформирован Национальный банк Республики Беларусь.

Так же, как и в странах бывшего Советского Союза, в Республике Беларусь существует двухуровневая банковская система: центральный банк (Национальный банк РБ) и коммерческие банки. Вне этой структуры функционируют инвестиционные, страховые, трастовые, благотворительные фонды и компании, и, что примечательно, ломбарды.

Основной функцией Национального банка Республики Беларусь (НБ РБ) является поддержка официальной экономической политики государства. НБ РБ является государственным учреждением и подотчетен напрямую Президенту Республики Беларусь (периодичность отчетов — каждые 3 месяца).

Право эмиссии денежных знаков и ценных бумаг закреплено за Национальным банком, что позволяет быть ему во главе иерархии банковской системы Республики Беларусь

НБ РБ лицензирует банковскую деятельность, определяет требования к банкам (например, минимальный размер уставного фонда и капитала), ведет надзор за их деятельностью. Также обеспечивает нормальное проведение межбанковских расчетов, выполняет инкассацию, доставляет в банки наличные денежные средства. Еще одна функция — создание и выпуск памятных монет.

Активами любого Национального банка являются золотовалютные резервы, кредиты, выданные Правительству и коммерческим банкам и государственные облигации. По состоянию на 1 ноября 2018 года золотовалютные резервы Республики Беларусь составили 7,395 млрд долларов США, а по состоянию на 1 января 2018 года они составляли около 4,897 млрд долларов США.

Цель коммерческих банков второго уровня — получение прибыли в виде процентов от сделок. Еще один источник дохода появился вместе с пластиковыми картами: банки получают финансовые средства от клиентов за выпуск некоторых видов платежных карт и их обслуживание, за интернет- и мобильный банкинг, SMS-оповещения об операциях, выписки по картам, с нерезидентов — за пользование банкоматом.

Банки второго уровня обменивают валюту; ведут кассовое обслуживание; выполняют операции с ценными бумагами (в том числе выпускают и продают облигации); покупают, продают и обменивают драгоценные металлы, камни.

Капитал банковской системы Беларуси в большей степени контролируется государством. Роль государственных банков можно назвать определяющей для экономики страны.

1.3 Депозитные вклады

Выбирать депозит всегда необходимо исходя из вашей цели. Она и обусловит вид необходимого Вам [депозита](https://benefit.by/page/show/articles/20) или перечень характеристик, которыми он должен обладать. А чтобы Вы имели представление, какие виды депозитов вообще бывают, проведем ознакомительный экскурс.

Депозиты в РБ делятся на три основные категории: до востребования, срочные и условные.

Под условным понимается депозит, по которому сумма вклада с начисленными процентами возвращается вкладчику при наступлении/ненаступлении определенного в заключенном договоре обстоятельства (события). Пока применяется для частных лиц достаточно редко.

По вкладам до востребования конкретный срок хранения не устанавливается. Это самый простой и удобный вклад. На него можно вносить, в том числе пополнять, и с него можно снимать любые суммы и в любое удобное для вас время. На таких вкладах крайне низкая процентная ставка.

Срочные депозиты открываются на определенный срок, в течение которого клиент не имеет права забрать деньги. Чем больше срок, тем выше доходность депозита. Доходность также зависит от суммы (чем больше сумма, тем больше ставка) и от частоты начисления процентов (чем чаще, тем меньше доходность). В случае, если Вы досрочно забираете средства со срочного депозита, проценты по вкладам обычно начисляются по ставке «до востребования», т.е. минимальной.

Если же вы забыли вовремя забрать деньги, то банк переводит ваши деньги на счёт до востребования или продлевает (пролонгирует) действие договора. Процентный доход за основной период начисляется в полном объеме согласно договору.

Срочные вклады делят на сберегательные и накопительные. Главное их отличие – в возможности или невозможности пополнения.

Если Вам не принципиально получать дополнительный доход ежемесячно, можете открыть сберегательный депозит. Как правило, по нему внесение денег в период действия договора не предусмотрено. Однако он – очень удобный способ сберечь свои накопления, не растратив их на мелочи.

Накопительные депозиты – это вклады с возможностью увеличения первоначальной суммы. Основная задача — накопить сумму для крупной покупки. Их главное преимущество в том, что, открыв такой депозит, можно реально увеличить итоговую сумму, причем проценты по вкладам будут начисляться на все увеличивающуюся сумму.

Процентные ставки по накопительным депозитам немного ниже, чем по сберегательным и доходным, так как банк заранее не знает, сколько же денег окажется на счете в определенный момент времени. Но если у клиента нет на руках большой суммы, но он может делать небольшие ежемесячные взносы, то размер ставки отходит на второй план.

Стоит отметить, что доход по депозитам в РБ начисляется в виде процентов, потому иногда банковские депозиты называют "процентные депозиты".

В Беларуси ставка процентов на сумму пополнения вклада, как правило, не отличается от основной. Однако в депозитном договоре банк может прописать различные ограничения, которые могут помешать вкладчику максимально эффективно пользоваться накопительным счетом.

Существуют также пополняемые депозиты с возможностью частичного изъятия средств без потери процентов. Такие счета можно пополнять, однако размер минимального взноса и сроки пополнения чётко оговариваются в договоре.

1.4 Кредитование

Банковский кредит – это предоставление банком денег в долг на условиях срочности, платности, возвратности.

* Срочность означает, что деньги всегда выдаются на строго определенный срок.
* Платность означает, что банк никогда не дает деньги безвозмездно.
* Возвратность означает, что банк всегда ожидает возврата выданных денег.

В Беларуси доступны все основные виды кредитов для частных лиц: потребительский, автокредит и кредит на недвижимость.

Кредит – ссуда в денежной или товарной форме, предоставляемая заемщику на условиях возвратности, срочности и платности.

Кредит имеет многообразные формы проявления: коммерческий, банковский, государственный и международный.

Банковская форма кредита (банковский кредит) - наиболее распространенная. Кредит предоставляется банками в денежной форме из собственного или привлеченного капитала и осуществляется в форме выдачи ссуд, учета векселей и др. По объему ссуда при банковской форме кредита значительно больше ссуд, выдаваемых при каждой из других его форм.

Срок кредита – это время, на которое выдается кредит. Выделяют полный срок кредита, срок погашения кредита, его использования и др.

Сумма кредита – размер денежных средств в соответствующей валюте, предоставляемых в кредит, определяемый потребностью и возможностью их получения клиентом.

Процентная ставка – установленная плата за пользованием кредитом, выраженная в процентах к размеру кредита за определенный период (год, месяц и т.д.).

Способ погашения кредита. Выделяется 2 основных способа погашения банковского кредита.

* Аннуитет – периодическая уплата относительно фиксированной суммы, включающая основной долг и проценты, в погашение кредита, регламентируемая условиями кредитного договора.  
  Дифференцированный платеж – способ погашения кредита, при котором размер оплаты исчисляется непосредственно от суммы задолженности по кредиту за определенный период времени.
* Имеет место также периодическое погашение фиксированной доли основного долга и начисленных процентов за определенный период. Например, «ежемесячно не менее 2% основного долга и сумму начисленных за период процентов за пользование кредитом». Такой способ часто используется для кредитных карточек.
* Овердрафт (Овердрафтный кредит). Сумма свыше собственных средств на карточке, которую держатель карточки может неоднократно расходовать с последующим ее возмещением и уплатой начисленных процентов в оговоренный срок. В настоящее время овердрафтный кредит уже не редкость. Овердрафтные кредиты часто предоставляются держателям зарплатных карт в соответствующем банке.
* Досрочное погашение кредита. Погашение всей либо части суммы кредита ранее срока погашения, оговоренного в кредитном договоре.

Обеспечение по кредиту. Дополнительная гарантия погашения клиентом кредита, предоставляемая банку.

Основными видами обеспечения по банковскому кредиту являются: поручительство, залог, перевод правового титула, гарантийный депозит.

1.5 Рейтинг банков Республики Беларусь за 2018 год по размеру чистой прибыли и рентабельности

Чистая прибыль – это разница между операционными доходами и операционными расходами. Операционные доходы включают в себя доходы от операций с ценными бумагами, доходы от участия в капитале других организаций, положительную переоценку ценных бумаг, средств в иностранной валюте, драгоценных металлов, комиссионные вознаграждения, доходы от сдачи имущества в аренду, от передачи активов в доверительное управление, от восстановления резервов на возможные потери по ссудам и некоторые другие доходы банка. Операционные расходы учитывают расходы по операциям с приобретенными ценными бумагами, отрицательную переоценку ценных бумаг, средств в иностранной валюте, драгоценных металлов, комиссионные сборы, расходы от передачи активов в доверительное управление, отчисления в резервы на возможные потери, расходы на содержание персонала, амортизацию имущества, организационные и управленческие расходы (в частности, на рекламу, охрану, услуги связи, служебные командировки, аудит, подготовку кадров).

По данным Интернет-ресурса «Infobank.by» рейтинг банков по величине чистой прибыли по итогам предыдущего года (в сравнении с ее размером по результатам 2017-го года) выглядит следующим образом.

Таблица 1.1 – Рейтинг банков по величине чистой прибыли за 2018 год (примечание: источник infobank.by)

|  |  |  |  |
| --- | --- | --- | --- |
| **Банк** | **Размер чистой прибыли, тыс. руб.** | | **Изменения размера чистой прибыли, тыс. руб.** |
| На 01.01.19 | На 01.01.18 |
| [Приорбанк](http://infobank.by/informaciya-o-banke/priorbank-oao/) | 185 254 | 109 019 | 76 235 |
| [Беларусбанк](http://infobank.by/informaciya-o-banke/oao-asb-belarusbank/) | 172 281 | 154 796 | 17 485 |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| [Белгазпромбанк](http://infobank.by/informaciya-o-banke/belgazprombank-oao/) | 104 260 | 108 556 | -4 296 |
| [БПС-Сбербанк](http://infobank.by/informaciya-o-banke/bps-sberbank-oao/) | 85 245 | 2 278 | 82 967 |
| [Банк БелВЭБ](http://infobank.by/informaciya-o-banke/bank-belveb-oao/) | 68 371 | 64 296 | 4 075 |
| [Альфа–Банк](http://infobank.by/informaciya-o-banke/aljfa-bank-zao/) | 56 592 | 35 949 | 20 643 |
| [МТБанк](http://infobank.by/informaciya-o-banke/mtbank-zao/) | 42 774 | 37 573 | 5 201 |
| [Банк ВТБ (Беларусь)](http://infobank.by/informaciya-o-banke/bank-vtb-belarusj-zao/) | 27 586 | 16 751 | 10 835 |
| [Белагропромбанк](http://infobank.by/informaciya-o-banke/belagroprombank-oao/) | 21 812 | 206 483 | -184 671 |
| [ТК Банк](http://infobank.by/informaciya-o-banke/tk-bank-zao/) | 18 767 | 48 068 | -29 301 |
| [Банк Москва–Минск](http://infobank.by/informaciya-o-banke/bank-moskva-minsk-oao/) | 14 296 | 4 646 | 9 650 |
| [Идея Банк](http://infobank.by/informaciya-o-banke/ideya-bank-zao/) | 13 457 | 19 964 | -6 507 |
| [Белинвестбанк](http://infobank.by/informaciya-o-banke/belinvestbank-oao/) | 10 205 | 806 | 9 399 |
| [БНБ–Банк](http://infobank.by/informaciya-o-banke/belorusskij-narodnyj-bank-oao/) | 10 039 | 3 154 | 6 885 |
| [Паритетбанк](http://infobank.by/informaciya-o-banke/paritetbank-oao/) | 8 664 | 8 492 | 172 |
| [БСБ Банк](http://infobank.by/informaciya-o-banke/bsb-bank-zao/) | 6 953 | 12 109 | -5 156 |
| [Франсабанк](http://infobank.by/informaciya-o-banke/fransabank-oao/) | 3 022 | 4 261 | -1 239 |
| [СтатусБанк](https://infobank.by/informaciya-o-banke/statusbank-oao/) | 2 832 | 6 028 | -3 196 |
| [БТА Банк](http://infobank.by/informaciya-o-banke/bta-bank-zao/) | 2 040 | 5 358 | -3 318 |
| [АБСОЛЮТБАНК](http://infobank.by/informaciya-o-banke/absolyutbank-zao/) | 1 847 | 3 144 | -1 297 |
| [РРБ–Банк](http://infobank.by/informaciya-o-banke/rrb-bank-zao/) | 1 641 | 3 855 | -2 214 |
| [Технобанк](http://infobank.by/informaciya-o-banke/texnobank-oao/) | 1 548 | 3 644 | -2 096 |
| [Цептер Банк](http://infobank.by/informaciya-o-banke/cepter-bank-zao/) | 213 | 1 696 | -1 483 |
| [Банк Решение](http://infobank.by/informaciya-o-banke/bank-reshenie-zao/) | 198 | -5 653 | 5 851 |
| **Всего** | **859 897** | **855 273** | **4 624** |

На первом месте расположился Приорбанк, Беларусбанк — занял вторую строчку. С третьим результатом закончил год Белгазпромбанк, который, тем не менее, получил прибыль меньшую, чем в 2017-м году.

На основании данных, опубликованных Рейтинговым агентством Белорусского государственного университета, «Infobank.by», опубликовал данные о рентабельности банков по итогам 2017-го года.

Никто не станет отрицать, что целью банка является получение прибыли. За 2018 год общая прибыль банков Беларуси достигла 886,7 млн рублей, увеличившись, по сравнению с 2017-м годом, на 1,8 млн рублей.

Прибыль банков официально еще не утверждена, однако, как отмечают аналитики Рейтингового агентства БГУ, небольшие ее изменения вряд ли повлияют на относительные показатели рентабельности.

Средняя рентабельность активов (ROA) белорусских банков составила 1,4 против 1,3 по итогам предыдущего года

Рентабельность активов (return on assets, ROA) — это финансовый коэффициент, исчисляемый как отношение чистой прибыли банка к величине его активов. Он характеризует банк эффективностью использования активов, показывает способность банков генерировать прибыль, качество управления активами.

В разрезе банков ситуация выглядит следующим образом.



Рисунок 1.1 – Рентабельность активов банков за 2018 год (примечание: источник infobank.by)

Наилучшие показатели эффективности работы среди крупных банков — у Приорбанка и Белгазпромбанка. Также неплохо по итогам 2018-го года отработали «средние и небольшие по величине» банки: ТК Банк, МТБанк, Идея Банк и Альфа-Банк.

Полученная банками прибыль обеспечила среднюю рентабельность капитала (ROE) в 9,6%, что на 1,2% меньше, чем в 2017 году.

Рентабельность капитала — отношение прибыли банка за период к среднему размеру его капитала за тот же период

По этому показателю первую строчку рейтинга, как и в прошлом году, занимает «средний по величине» МТБанк. На втором месте — крупный Приорбанк.



Рисунок 1.2 – Рентабельность капитала банков за 2018 год (примечание: источник infobank.by)

Пятерку аутсайдеров как по рентабельности активов, так и по рентабельности капитала сформировали: Технобанк, Белинвестбанк, Белагропромбанк, Банк Решение и Цептер Банк.

1.6 Краткая характеристика ЗАО «РРБ-Банк»

Закрытое акционерное общество «Акционерный банк реконверсии и развития» (ЗАО «РРБ-Банк») было создано 22 февраля 1994 года и является частным банком (регистрационный номер 37 от 22.02.1994). В 1995—1996 годах акционерами ЗАО «РРБ-Банк» и ЗАО «Белмедбанк» были достигнуты договоренности по вопросу объединения банков. Национальный банк Республики Беларусь 30 сентября 1996 года зарегистрировал изменения в учредительных документах ЗАО «РРБ-Банк» в связи с присоединением к нему ЗАО «Белмедбанк».

Лицензия на осуществление банковской деятельности № 21 выдана 7 июня 2013 года Национальным банком Республики Беларусь. Перечень операций, которые Банк вправе осуществлять:

* привлечение денежных средств физических и юридических лиц во вклады (депозиты);
* размещение привлеченных денежных средств от своего имени и за свой счет на условиях возвратности, платности и срочности;
* открытие и ведение банковских счетов физических и юридических лиц;
* осуществление расчетного и кассового обслуживания физических и юридических лиц, в том числе банков-корреспондентов;
* валютно-обменные операции;
* выдача банковских гарантий;
* выпуск в обращение банковских платёжных карточек;
* выдача ценных бумаг, подтверждающих привлечение денежных средств во вклады (депозиты) и размещение их на счета;
* финансирование под уступку денежного требования (факторинг);
* предоставление физическим и юридическим лицам специальных помещений или находящихся в них сейфов для банковского хранения документов и ценностей (денежных средств, ценных бумаг, драгоценных металлов и драгоценных камней и др.).

Наиболее крупные акционеры (с долей более 5% уставного фонда Банка):

* ООО «Дикрис» — 79,94%
* ООО «Спидстер-М» — 6,86%
* Цыбулин В.А. — 10,06%

РРБ-Банк – член Ассоциации белорусских банков, Сообщества всемирных межбанковских финансовых коммуникаций (S.W.I.F.T.), международной платежной системы VISA International и национальной системы БЕЛКАРТ. Включен в реестр банков и небанковских кредитно-финансовых организаций, признанных таможенными органами гарантами уплаты таможенных платежей [3].

РРБ-Банк является членом РКП (Республиканской конфедерации предпринимательства).

С 2005 года Банк проводит аудит в соответствии с международными стандартами финансовой отчетности.

Основные метрики банка ЗАО «РРБ-Банк»:

Таблица 1.2 – Основные метрики банка ЗАО «РРБ-Банк»  
(прим. источник ЗАО «РРБ-Банк»)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Наименование показателя** | **Норматив НБРБ** | **Значение на** **01.01.16** | **Значение на 01.01.17** | **Значение на 01.12.17** | **Значение на 01.01.18** | **Значение  на 01.04.19** |
| Нормативный капитал банка (млн.руб.) | - | - | 464 496.4 | 49,9152 | 50,555 | 55,9996 |
| Достаточность нормативного капитала, % | 10 | 26,5 | 38,4 | 45,971 | 43,848 | 36,386 |
| Достаточность основного капитала I уровня, % | 5 | 13,9 | 19,1 | 19,303 | 18,404 | 18,046 |
| Достаточность капитала I уровня, % | - | - | - | 22,281 | 21,176 | 20,048 |
| Краткосрочная ликвидность | 1 | 1,3 | 1,0 | 1,6 | 1,6 | - |
| Мгновенная ликвидность, % | 20 | 1 054,1 | 288,5 | 122,6 | 266,9 | - |
| Текущая ликвидность, % | 70 | 181,7 | 97,8 | 418,1 | 127,7 | - |
| Соотношение ликвидных и суммарных активов, % | 20 | 40,3 | 53,1 | 55 | 52,6 | - |
| Норматив покрытия ликвидности | 100 | - | - | - | - | 185,8 |
| Норматив чистого стабильного фондирования | 100 | - | - | - | - | 111,1 |

1. Описание основного процесса предметной области

В последнее время разработка программного обеспечения осуществляется по CASE-технологии с применением SADT (Structure Analysis and Design Technique – Методология структурного анализа и проектирования). Это позволяет создавать единое информационное пространство на уровне менеджеров, обеспечить функциональную обозримость систем управления, осуществлять развитие самой системы с наименьшими затратами, производить актуализацию проектной документации в электронном виде.

Международный стандарт SADT поддерживается специальным языком IDEF (ICAM DEFinition Methodology – Методология определения ICAM Integrated Computer-Aided Manufacturing – Интегрированная программа компьютеризации промышленности) для описания проектов информационно-управляющих систем.

Поскольку IDEF0 есть методология функционального моделирования, имя блока, описывающее функцию, должно быть глаголом или глагольным оборотом; например, имя блока "Выполнить проверку", означает, что блок с

таким именем превращает непроверенные детали в проверенные. После присваивания блоку имени, к соответствующим его сторонам присоединяются

входные, выходные и управляющие стрелки, а также стрелки механизма, что

и определяет наглядность и выразительность изображения блока IDEF0.

Чтобы гарантировать точность модели, следует использовать стандартную терминологию. Блоки именуются глаголами или глагольными оборотами и эти имена сохраняются при декомпозиции Стрелки и их сегменты, как отдельные, так и связанные в «пучок», помечаются существительными или оборотами существительного. Метки сегментов позволяют конкретизировать данные или материальные объекты, передаваемые этими сегментами, с соблюдением синтаксиса ветвлений и слияний.

Каждая сторона функционального блока имеет стандартное значение с точки зрения связи блок/стрелки. В свою очередь, сторона блока, к которой присоединена стрелка, однозначно определяет ее роль. Стрелки, входящие в левую сторону блока - входы. Входы преобразуются или расходуются функцией, чтобы создать то, что появится на ее выходе. Стрелки, входящие в блок сверху - управления. Управления определяют условия, необходимые функции, чтобы произвести правильный выход. Стрелки, покидающие блок справа – выходы, т.е. данные или материальные объекты, произведенные функцией. Стрелки, подключенные к нижней стороне блока, представляют механизмы. Стрелки, направленные вверх, идентифицируют средства, поддерживающие выполнение функции. Другие средства могут наследоваться из родительского блока. Стрелки механизма, направленные вниз, являются стрелками вызова. Стрелки вызова обозначают обращение из данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь, т.е. разные модели или разные части одной и той же модели могут совместно использовать один и тот же элемент (блок). Стандартное расположение стрелок показано на рисунке 3.1.

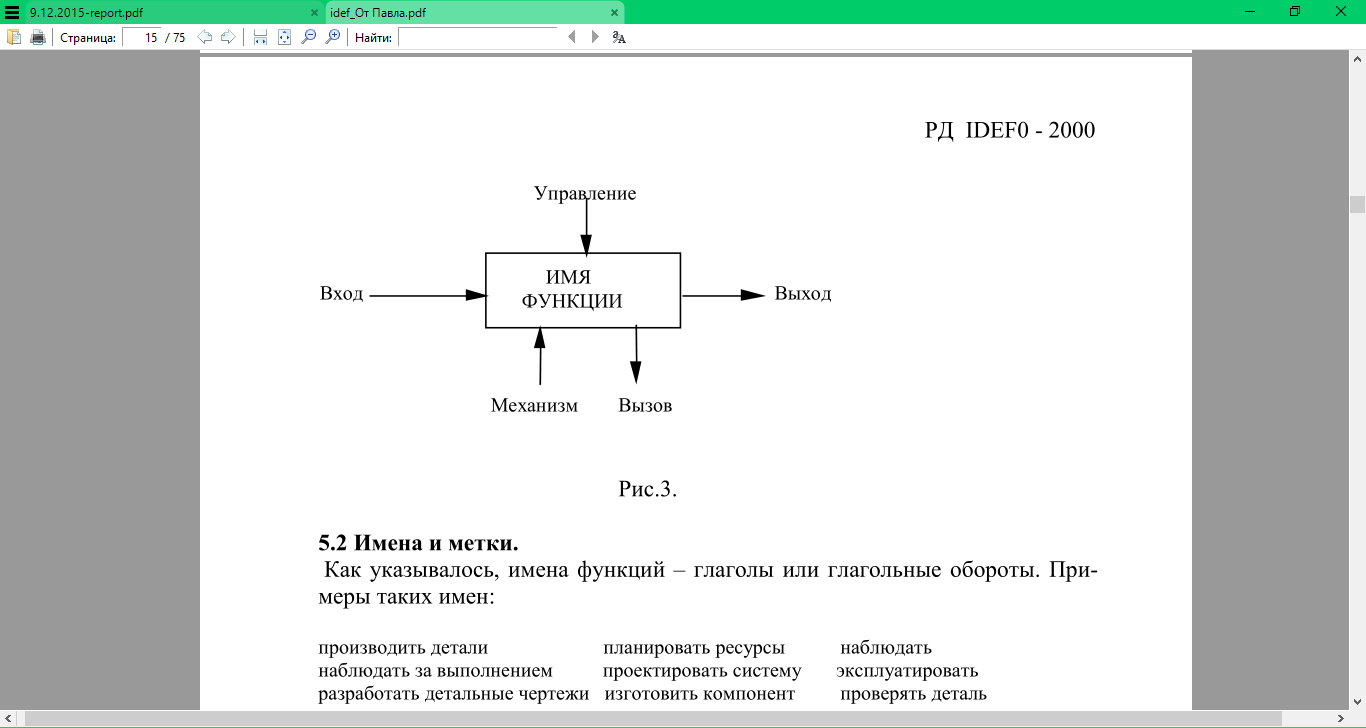


Рисунок 3.1. Теория. Функциональный блок.

Рисунок 3.2 отражает основной процесс «Оказать услугу клиенту банка». На вход функционального блока подаются сам клиент и его документы. Управлением являются «Нормативно-правовые документы», так как работа с денежными потоками осуществляется на основании этого документа, «Финансовые показатели банка», так как это необходимое условие для повышения виральности клиента, а также «Правила оформления услуг» для корректного осуществления процесса. Выходом данной модели будем считать оказание услуги или отказ об оказании услуги. Механизмом будем считать кредитного работника и консультанта банка.

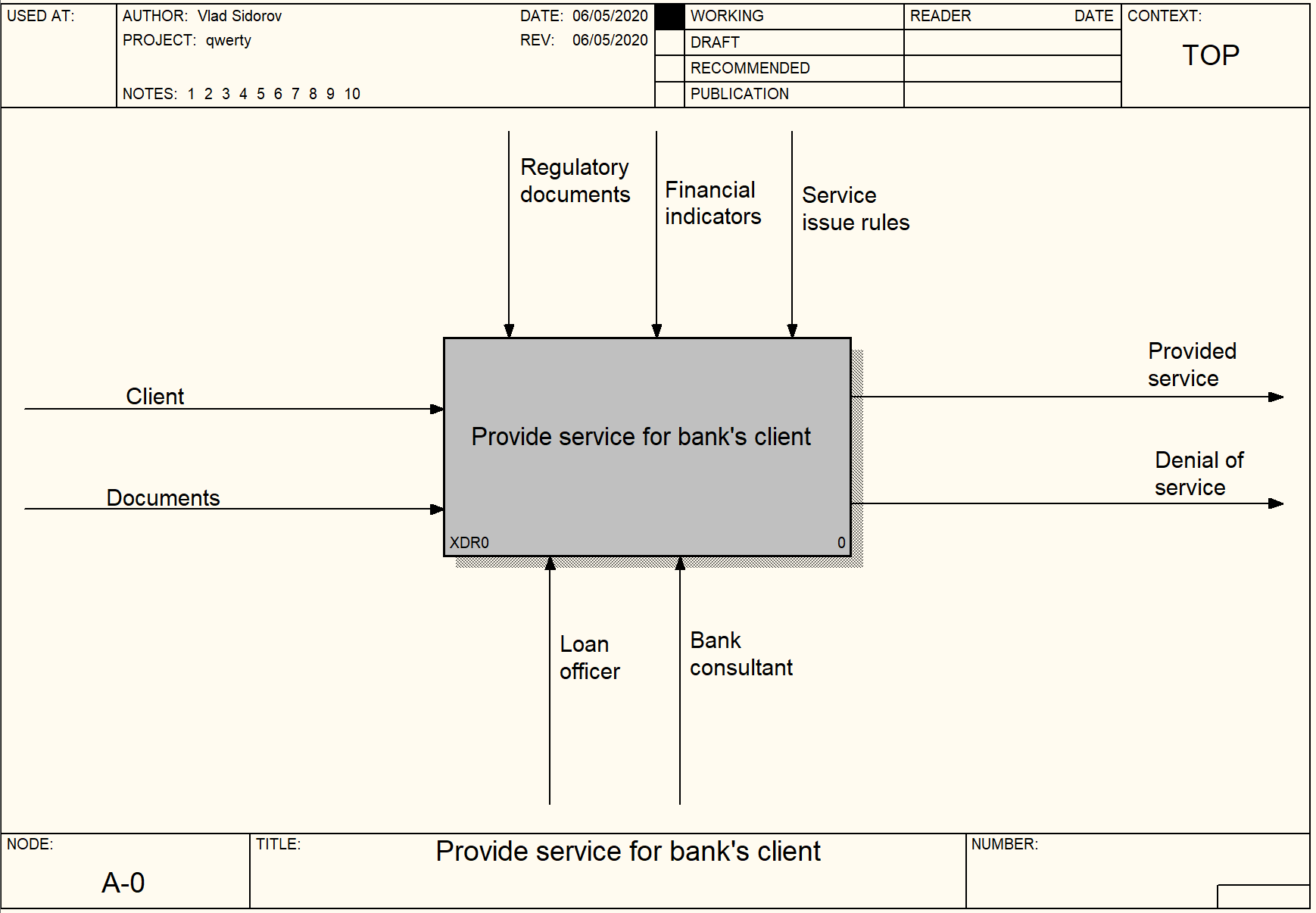


Рисунок 3.1 – Модель IDEF0. Контекстная диаграмма

Далее взглянем на рисунок 3.2 того же приложения А. Проведя декомпозицию контекстной диаграммы получилось несколько блоков подзадач. Для того, чтобы получить на выходе «Оказание услуги» или «Отказ об оказании услуг» необходимо выполнить следующие действия: «Выбрать банк», «Выбрать услугу» и «Оформить услугу».



Рисунок 3.2 – Декомпозиция контекстной диаграммы А-0

(первый уровень декомпозиции)

На рисунке 3.3 для наглядного выделения блока «Выбрать банк» была проведена декомпозиция на 2 блока. В начале нам предоставляется на входе «Клиент». Мы видим разветвленную линейную последовательность блоков: «Рассмотреть список предложений банка» и «Принять окончательное решение по выбору банка». Следовательно, на выходе из этого уровня мы будем иметь коммерческий банк для данного клиента.

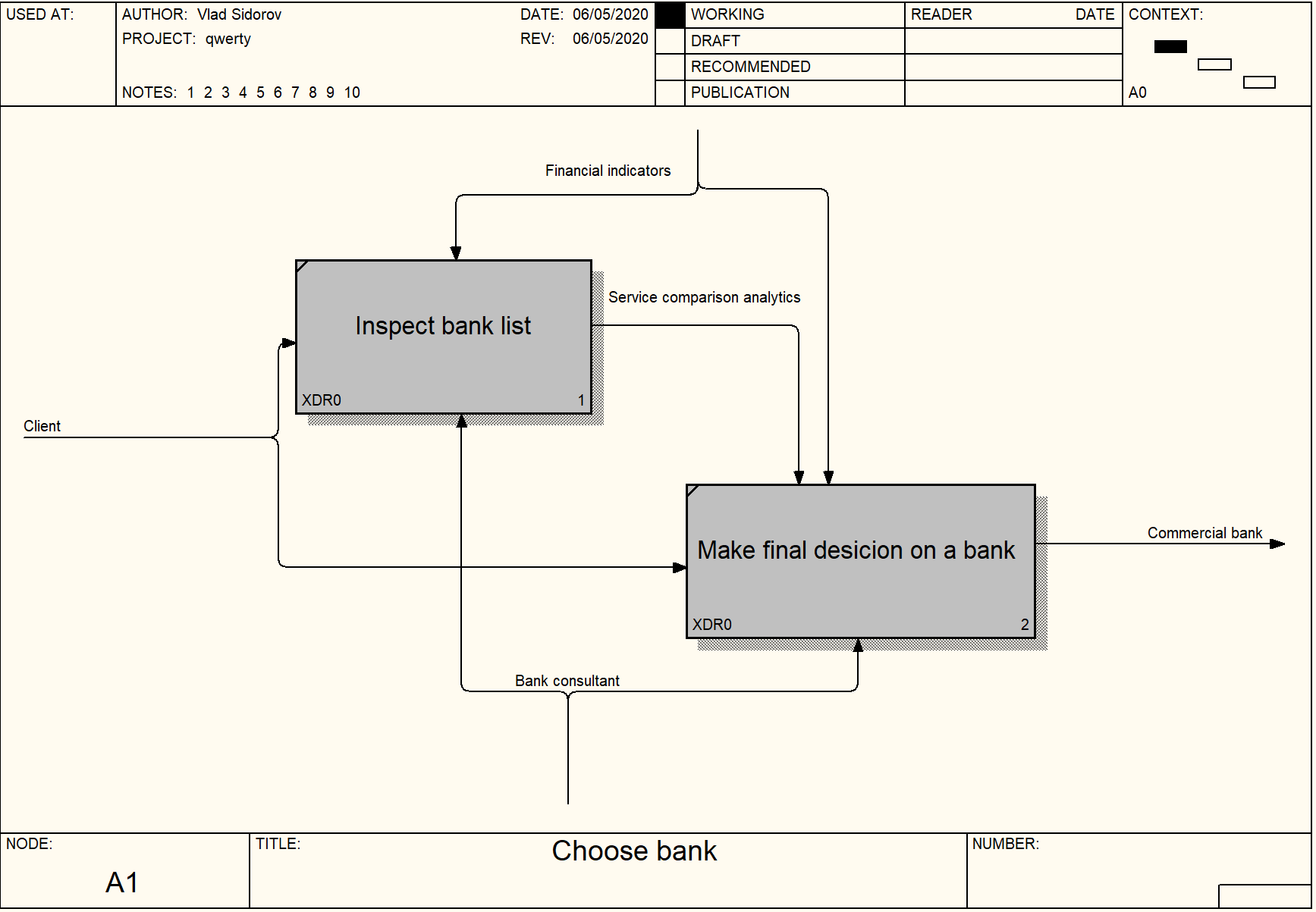


Рисунок 3.3 – Декомпозиция процесса «Выбрать банк»

Возвращаемся по дереву наверх к блоку А0 и декомпозируем процесс «Выбрать услугу» на 3 блока: «Проконсультироваться с работниками банка» и «Выбрать кредит» или «Выбрать депозит». Данный процесс декомпозируется на процесс с ветвлением, так как дает право выбора между двумя разными услугами.

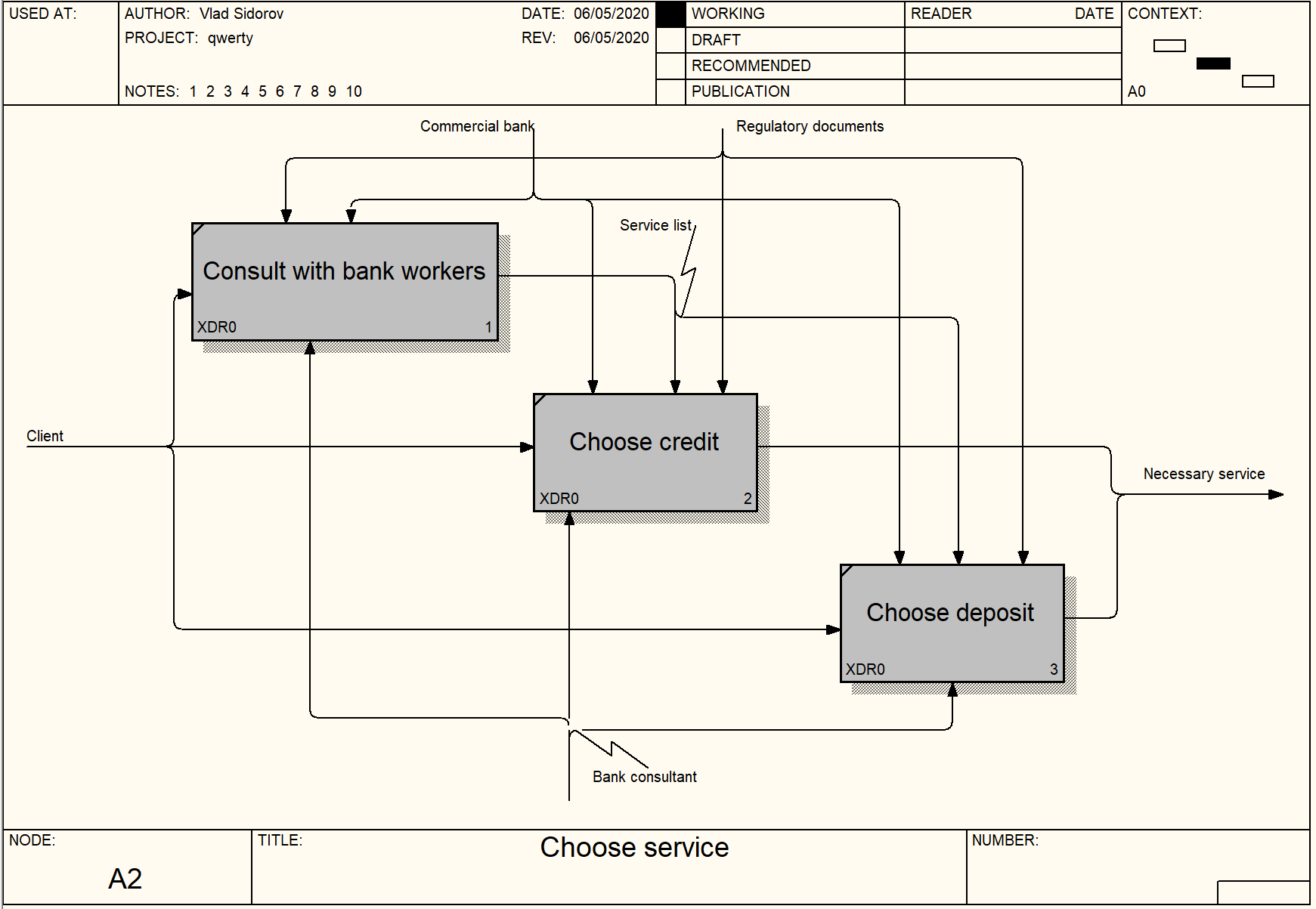


Рисунок 3.3 – Декомпозиция процесса «Выбрать услугу»

Проведем декомпозицию третьего процесса из блока А0 «Оформить услугу». На рисунке 3.4 мы видим декомпозицию на 3 процесса «Собрать данные о клиенте», «Обработать данные о клиенте» и «Оформить документы о предоставлении услуги». Выходными результатами будут либо «Оказание услуги», либо «Отказ об оказании услуги».

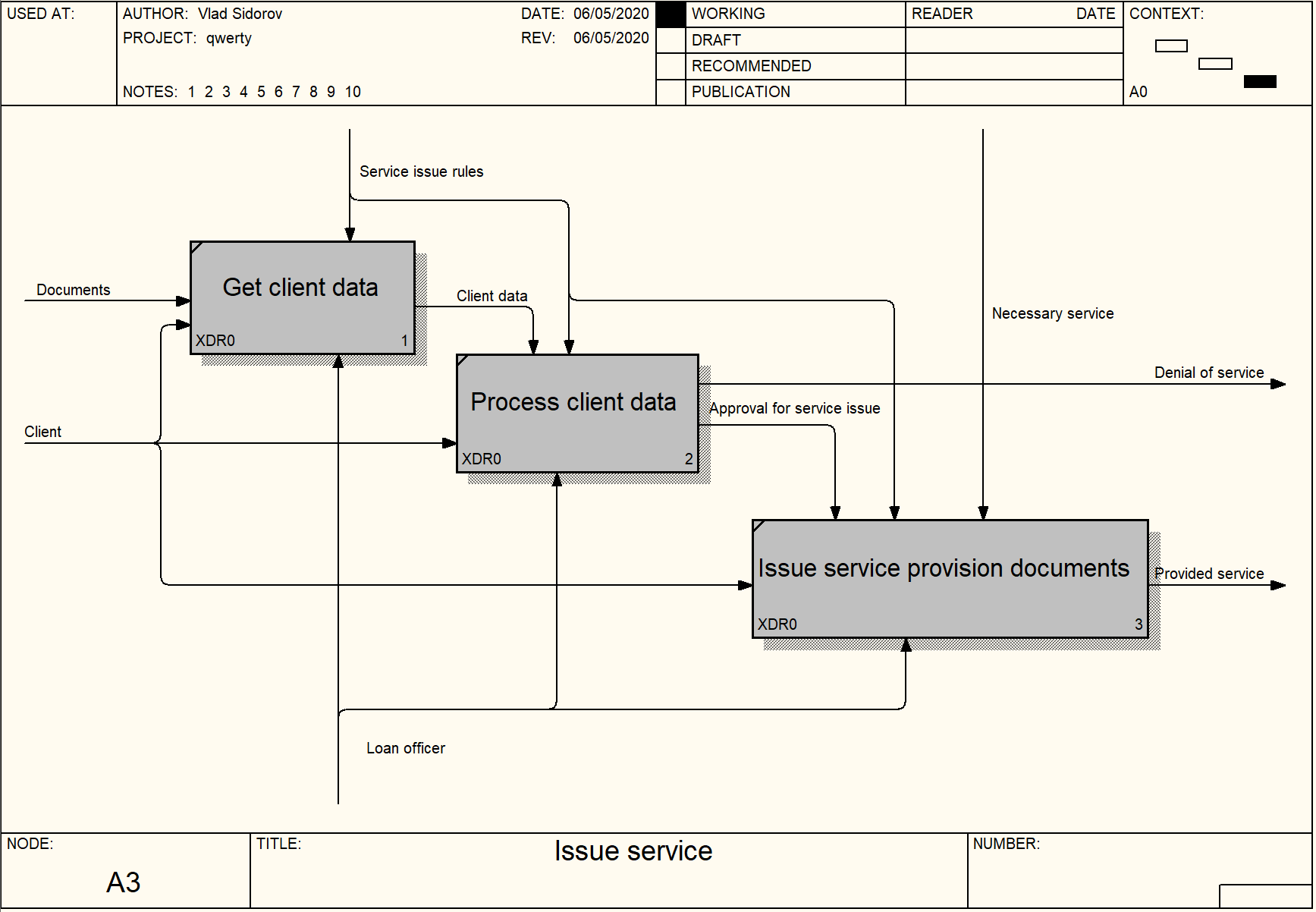


Рисунок 3.4 – Декомпозиция процесса «Оформить услугу»

Для конкретизации входных и выходных параметров был декомпозирован процесс «Собрать данные о клиенте» на 2 блока: «Сделать запрос по месту работы» и «Получить необходимые сведения». На выходе мы получаем «Данные о клиентах».

1. ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ

Цель информационной модели заключается в выработке непротиворечивой интерпретации данных и взаимосвязей между ними, что необходимо для интеграции, совместного использования и управления целостностью данных. Появление понятия концептуальной схемы данных привело к методологии семантического моделирования данных, то есть к определению значений в контексте их взаимосвязей с другими данными. Методологией семантического моделирования данных является IDEF1X, основанная на концепции Сущность-Отношение. Это инструмент для анализа информационной структуры систем различной природы. Информационная модель, построенная с помощью IDEF1X-методологии, представляет логическую структуру информации об объектах системы.

Логическая схема информационной модели в русскоязычном варианте находится в Приложении Б. Физическая схема информационной модели находится в Приложении В.

В базе данных bank хранится 6 сущностей, объединенных зависимой

6-ой сущностью, представляющей данные о компании. Каждая из 6 сущностей идентифицирована собственным ключом. У сущности, представляющей информацию о компании, имеются также внешние ключи, отражающие зависимости данной сущности от каждой из 5 остальных.

Сущность Company хранит в себе primary key «idcompany», который содержит уникальный номер компании. Также хранит в себе 5 вторичных ключей:

* «company\_accounts\_id», является вторичным ключом для таблицы «Accounts»;
* «сompany\_credits\_idcredits», является вторичным ключом для таблицы «Credits»;
* «company\_creditlist\_idcred», является вторичным ключом для таблицы «Creditlist»;
* «company\_deposites\_iddeposite»», является вторичным ключом для таблицы «Deposites»;
* «company\_depositelist\_iddeposite»», является вторичным ключом для таблицы «Depositslist».

Сущность «Credits» содержит в себе уникальный ключ «idcredits». Также поля:

* value (размер кредита);
* percent (процентная ставка по кредиту);
* term (срок погашения кредита);
* creditname (название кредита);
* idclients (идентификатор клиента).

Сущность «Accounts» содержит в себе уникальный ключ «id». Также поля:

* login (логин пользователя системы);
* password (пароль пользователя системы);
* money (сумма располагаемых средств);
* role (роль в системе: администратор или сотрудник отдела кредитования).

Сущность «Creditlist» содержит в себе уникальный ключ «idcred». Также поля:

* iduser (вторичный ключ для обработки списка кредитов конкретного сотрудника отдела кредитования)
* date (дата формирования отчета).

Сущность «Deposites» содержит в себе уникальный ключ «iddeposites». Также поля:

* value (размер депозита);
* percent (процентная ставка по депозиту);
* term (срок действия депозита);
* depositname (название вклада);
* idclient (идентификатор клиента).

Сущность «Depositelist» содержит в себе уникальный ключ «iddeposite». Также поля:

* iduser (вторичный ключ для обработки списка кредитов конкретного сотрудника отдела кредитования)
* date (дата формирования отчета).

Сущность «Clients» содержит в себе уникальный ключ «idclients». Также поля:

* surname (фамилия клиента);
* name (имя клиента);
* secondname (отчество клиента).

5 ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩУЮ БИЗНЕС-ЛОГИКУ ВЕБ-ПРИЛОЖЕНИЯ НА ФРЕЙМВОРКЕ SPRING

Фреймворк Spring MVC обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение (далее — Вид) — Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними.

Model (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO («Старых добрых Java-объектов», или бинов).

View (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.

Controller (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.Сокет может служить для подключения системы ввода-вывода в Java к другим программам, которые могут находиться как на локальной машине, таки на любой другой машине в Интернете.

DispatcherServlet

Вся логика работы Spring MVC построена вокруг DispatcherServlet, который принимает и обрабатывает все HTTP-запросы (из UI) и ответы на них.

Ниже приведена последовательность событий, соответствующая входящему HTTP-запросу:

После получения HTTP-запроса DispatcherServlet обращается к интерфейсу HandlerMapping, который определяет, какой Контроллер должен быть вызван, после чего, отправляет запрос в нужный Контроллер.

Контроллер принимает запрос и вызывает соответствующий служебный метод, основанный на GET или POST. Вызванный метод определяет данные Модели, основанные на определённой бизнес-логике и возвращает в DispatcherServlet имя Вида (View).

При помощи интерфейса ViewResolver DispatcherServlet определяет, какой Вид нужно использовать на основании полученного имени.

После того, как Вид (View) создан, DispatcherServlet отправляет данные Модели в виде атрибутов в Вид, который в конечном итоге отображается в браузере. [ССЫЛКА]

Ниже приведена схема работы spring-приложения (рис. 5.2)

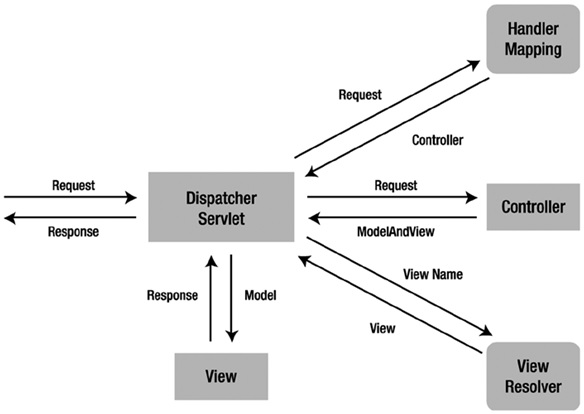


Рисунок 5.2 – Описание алгоритма работы Spring-приложения

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для работы с программой необходимо запустить сначала сервер, а потом клиент. При запуске клиента появляется окно авторизации. Введем реквизиты администратора. Предусмотрено скрытие введенного пароля.

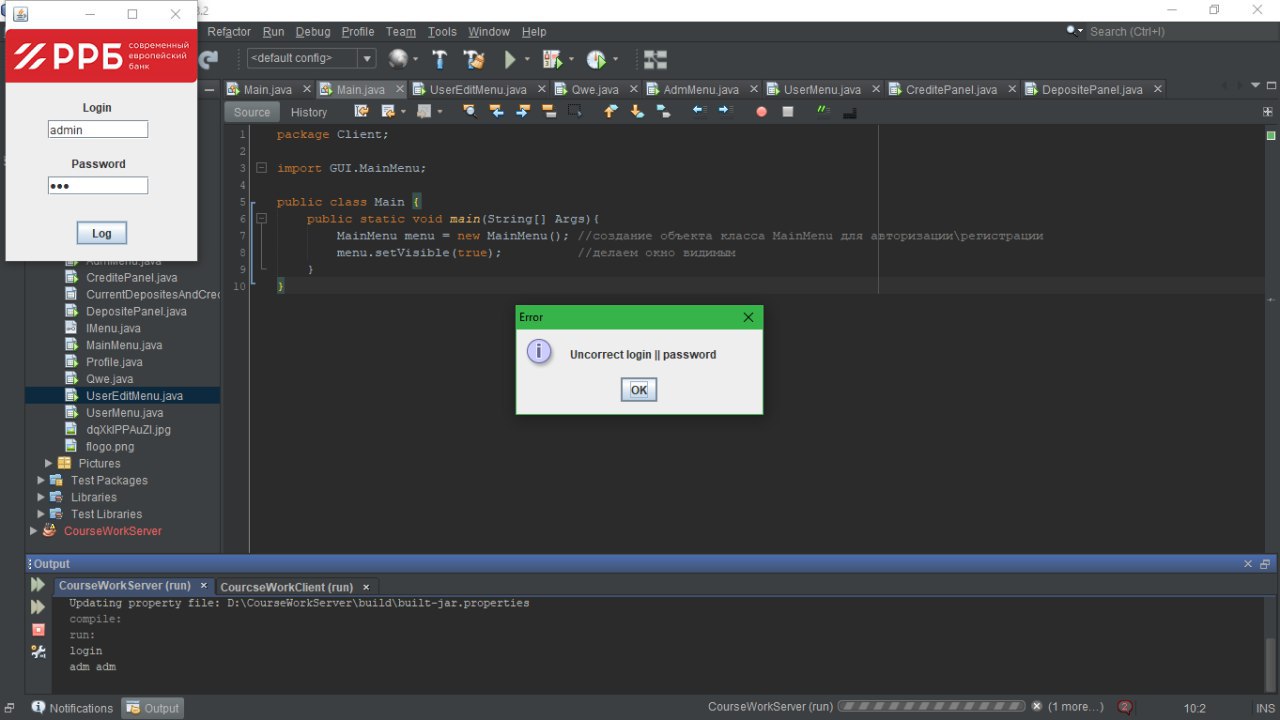


Рисунок 5.10 – Авторизация в программе администратора

Выше были введены неверные данные, и мы получаем сообщение об ошибке как на стороне клиента, так и на стороне сервера.

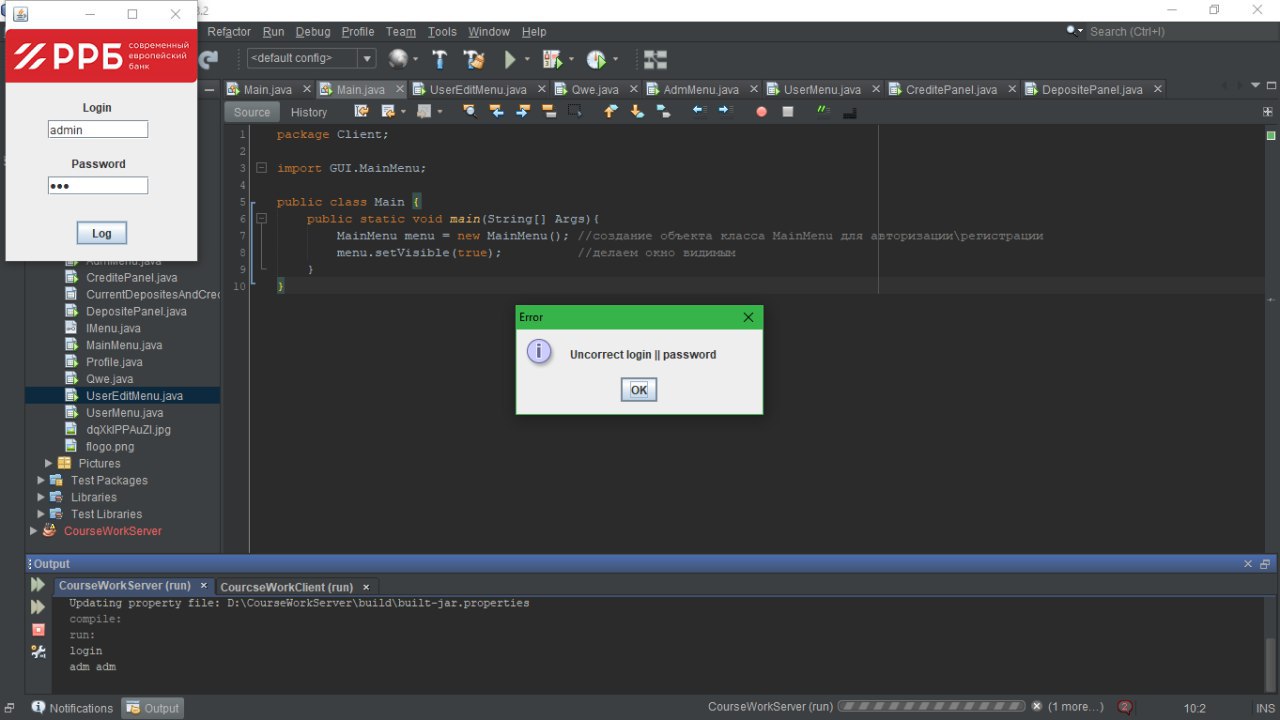


Рисунок 5.11 – Сообщение «Ошибка авторизации» на стороне клиента

После корректно введенных данных мы попадаем в панель администратора, где видим четыре панели для работы: «Пользователи», «Кредиты», «Депозиты» и «Назад».

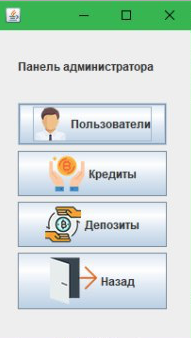


Рисунок 5.12 – Панель Администратора

Переходим в меню управления пользователей, где видим таблицу всех пользователей.

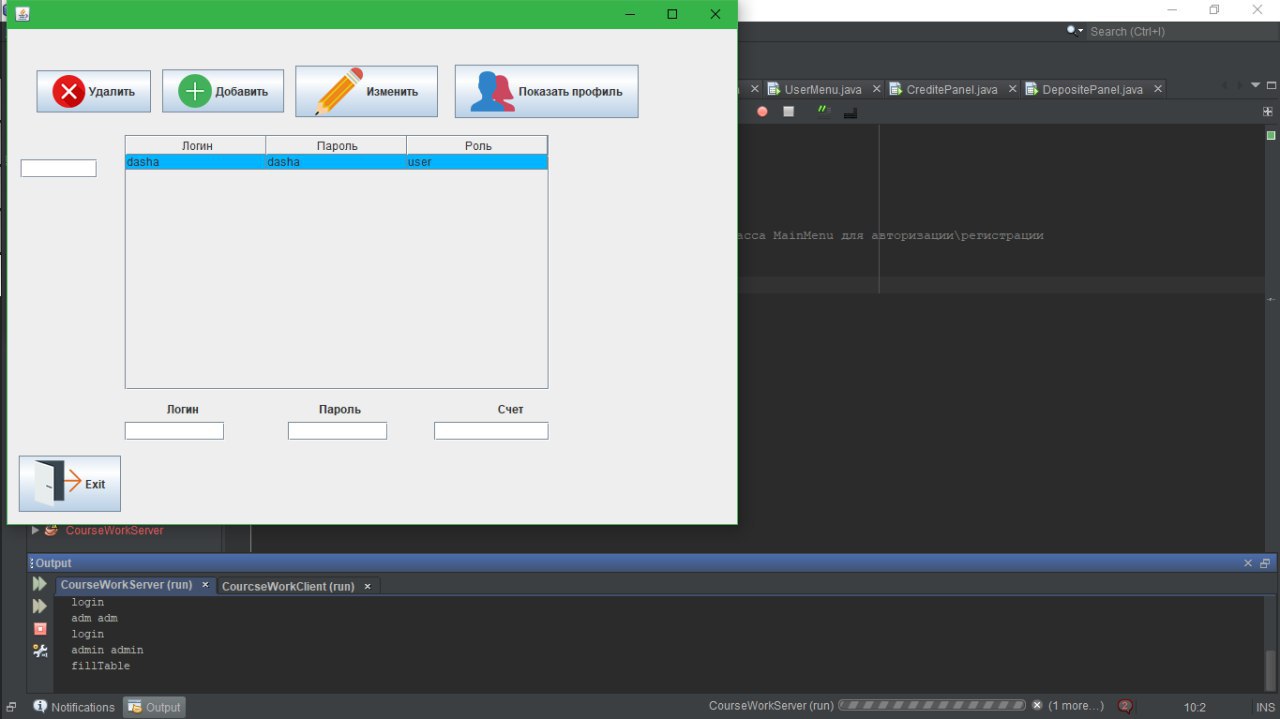


Рисунок 5.12 – Панель управления пользователями

Добавим в таблицу нового пользователя со следующими полями.

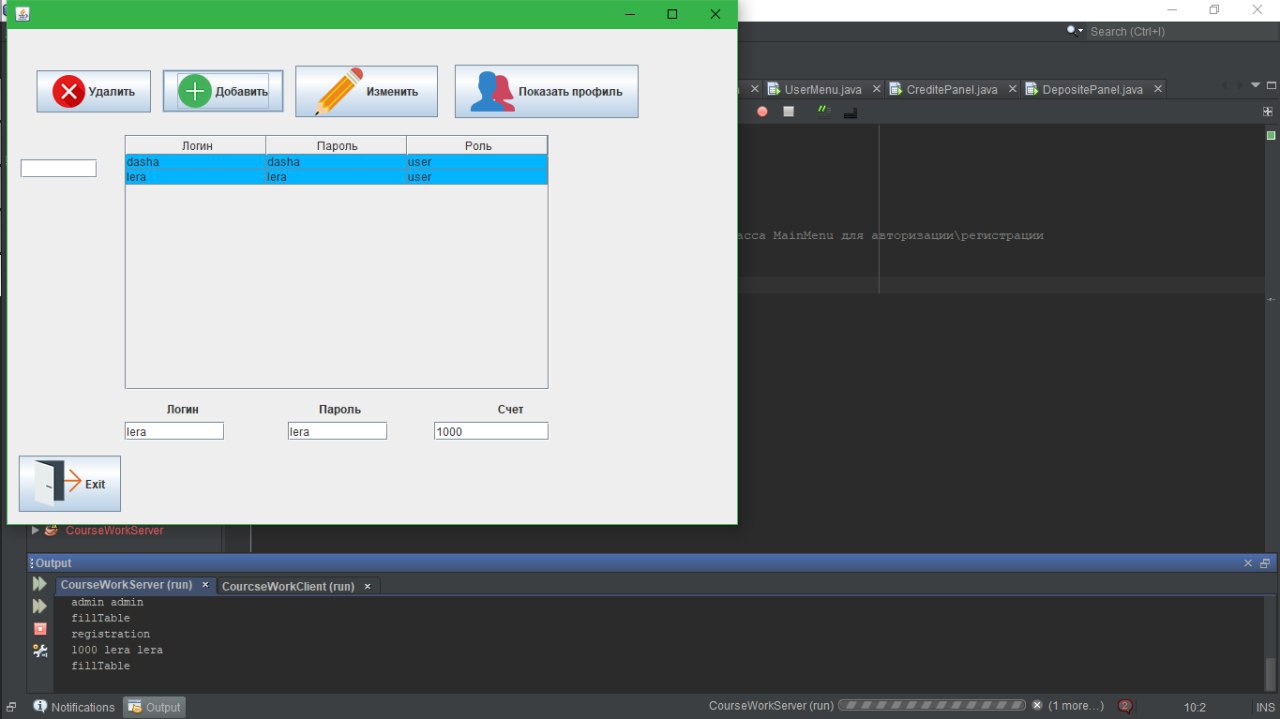


Рисунок 5.13 – Добавление нового пользователя.

В данной системе есть только один глобальный администратор и неограниченное количество возможных пользователей. Это необходимо для того, чтобы данные управлялись только одним человеком. Далее удалим нового созданного поьзователя.

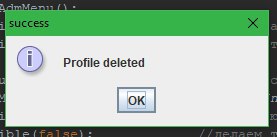


Рисунок 5.14 – Подтверждение удаления пользователя

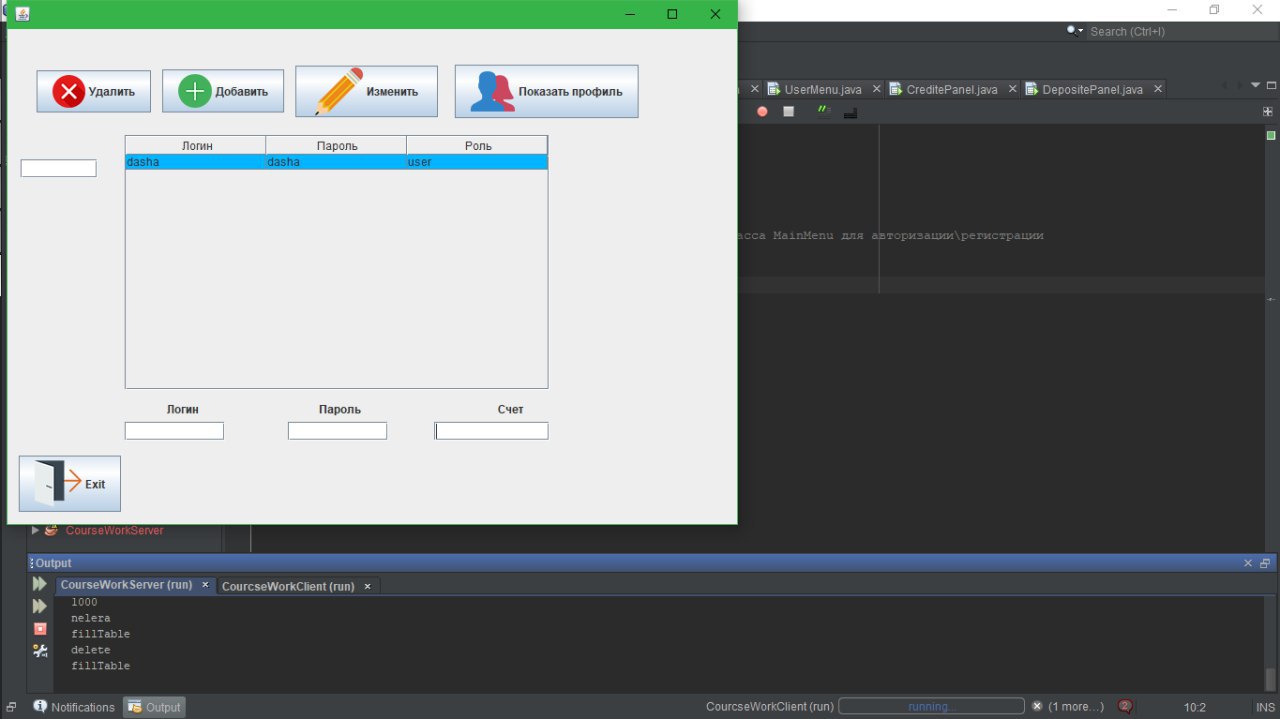


Рисунок 5.15 – Таблица пользователей

Все действия, которые выполняются в клиенте, отображаются в консоли сервера.

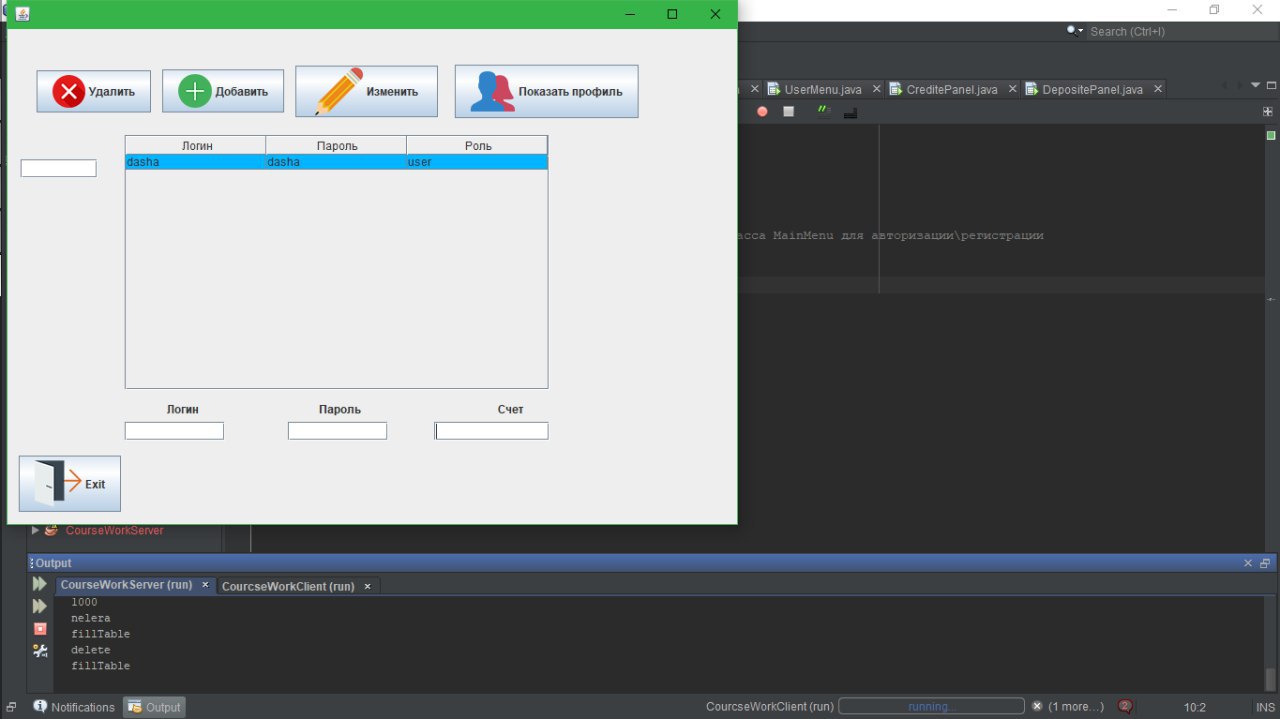


Рисунок 5.16 – Отображение действий в консоли сервера

Теперь добавим несколько пользователей в систему. У каждого пользователя уровень доступа – пользователь системы.

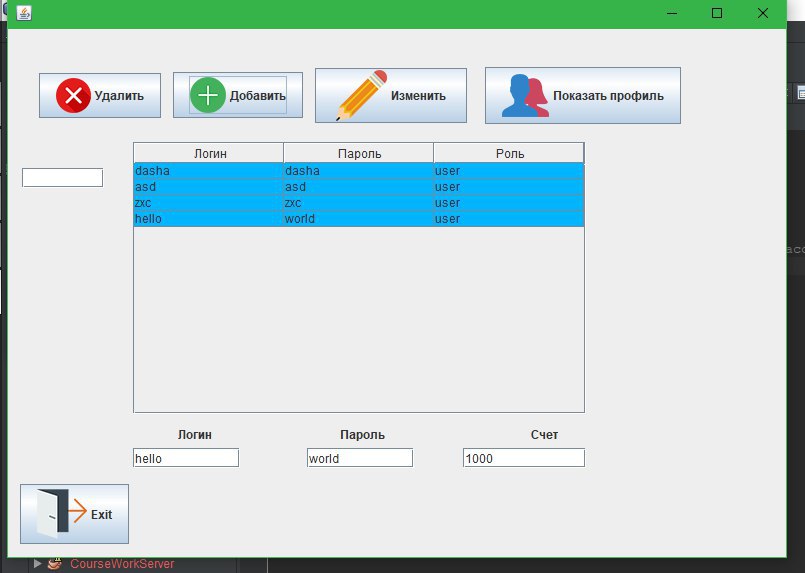


Рисунок 5.17 – Таблица пользователей

Теперь продемонстрируем поиск по любому параметру. Для демонстрации введем логин одного из сотрудников. Видим вывод совпадений по значениям.

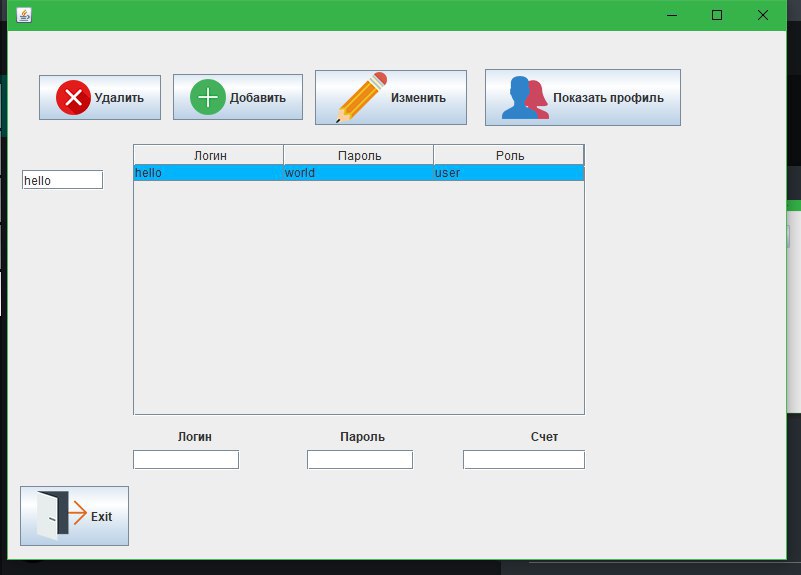


Рисунок 5.18 – Поиск по параметру

Можно посмотреть профиль пользователя для ознакомления баланса средств на оказание им услуг и данными по услугам.

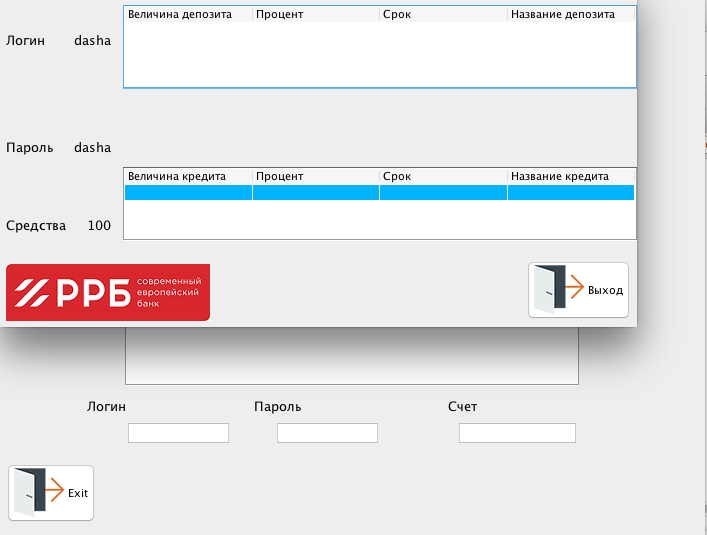


Рисунок 5.19 – Профиль сотрудника

Теперь выйдем из «Панели работы с пользователями» и выберем «Работу с депозитами». В данной таблице мы можем добавлять, редактировать, сортировать и удалять данные.

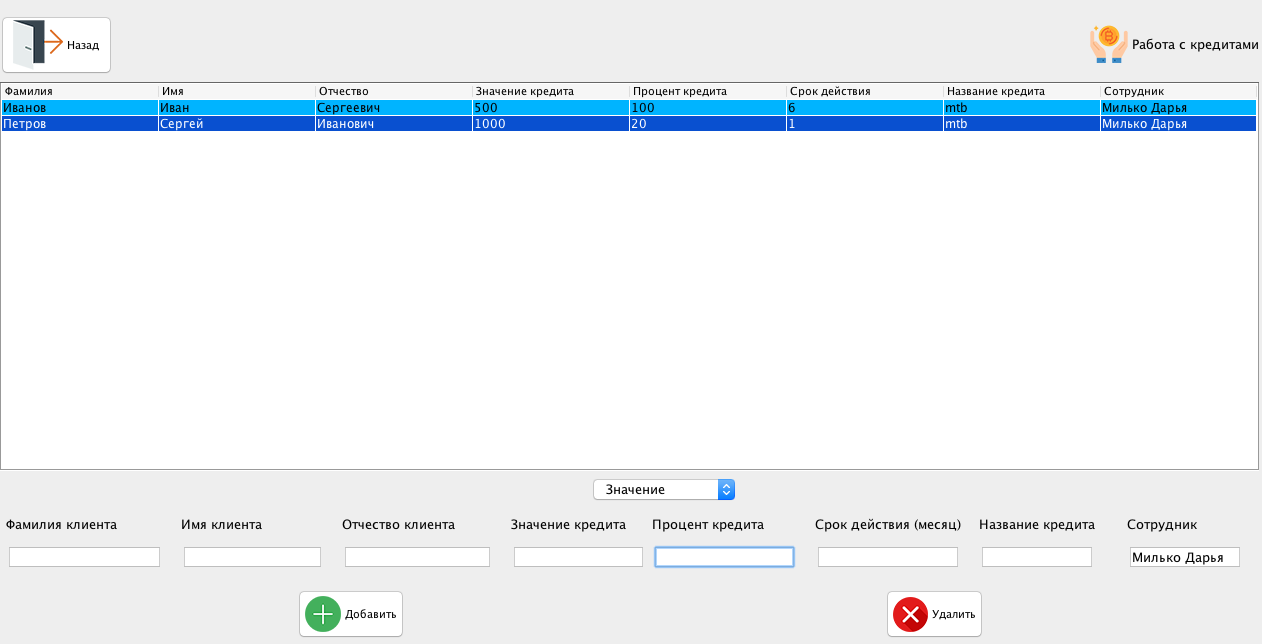


Рисунок 5.20 – Таблица «Работа с депозитами»

Отсортируем нашу таблицу по процентам. Также нашу таблицу можно сортировать по любым параметрам. В добавок, можно удалять и добавлять данные по депозитам.

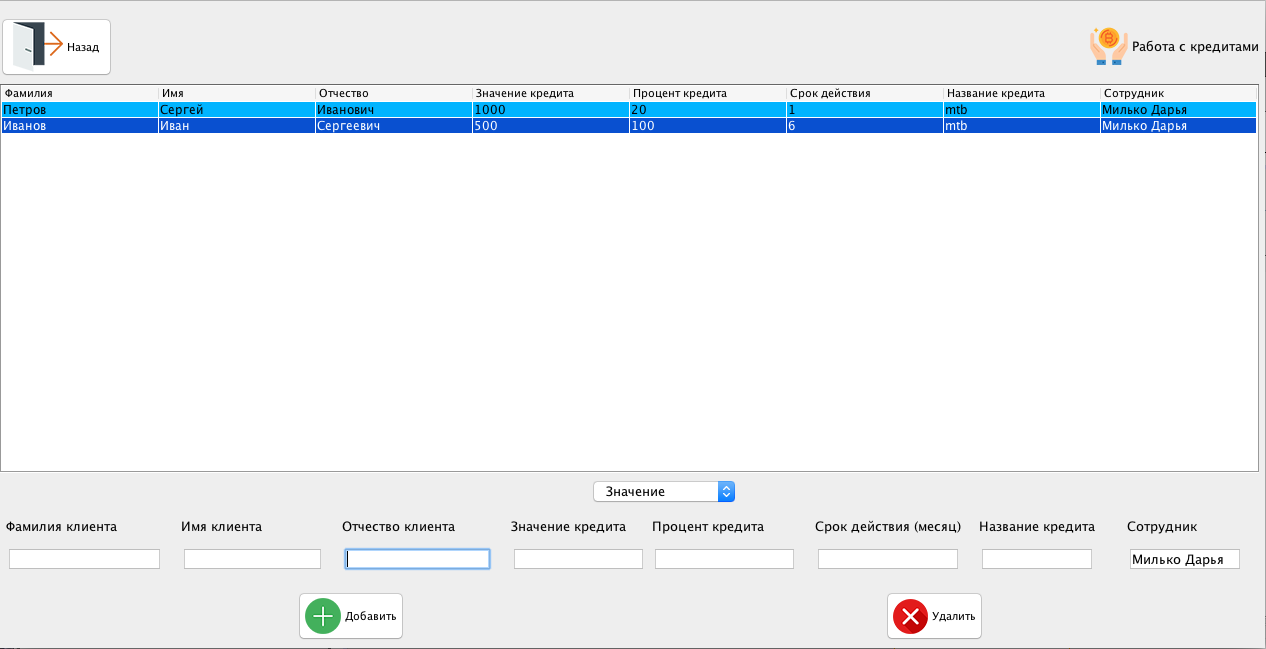


Рисунок 5.21 – Сортировка таблицы по параметру

Такая же таблица существует и для работы с кредитами с тем же функционалом. Добавим новый кредит в таблицу. Для этого заполним форму.

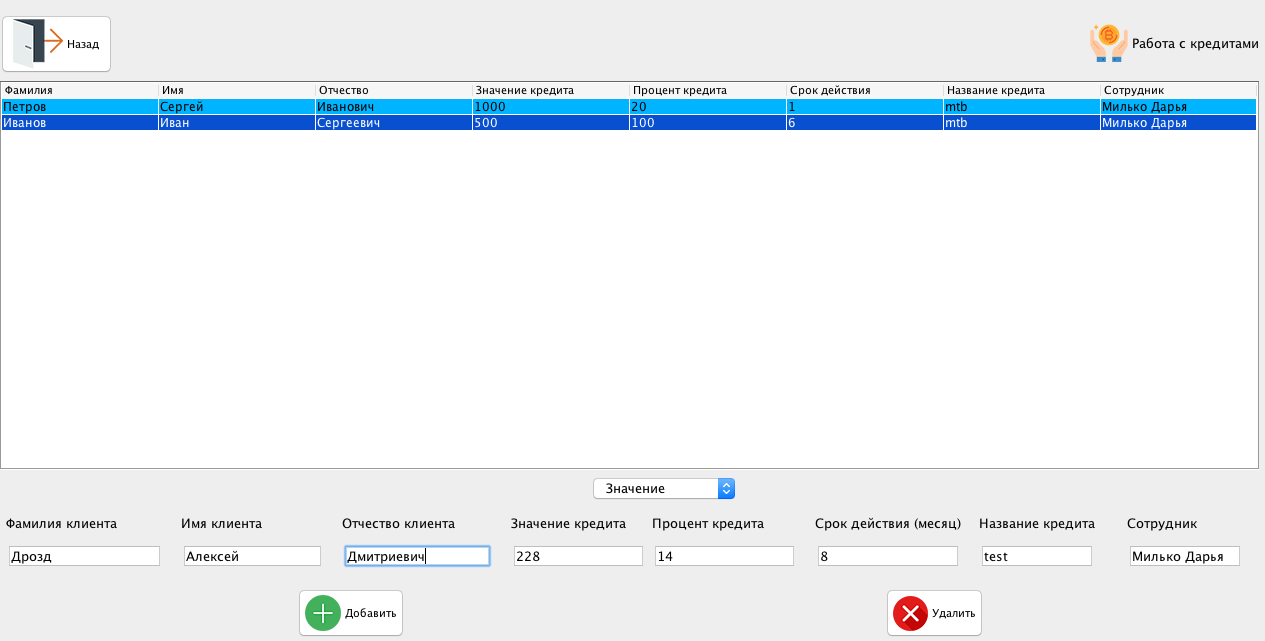


Рисунок 5.22 – Добавление кредита

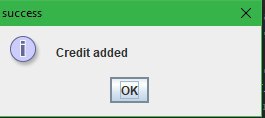


Рисунок 5.23 – Уведомление о добавлении кредита

Удалим данные о кредите.

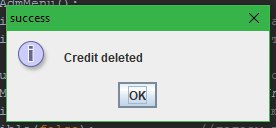


Рисунок 5.24 – Уведомление об удалении кредита администратором

Вернемся в главное меню и войдем под реквизитами пользователя. После корректно введенных данных мы видим меню пользователя сотрудника банка.



Рисунок 5.25 – Панель работы пользователя

Выберем оформление кредита. Добавим новый кредит.

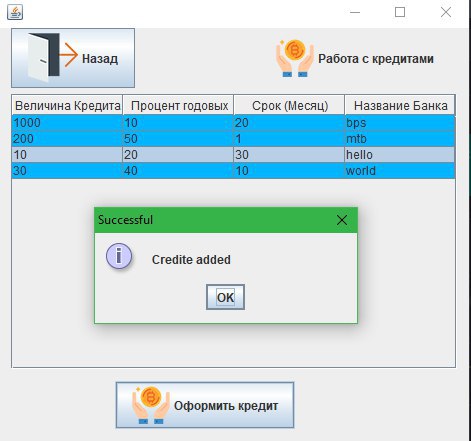


Рисунок 5.26 – Добавление кредита

Удалим кредиты все, кроме первого.

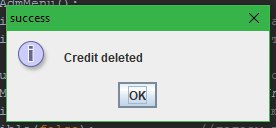


Рисунок 5.27 – Удаление кредита

Система внутри рассчитывает значения и определяет период кредитования и заносит его в базу данных. В работе с депозитами имеется такой же функционал. Перейдем в меню профиля и видим следующий функционал: «Погасить депозит», «Погасить кредит», «Распечатать документ по депозитам» и «Распечатать документ по кредитам». При погашении депозита и кредита, денежные средства с расчетного счета снимаются и зачисляются, соответственно. Попробуем распечатать заполненный формуляр кредитного договора для подписи клиентом банка.

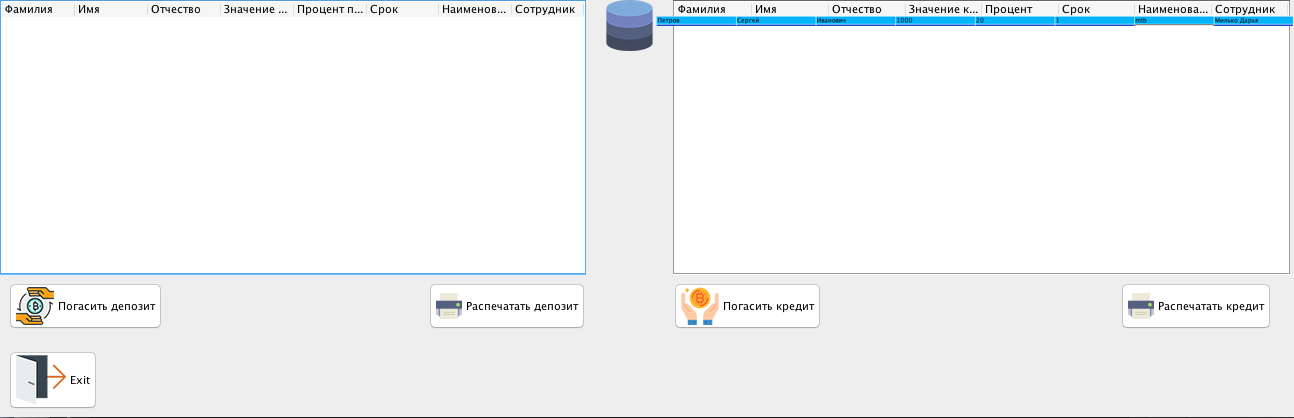


Рисунок 5.28 – Профиль сотрудника для печати

Формуляр выводиться не совсем корректно, так как библиотека для работы с PDF не поддерживает кириллицу, поэтому формуляр подготовлен транслитом.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта, была разработана программа, которая позволяет банковским работникам отдела кредитования регистрировать новые кредиты и депозиты, так же позволяет создавать отчеты по движениям денежных средств. В том числе, программа позволяет управлять сотрудниками отдела кредитования и регламентировать их деятельность.

Программа имеет интуитивно понятный, удобный интерфейс, что безусловно является притягательным качеством данного продукта и сотруднику не придется тратить время на изучение сложных программных модулей. Исходя из вышеизложенного, можно сделать вывод о том, что программа может использоваться в качестве начальным прототипом для реальной банковской системы. Однако для практического внедрения данного программного приложения необходимо добавить ряд возможностей, к которым, на мой взгляд, можно отнести следующие:

Если говорить об архитектуре, то было разработано клиент-серверное приложение, представляющее собой систему планирования бюджета на предприятии.

Тот факт, что приложение клиент-серверное, предполагает возможность нахождения пользователя клиентской стороны на одной машине, а сервера с базой данных – на другой. Поскольку приложение написано средствами языка Java, существует возможность его использования на различных платформах.

Все финансовые сведения о деятельности банка и его сотрудниках сохранены в таблицах базы данных, что соответствует современному подходу хранения информации. Работа с базой данных, и вся бизнес-логика системы реализована только на серверной части программы.

Системы расчета кредитов и депозитов передаются напрямую в базу данных, позволяя пользователям системы изменять их, удалять, просматривать, сохранять отчеты.

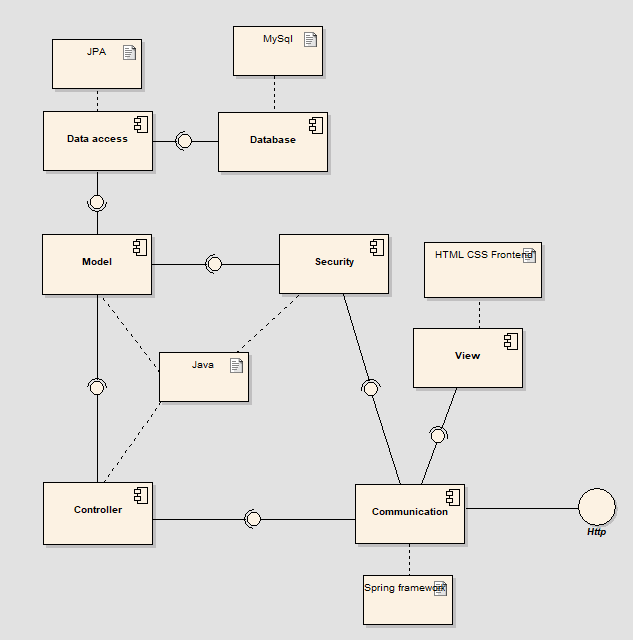
Предусмотрена обработка исключительных ситуаций, благодаря чему возникновение ошибок при работе с системой снижена до возможного минимума.

На основе выполненной курсовой работы можно сделать вывод о том, что все задачи, поставленные в курсовом проекте, достигнуты, а требования соблюдены.

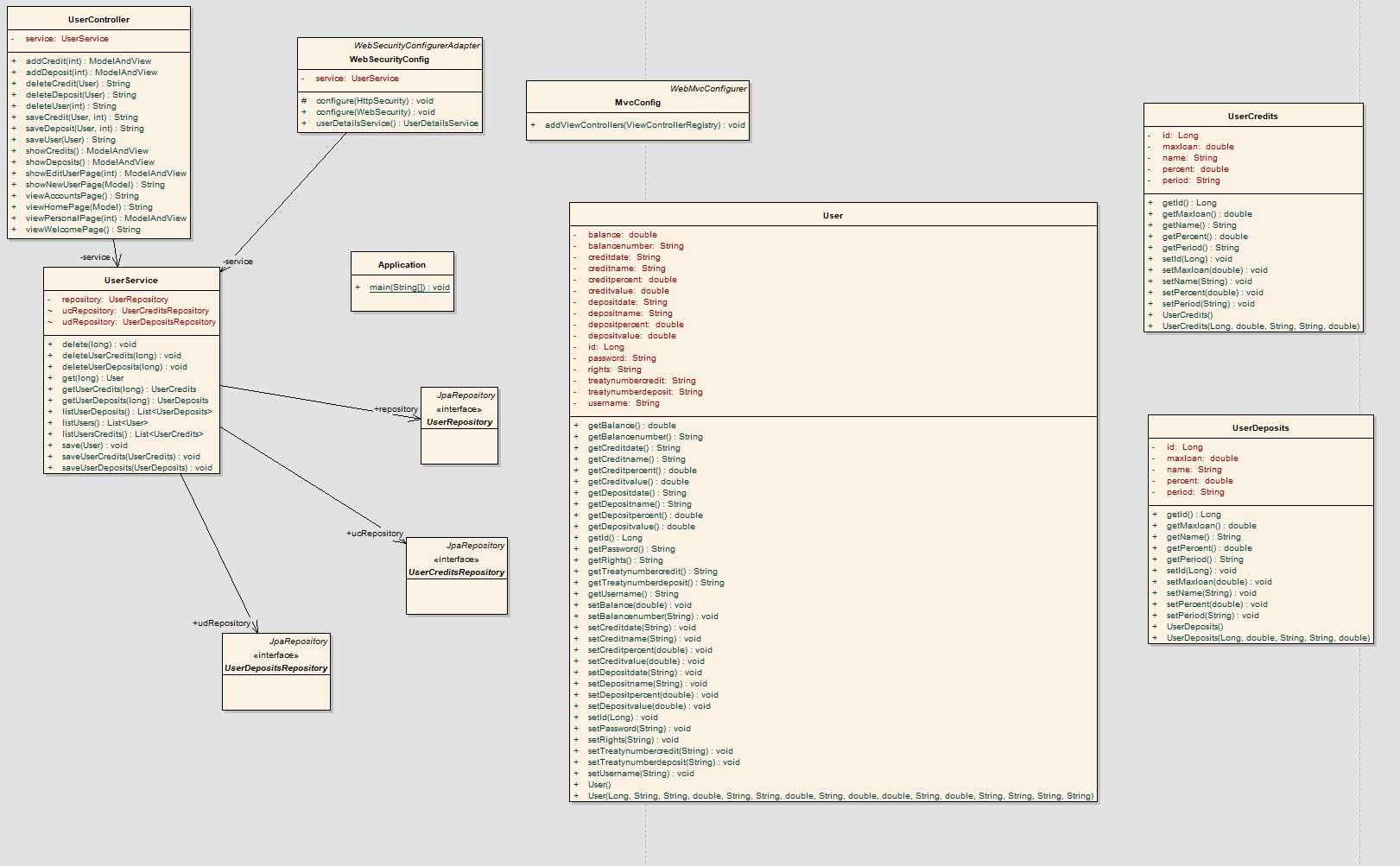
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://habr.com/ru/post/336816/>
2. <https://infobank.by/oficialjnyj-rejting-bankov-aktivy-obyazateljstva-kapital-2017/>.
3. Хорстманн, Кей С., Корнелл, Гари – Java 2. Библиотека профессионала, том II. Тонкости программирования, 7-е изд. : Пер. с англ. – М.: Издательский дом «Вильямс», 2008. – 1168 с.: ил. – Парал. тит. англ.
4. Шилдт, Герберт, Холмс, Джеймс – Искусство программирования на Java. : Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 336 с.: ил. – Парал. тит. англ.
5. Java Code Conventions
6. Брюс Эккель – Философия Java. Библиотека программиста. Издание 3-е. : Издательство «Питер», 2003. – 976 с.
7. Дюбуа Поль – MySQL. Пер. с англ. : Уч. пос. – М. .: Издательский дом «Вильямс», 2001. – 816 с.: ил. – Парал. тит. англ.
8. Аткинсон Леон – MySQL. Библиотека профессионала, том II. Тонкости программирования, 7-е изд. : Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 624 с.: ил. – Парал. тит. англ.
9. MySQL / Ларри Ульман Пер. с англ. Слинкина А. А. – М. : ДМК «Пресс» СПб, Питер – 2004. – 352 с.: ил.
10. Шилдт, Герберт – SWING: руководство для начинающих. : Пер. с англ. – М.: ООО «И.Д. Вильямс», 2007. – 704 с.: ил. – Парал. тит. англ.
11. Steven GUTZ – Up to Speed with Swing. User interfaces with Java foundation classes. Second Edition – Manning

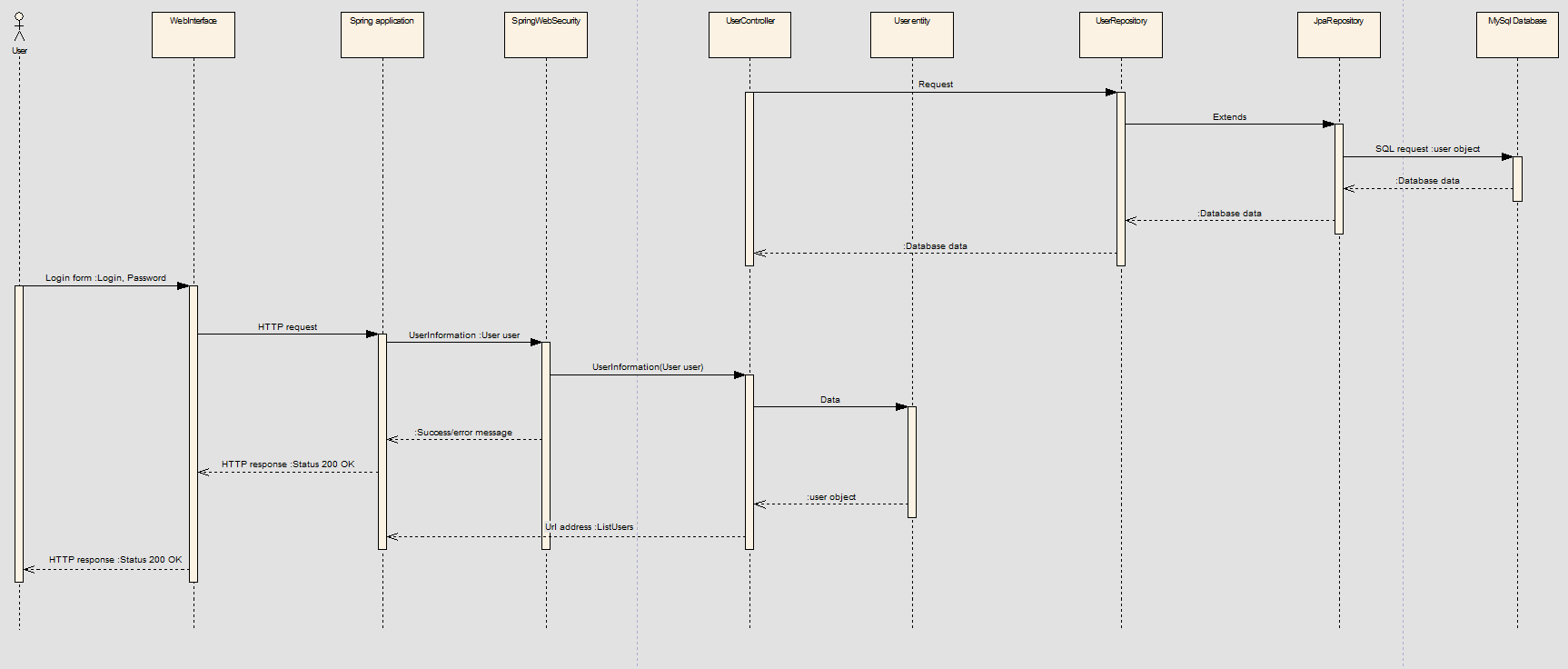
Приложение В  
(диаграмма компонентов)



Приложение Г  
(диаграмма классов приложения)



Приложение Ж  
(диаграмма последовательности)



Приложение З  
(листинг генерации базы данных)

create schema if not exists sales collate utf8mb4\_0900\_ai\_ci;  
  
create table if not exists user  
(  
 id int auto\_increment  
 primary key,  
 username varchar(45) not null,  
 password varchar(45) not null,  
 balance double not null,  
 rights varchar(45) not null,  
 creditdate varchar(45) null,  
 creditvalue double null,  
 depositdate varchar(45) null,  
 depositvalue double null,  
 creditpercent double null,  
 creditname varchar(45) null,  
 depositpercent double null,  
 depositname varchar(45) null,  
 treatynumbercredit varchar(45) null,  
 treatynumberdeposit varchar(45) null,  
 balancenumber varchar(45) null  
)  
charset=utf8;  
  
create table if not exists user\_credits  
(  
 id int not null  
 primary key,  
 percent double null,  
 name varchar(200) null,  
 period varchar(45) null,  
 maxloan double null  
);  
  
create table if not exists user\_deposits  
(  
 id int not null  
 primary key,  
 percent double null,  
 name varchar(200) null,  
 period varchar(45) null,  
 maxloan double null  
);

Приложение И

UserController

package com.project.controller;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.List;  
  
import com.project.UserService;  
import com.project.entity.User;  
import com.project.entity.UserCredits;  
import com.project.entity.UserDeposits;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.servlet.ModelAndView;  
  
@Controller  
public class UserController {  
  
 @Autowired  
 private UserService service;  
  
 @RequestMapping(value = "/")  
 public String viewWelcomePage() {  
  
 return "index";  
 }  
  
 @RequestMapping(value = "accounts")  
 public String viewAccountsPage() {  
  
 return "accounts\_plan";  
 }  
  
 @RequestMapping("/home")  
 public String viewHomePage(Model model) {  
 List<User> listUsers = service.listUsers();  
 model.addAttribute("listUsers", listUsers);  
  
 return "home";  
 }  
  
 @RequestMapping("/personal/{id}")  
 public ModelAndView viewPersonalPage(@PathVariable(name = "id") int id) {  
 ModelAndView mav = new ModelAndView("personal\_page");  
 User user = service.get(id);  
 mav.addObject("user", user);  
  
 return mav;  
 }  
  
 @RequestMapping("/new")  
 public String showNewUserPage(Model model) {  
 User user = new User();  
 user.setCreditdate("No credit");  
 user.setCreditvalue(0);  
 model.addAttribute("user", user);  
  
 return "new\_user";  
 }  
  
 @RequestMapping(value = "/save", method = RequestMethod.*POST*)  
 public String saveUser(@ModelAttribute("user") User user) {  
 service.save(user);  
  
 return "redirect:/home";  
 }  
  
 @RequestMapping(value = "/deletecredit", method = RequestMethod.*POST*)  
 public String deleteCredit(@ModelAttribute("user") User user) {  
 user.setBalance(user.getBalance() - ((user.getCreditvalue() \* (user.getCreditpercent() / 100)) + user.getCreditvalue()));  
 user.setCreditdate("No credit");  
 user.setCreditvalue(0);  
 user.setCreditpercent(0);  
 user.setCreditname("No credit");  
 user.setTreatynumbercredit("No credit");  
 service.save(user);  
  
 return "redirect:/personal/" + user.getId();  
 }  
  
 @RequestMapping(value = "/deletedeposit", method = RequestMethod.*POST*)  
 public String deleteDeposit(@ModelAttribute("user") User user) {  
 user.setBalance(user.getBalance() + ((user.getDepositvalue() \* (user.getDepositpercent() / 100)) + user.getDepositvalue()));  
 user.setDepositdate("No deposit");  
 user.setDepositvalue(0);  
 user.setDepositpercent(0);  
 user.setDepositname("No deposit");  
 user.setTreatynumberdeposit("No deposit");  
 service.save(user);  
  
 return "redirect:/personal/" + user.getId();  
 }  
  
 @RequestMapping("/edit/{id}")  
 public ModelAndView showEditUserPage(@PathVariable(name = "id") int id) {  
 ModelAndView mav = new ModelAndView("edit\_user");  
 User user = service.get(id);  
 mav.addObject("user", user);  
  
 return mav;  
 }  
  
 @RequestMapping("/delete/{id}")  
 public String deleteUser(@PathVariable(name = "id") int id) {  
 service.delete(id);  
 return "redirect:/home";  
 }  
  
 @RequestMapping("/addcredit/{id}")  
 public ModelAndView addCredit(@PathVariable(name = "id") int id) {  
 ModelAndView mav = new ModelAndView("add\_credit");  
 User user = service.get(id);  
 mav.addObject("user", user);  
 List<UserCredits> userCredits = service.listUsersCredits();  
 mav.addObject("userCredits", userCredits);  
  
 return mav;  
 }  
  
 @RequestMapping(value = "/savecredit/{id}", method = RequestMethod.*POST*)  
 public String saveCredit(@ModelAttribute("user") User user, @PathVariable(name = "id") int id) {  
 UserCredits userCredits = service.getUserCredits(id);  
 user.setCreditpercent(userCredits.getPercent());  
 user.setCreditname(userCredits.getName());  
 String currentDate = new SimpleDateFormat("yyyy-MM-dd").format(new Date());  
 user.setCreditdate(currentDate);  
 service.save(user);  
  
 return "redirect:/personal/" + user.getId();  
 }  
  
 @RequestMapping("/adddeposit/{id}")  
 public ModelAndView addDeposit(@PathVariable(name = "id") int id) {  
 ModelAndView mav = new ModelAndView("add\_deposit");  
 User user = service.get(id);  
 mav.addObject("user", user);  
 List<UserDeposits> userDeposits = service.listUserDeposits();  
 mav.addObject("userDeposits", userDeposits);  
  
 return mav;  
 }  
  
 @RequestMapping(value = "/savedeposit/{id}", method = RequestMethod.*POST*)  
 public String saveDeposit(@ModelAttribute("user") User user, @PathVariable(name = "id") int id) {  
 UserDeposits userDeposits = service.getUserDeposits(id);  
 user.setDepositpercent(userDeposits.getPercent());  
 user.setDepositname(userDeposits.getName());  
 String currentDate = new SimpleDateFormat("yyyy-MM-dd").format(new Date());  
 user.setDepositdate(currentDate);  
 service.save(user);  
  
 return "redirect:/personal/" + user.getId();  
 }  
  
 @RequestMapping("/showcredits")  
 public ModelAndView showCredits() {  
 ModelAndView mav = new ModelAndView("view\_credits");  
 List<UserCredits> userCreditsList = service.listUsersCredits();  
 mav.addObject("userCreditsList", userCreditsList);  
  
 return mav;  
 }  
  
 @RequestMapping("/showdeposits")  
 public ModelAndView showDeposits() {  
 ModelAndView mav = new ModelAndView("view\_deposits");  
 List<UserDeposits> userDepositsList = service.listUserDeposits();  
 mav.addObject("userDepositsList", userDepositsList);  
  
 return mav;  
 }  
}

WebSecurityConfig

package com.project.security;  
  
import com.project.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.builders.WebSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
  
import java.util.List;  
  
@Configuration  
@EnableWebSecurity  
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {  
  
 @Autowired  
 private UserService service;  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http  
 .authorizeRequests()  
 .antMatchers("/").permitAll()  
 .antMatchers("/new", "/delete/\*\*", "/edit/\*\*").hasRole("ADMIN")  
 .anyRequest().authenticated()  
 .and()  
 .formLogin()  
 .loginPage("/login")  
 .permitAll()  
 .and()  
 .logout()  
 .permitAll();  
 }  
  
 @Override  
 public void configure(WebSecurity web) {  
 web  
 .ignoring()  
 .antMatchers("/resources/\*\*", "/static/\*\*", "/css/\*\*", "/js/\*\*", "/images/\*\*", "/icon/\*\*");  
 }  
  
 @Bean  
 public UserDetailsService userDetailsService() {  
 List<com.project.entity.User> listUsers = service.listUsers();  
 @Deprecated  
 User.UserBuilder users = User.*withDefaultPasswordEncoder*();  
 InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();  
 for (int i = 0; i < service.listUsers().size(); i++)  
 manager.createUser(users  
 .username(listUsers.get(i).getUsername())  
 .password(listUsers.get(i).getPassword())  
 .roles(listUsers.get(i).getRights()).build());  
 return manager;  
  
 }  
}

MvcConfig

package com.project.security;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
  
@Configuration  
public class MvcConfig implements WebMvcConfigurer {  
  
 public void addViewControllers(ViewControllerRegistry registry) {  
 registry.addViewController("/home").setViewName("home");  
 registry.addViewController("/").setViewName("home");  
 registry.addViewController("/login").setViewName("login");  
 }  
  
}

UserService

package com.project;  
  
import java.util.List;  
  
import com.project.entity.User;  
import com.project.entity.UserCredits;  
import com.project.entity.UserDeposits;  
import com.project.repository.UserCreditsRepository;  
import com.project.repository.UserDepositsRepository;  
import com.project.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
@Service  
@Transactional  
public class UserService {  
  
 @Autowired  
 private UserRepository repository;  
  
 public List<User> listUsers() {  
 return repository.findAll();  
 }  
  
 public void save(User user) {  
 repository.save(user);  
 }  
  
 public User get(long id) {  
 return repository.getOne(id);  
 }  
  
 public void delete(long id) {  
 repository.deleteById(id);  
 }  
  
 @Autowired  
 UserCreditsRepository ucRepository;  
  
 public List<UserCredits> listUsersCredits() {  
 return ucRepository.findAll();  
 }  
  
 public void saveUserCredits(UserCredits userCredits) {  
 ucRepository.save(userCredits);  
 }  
  
 public UserCredits getUserCredits(long id) {  
 return ucRepository.getOne(id);  
 }  
  
 public void deleteUserCredits(long id) {  
 ucRepository.deleteById(id);  
 }  
  
 @Autowired  
 UserDepositsRepository udRepository;  
  
 public List<UserDeposits> listUserDeposits() {  
 return udRepository.findAll();  
 }  
  
 public void saveUserDeposits(UserDeposits userDeposits) {  
 udRepository.save(userDeposits);  
 }  
  
 public UserDeposits getUserDeposits(long id) {  
 return udRepository.getOne(id);  
 }  
  
 public void deleteUserDeposits(long id) {  
 udRepository.deleteById(id);  
 }  
  
}

home.html

<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org" lang="en">  
<head>  
 <meta charset="utf-8"/>  
 <link rel="stylesheet" type="text/css" href="/../static/css/style.css" th:href="@{/css/style.css}"/>  
 <link rel="stylesheet" type="text/css"  
 href="C:/Users/nemoCapt/Desktop/Uni/Прога/StatusBank/src/main/resources/static/css/style.css"/>  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">  
 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.0.7/css/all.css">  
 <title>User manager</title>  
</head>  
<body>  
<div class="back">  
 <form action="new">  
 <input class="buttonAdd" type="submit" value="&#43;"/>  
 <div class="invisibleField">Add</div>  
 </form>  
 <form action="/showcredits">  
 <button class="buttonCredits">  
 <i class="far fa-credit-card"></i>  
 </button>  
 <div class="invisibleField" style="margin-top:235px">Credits</div>  
 </form>  
 <form action="/showdeposits">  
 <button class="buttonDeposits">  
 <i class="fas fa-university"></i>  
 </button>  
 <div class="invisibleField" style="margin-top:370px">Deposits</div>  
 </form>  
 <form th:action="@{/logout}" method="post">  
 <button class="buttonLogout">  
 <i class="fas fa-sign-out-alt"></i>  
 </button>  
 <div class="invisibleField" style="margin-top: 505px">Log Out</div>  
 </form>  
 <div class="innerDiv" align="center">  
  
 <div class="miniHeader">  
 <h1 class="topLabel">User List</h1>  
 </div>  
 <div class="miniDiv">  
 </div>  
 <div class="miniFooter" style="position: relative;">  
 <table cellpadding="10" align="center">  
 <thead>  
 <tr style="background: #23272A; color: white;">  
 <th>User ID</th>  
 <th>Username</th>  
 <th>Password</th>  
 <th>Balance</th>  
 <th>Balance number</th>  
 <th>Rights</th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr align="center" th:each="user : ${listUsers}">  
 <td style="color: white" th:text="${user.id}">User ID</td>  
 <td>  
 <form action="#" th:action="@{'/personal/' + ${user.id}}" method="post">  
 <button class="buttonUser">  
 <i class="fas fa-user" style="margin-left: -110%;"></i>  
 </button>  
 </form>  
 <a th:text="${user.username}"></a>  
 </td>  
 <td th:text="${user.password}"></td>  
 <td th:text="${user.balance}"></td>  
 <td th:text="${user.balancenumber}"></td>  
 <td th:text="${user.rights}"></td>  
 <td>  
 <a style="color: white" th:href="@{'/edit/' + ${user.id}}">Edit</a>  
 <a style="color: #c0001b" onclick="return confirm('Delete user?')"  
 th:href="@{'/delete/' + ${user.id}}">Delete</a>  
 </td>  
 </tr>  
 </tbody>  
 </table>  
 <form action="/accounts">  
 <button class="buttonAccounts">  
 <i class="fas fa-info-circle"></i>  
 </button>  
 <div class="invisibleField"  
 style="position: absolute; bottom: 95px; right: 13px; font-size: 13px; padding-top: 12px;">Accounts  
 plan  
 </div>  
 </form>  
 </div>  
 </div>  
</div>  
</body>  
</html>

application.properties

spring.jpa.hibernate.ddl-auto=none  
spring.datasource.url=jdbc:mysql://localhost:3306/sales?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC  
spring.datasource.username=root  
spring.datasource.password=root  
logging.level.root=WARN

pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0   
 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.project</groupId>  
 <artifactId>ProductManager</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.1.3.RELEASE</version>  
 </parent>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>   
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
   
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>   
</project>