Варианты тем:

«Разработка АРМ бухгалтера по учету аренды помещение в бюджетном/государственном учреждении»

«Система управления бух учетом арендной платы помещений»

Введение

В различных отраслях экономики бухгалтерский учет имеет множество специфических особенностей. В зависимости от отрасли, в которой проводится учет, меняется и весь процесс работы бухгалтера.

С давних времен процесс учета сильно изменился, начиная от простого запоминания или записи на бумаге до работы с файлами на компьютере в настоящее время. Но само использование компьютера не избавляет от ручного труда. Для решения проблем работников бухгалтерии и ускорении их работы необходимо создать соответствующие инструменты, которые автоматизируют рабочий процесс.

В данной работе в качестве участка бухгалтерского учета, который будет подвержен автоматизации, будет выступать аренда помещений государственного учреждения.

Функционал:

1. Прием данных с помощью предоставленного удобного интерфейса, абстрагированного от таблиц, колонок и столбцов, как в Excel. Вместо этого – работа с сущностями (арендатор, договор, компания).
2. Автоматизация расчётов, связанных с арендой помещений (арендной платой).
3. Система уведомлений для контроля за сроками оплаты (просроченные, не просроченные)
4. Наглядное представление информации по арендаторам, договорам, счетам.
5. Гибкая настройка шаблонов, позволяющая использовать приложение без его привязки к конкретному учреждению.
6. Быстрое предоставление необходимой информации с помощью поиска
7. Архивация данных. Экспорт для переноса. Автоматический бэкап для защиты от потери.
8. Объединение счетов за оплату в единые ведомости и ордер. (без необходимости copy-paste из файла в файл).
9. Вывод данных в Excel: от простого счета для оплаты по договору до объединенных в единый мемориальный ордер документов.
10. Поддержка различных форматов для вывода данных. (.xsl – Excel 2003, xslx).
11. Быстрый доступ к истории оплаты каждого арендаторы

Подробнее о том, как приложение решает поставленные задачи и реализует функционал.

Рассмотреть процесс ручной работы и описать преимущества, которые даст автоматизация.

Регистрация и учет договоров по арендной плате в простейшем виде включает в себя перенос данных, полученных с утвержденного арендодателем договора в какой-нибудь электронный файл, к примеру, формата Excel. Для каждого нового договора вводится информация об арендаторе, о помещении и дате аренды. Также, договор содержит ячейки для расчета платы за саму аренду помещения, оборудования и коммунальные услуги. Этот расчет производится каждый месяц для каждого договора. По итогу, ежемесячные расчеты копируются в отдельные ведомости – накопительную, мемориальную,….

Все эти ведомости и счета обладают большим недостатком – они не взаимосвязаны друг с другом, поэтому чтобы сделать 2 ведомости, надо вручную копировать или набирать вновь все значения.  
А также для создания новых договоров, но с повторяющими арендаторами или помещениями – также приходится набирать все заново.

Автоматизация даст…  
Создание автоматизированного рабочего места позволяет не только уменьшить трудозатраты на рабочий процесс и ускорить выполнение задач, связанных с расчетом аренды, но и предоставит цельную и логически структурированную базу данных вместо рассредоточенных по разным документам договоров, расчетов, счетов и ведомостей.  
Это позволяет эффективно и быстро оценивать имеющиеся данные, и, при необходимости, сразу выдавать информацию об аренде или выводить её на печать.  
Так как сущности связаны друг с другом и сохранены в базе данных, то благодаря предоставленному программой интерфейсу, создав единожды экземпляр арендатора при заключении первого договора нет необходимости создавать его заново при заключении каждого последующего договора. Достаточно выбрать ранее созданного арендатора в списке арендаторов. Аналогичная ситуация обстоит и с любыми другими сущностями, участвующими в аренде. Например, здание аренды и помещение.

Целесообразность написания десктоп приложения.

Исходя из требований к проекту, было принято решение о разработке декстоп приложения. Основное требование, которое привело к этому решению, было возможность доступа к приложению вне зависимости от наличия интернет-соединения или веб-обозревателя на рабочей станции. Также, в виду того, что приложение задумано однопользовательским, его размещение в сети интернет не обусловлено целями и не соответствует решению задач проекта.

Нецелесообразность использования Excel и макросов VBA.

Целесообразность использования инструментов java javafx fxml, библиотеки с excel.

Для разработки автоматизированного рабочего места выбран язык программирования Java. Огромная популярность языка, и, соответственно, большое сообщество обеспечивает возможность быстро найти ответ на любые всплывающие в процессе разработки вопросы для разрешения проблем.

Библиотека JavaFX предлагает все необходимые инструменты для построения интерфейсов на языке Java, и в то же время не требует слишком много затрат времени на изучение.   
Вместе с ней используется fxml – специальный синтаксис для написания визуальной части javafx приложения с xml-подобной структурой. Таким образом происходит разделение представления от реализации, что облегчает понимание исходного кода.

Целесообразность использования системы управления базами данных h2.

Выбор СУБД во многом определяется спецификой данных, размерами приложения и требованиями к скорости доступа к данным. Приложение имеет небольшой размер и потоки данных, не требует высокую скорость обработки данных. Открытая кросплатформенная СУБД h2, написанная на Java, имеет маленький вес, около 1мБ, т.к. не использует внешние библиотеки, и написана нативно. Также, в h2 применяется стандартный SQL-синтаксис для осуществления запросов, а грамотно написанная, подробная и понятная документация помогает быстро получать ответы на возникающие вопросы.

Согласно результатам сравнения с помощью тестов производительности, H2 и некоторых других популярных СУБД (в частности, HSQLDB, PostgreSQL и MySQL), H2 в целом более производительна, чем указанные СУБД. То есть в скорости обработки данных она не уступает другим СУБД, а встроенная версия h2, которая используется в данном приложении, не требует создания сервера и наличия интернет-соединения, отсутствие которых прописано в требованиях к проекту.

Потоки данных

Управляющее воздействие: стандарт «Аренда», налоговый кодекс, земельный кодекс, постановление, инструкции и законы.

Механизмы: экономист

Выходные данные: готовые Excel-файлы, сформированные по определенному шаблону

Входные данные: договор с арендатором

Рабочий процесс пользователя и программы

Алгоритм учета аренды помещений:

1. Экономист получает подписанный договор с арендатором.
2. Создание нового арендатора в разделе «Список арендаторов»
3. //Создание сущности корпуса в разделе «Список корпусов»
4. Выбираем корпус/помещение
   1. Помещение отсутствует в списке - создание сущности корпуса в разделе «Список корпусов»
5. Внесение информации о договоре в программу
   1. Нажать кнопку «Новый договор» в разделе «Договоры»
      1. Выбрать арендатора
      2. Выбрать корпус
      3. Заполнить информацию об арендованном помещении

Создание нового договора осуществляется в отдельной вкладке/окне.  
На этом этапе заполняются все поля на основе сведений о договоре. В выпадающем списке, при помощи поиска, можно выбрать предварительно созданные для договора сущности **(\*)**.  
После создания договора, он появляется в разделе «Договоры»

В списке договоров можно перейти к подробной информации о конкретном договоре, а также произвести расчет арендной платы, коммунальных услуг, земельного налога и аренды оборудования.  
Расчет производится помесячно, для каждого месяца формируется свой Excel-документ для распечатки счёта.  
  
При оплате счёта арендатором, экономист помечает соответствующий месяц погашенным.  
В случае неуплаты, месяц помечается как просроченный, и идет автоматический расчет пени.  
За некоторое количество дней до окончания срока аренды срабатывает система уведомлений.

Для каждого месяца есть возможность сформировать обобщающие документы – накопительную ведомость, мемориальный ордер и т.д.   
Они охватывают различные данные по всем договорам за весь месяц.

В профиле для каждого арендатора есть соответствующая для него история договоров, его счетов и оплат. Таким образом, при необходимости, можно быстро проверить арендатора либо выдать по нему всю информацию.

**(\*)** Для удобного и быстрого занесения договоров в БД мы предварительно создаем отдельные сущности – отдельные компоненты договора. Например, арендатор, корпус.

Преимущество такого подхода состоит в возможности использовать повторяющиеся сущности от одного договора к другому, без необходимости их повторного ввода.  
Занеся однажды арендатора в список арендаторов, нам не придется заново вводить данные об этом арендаторе при последующем заключении новых договоров с этим арендатором.  
Это позволяет ускорить работу в дальнейшем.

Базовая теоретико-аналитическая часть

Анализ аналогов

**1С: Предприятие**

1С:Предприятие 8.3.13.1513

Система программ «1С:Предприятие 8» включает в себя платформу и прикладные решения, разработанные на ее основе, для автоматизации деятельности организаций и частных лиц. Сама платформа не является программным продуктом для использования конечными пользователями, которые обычно работают с одним из многих прикладных решений (конфигураций), разработанных на данной платформе. Такой подход позволяет автоматизировать различные виды деятельности, используя единую технологическую платформу.

Гибкость платформы позволяет применять 1С:Предприятие 8 в самых разнообразных областях:

* автоматизация производственных и торговых предприятий, бюджетных и финансовых организаций, предприятий сферы обслуживания и т.д.
* поддержка оперативного управления предприятием;
* автоматизация организационной и хозяйственной деятельности;
* ведение бухгалтерского учета с несколькими планами счетов и произвольными измерениями учета, регламентированная отчетность;
* широкие возможности для управленческого учета и построения аналитической отчетности, поддержка многовалютного учета;
* решение задач планирования, бюджетирования и финансового анализа;
* расчет зарплаты и управление персоналом;
* другие области применения.

**1С:Аренда и управление недвижимостью**

Продукт предназначен для собственников коммерческой недвижимости, управляющих компаний, девелоперских организаций. Продукт позволяет повысить эффективность бизнеса компаний за счет автоматизации процессов по ведению реестрового учета объектов недвижимости, управлению договорами аренды и расчетами с арендаторами, эксплуатации объектов недвижимости. Программа обеспечивает решение задач бухгалтерского, управленческого, юридического и административного учета.

Программа позволяет эффективно управлять недвижимостью разных типов: торговыми и офисными центрами, рынками, выставочными площадями, складами и т.п.

Продукт разработан на последней версии технологической платформы «1С:Предприятие 8» и выпускается в двух вариантах- на основе типовых конфигураций «Бухгалтерия предприятия» и «Управление производственным предприятием».

Учет аренды, подобный тому, который реализуется данным проектом, осуществляет конфигурация "Аренда и управление недвижимостью" для "Бухгалтерия государственного учреждения".

Доистоинства:

1. Мощный функционал
2. Гибкость в настройке
3. Максимальное соответствие стандартам
4. Удобный интерфейс
5. Постоянное развитие и поддержка
6. Отечественный продукт

Исходя из анализа конфигурации аренды и системы программ 1С в целом, можно выделить следующие недостатки, отталкивающие конечных пользователей:

1. Высокая сложность программного продукта. Высокая сложность затрудняет понимание пользователя и увеличивает порог вхождения для использования программы.
2. Стоимость ПО. 1С – коммерческая организация, за использование продуктов которой необходимо платить деньги. В виду невозможности некоторых организаций оплачивать программное обеспечение и тенденции к использованию бесплатных аналогов, это уводит от программного продукта множество потенциальных клиентов. В 1С же весьма дорогие расценки, и надо платить как за саму платформу, так и за большинство конфигураций, которые захочется использовать.
3. Нагроможденность. Этот недостаток присутствует как в функциональном плане в виде довольно большого скопления функций и возможностей в одном месте, так и в визуальном плане – интерфейс непростой.
4. Отсутствие узкой специализации. Система программ «1С: Предприятие» рассчитана на выполнение большого количества задач. В данной конфигурации «1С:Аренда и управление недвижимостью» есть огромное количество инструментов, которые не связаны напрямую с арендой, и которые сильно влияют на способность пользователей учиться пользоваться программой. Слишком большое количество нюансов в управлении арендой в том числе.
5. Долгая настройка конфигураций. Для подготовки рабочего окружения в первый раз требуется значительное количество времени, прежде чем можно будет приступить к работе.
6. Невозможность печатать отчёты в формате Excel.

Достоинства отдельной программы:

1. Узкая специализация. Возможность сосредоточиться на конкретной функциональности – аренде помещений. Это позволит избежать возможного испуга и недопониманий со стороны пользователя, который рассчитывает получить инструменты для решения своих задач, а по итогу получает сборник из множества намешанных возможностей.
2. Простота в обучении и легкий интерфейс, понятный и удобный для простых пользователей. Использование присущих предметной области терминов, минимализм в управлении и функционале отвергает возможную путанницу в работе приложения.
3. Распечатка в формат Excel. Для клиентов, которым исходя из требований к работе необходимо создавать и фиксировать Excel-документы по аренде, этот пункт будет весьма полезным.
4. Маленький размер и высокая производительность. В виду небольшого размера и отсутствия связи с сетью, и использованию встроенной БД, программный продукт устанавливается, запускается, и выдает полезный результат гораздо быстрей аналогов.
5. Бесплатность. Продукт не требует оплаты для пользования

БД

Надо ли хранить данные в «Коммунальные услуги» типа «Начислено», если мы можем рассчитать их без хранения.

Счет скорей всего должен выступать в качестве таблицы БД, т.к. для его составления нужны реквизиты.  
А вот накопительная ведомость, мемориальный ордер и ведомость начисления – вычисляются исходя из множества уже существующих таблиц. То есть они просто объединяют в себе данные других таблиц и вычисление происходит на их основе, поэтому скорей всего не нужны в виде отдельных сущностей.  
Дебет и кредит тот же, вычисляется так же исходя из других таблиц. Имеет ли смысл выделять их в новые таблицы?  
Можно конечно выделить дебет и кредит в таблицу Счет, и сделать из нее просто общий сборник данных для всего экспорта во все счета и ведомости.

Вопрос – выводить счет на основе арендатора или договора? Где будет кнопка вывода счета?

Сотрудник напрямую не контактирует с арендатором – только с договором.  
И сущность арендатора позволяет посмотреть список его договором. Но не сформировать счет.  
Но можно выбрать его договор, и уже исходя из конкретного договора – вывести счет.  
У одного счета – лишь 1 договора. У 1 договора – множество счетов(за каждый месяц)

Настроено подключение приложения к СУБД h2 во встроенном режиме. В зависимости проекта папки lib установлен JDBC драйвер, обеспечивающий стандартизированный интерфейс для взаимодействия приложения к СУБД.  
Возможно в скором времени использование hibernate для более быстрого и удобного доступа к БД.

Для использования методов драйвера был выделен отдельный класс Database, который включает в себя логику работы с СУБД, а именно – подключение к ней.  
Планируется использовать порождающий паттерн проектирования «Одиночка», чтобы подключение к БД происходило единожды во всем приложении.

Информация для подключения к СУБД вынесена в отдельный конфигурационный database.properties файл, чтобы отделить логику работы программы от входящих данных, а также для удобного доступа к конфигурации проекта, без необходимости изменять код приложения.

На первое время для теста реализованы методы для добавления и получения простой выборки с БД для проверки работоспособности подключения.  
Но кроме этого, методы сделаны статическими и выделены в модель приложения – класс RenterModel, с задумкой на проектирование приложения по архитектуре MVC.  
Класс RenterModel будет охватывать всех методы для работы с данными арендаторов.

Логика работы вывода списка по нажатию на кнопку «Показать» выделена в метод UpdateListRenters, который добавлен также в логику для «Добавить» и «Удалить» арендаторов.  
UpdateListRenters занимается обновлением списка арендаторов для пользователя, через получение данных из RenterModel и удаление старых.

Созданы директории для контроллеров, моделей и представлений. Предполагается использовать архитектуру MVC.

Для конструирования меню и сайдбара написаны контроллеры MenuController и SidebarController, который содержат в себе порядок построения элементов для этих компонентов GUI.   
Предполагается заменить логику конструирования в файлы fxml.  
(Конец 23.10.2018)

ПТ - 26.10.2018

Осуществлен перенос элементов интерфейса из RenterController в отдельный файл представления renter.fxml. Таким образом контроллер стал более чистым и удобным для написания логики работы с сущностями арендаторов. Но пришлось изменить методы – их параметры и статичность, также добавить сеттер для добавления ссылки на БД к объекту класса.  
В виду вышеперечисленных изменений, соответственно, изменен вызов контроллера из SidebarController, так как в текущей реализации нам необходимо создавать объект класса RenterController, в отличии от предыдущей статической версии работы.

Из главного класса EconomistWorkstation удалена часть, ответственная за отображение страницы Renter, чтобы в скором времени, при необходимости, создать настоящую стартовую страницу приложения.

В планах расширить логику и представления для новых полей таблицы Renter, чтобы получить полностью сформированную сущность арендаторов.  
Таким образом логика (функционал создания арендаторов) будет завершена.  
Текущего уровня представления будет достаточно. Потом интерфейс будет улучшен.

Также надо сделать вывод арендаторов в списке арендаторов без кнопки показать.  
Потом сделать конкретную страницу об арендаторе для операций CRUD.

Ускорься - пора сдавать 2 главы первые, теор часть и про БД.   
Исследуй аналоги, алгоритм действий и расписывай. Потом прогу допишешь.

ПН 29.10.18

Убрана кнопка «Показать» и обеспечен вывод арендаторов.

ВТ 30.10.18

Так. Я перенес логику создания формы в отдельный контроллер RenterFormController и представление RenterForm для более удобной работы с кодом за счет декомпозиции сложной задачи и в виду некоторых сложностей, которые препятствовали объединению формы и списка арендаторов, так как форма предполагалась быть в отдельном окне.  
Объединить в теории можно, но будет большая путаница и нагромождение условных операторов.

Для каждого арендатора добавлена кнопка удаления с соответствующей обработкой.

Реализован порождающий паттерн проектирования «Одиночка» (Singleton) для единого доступа к базе данных из любого класса, без необходимости передавать объект базы данных от класса к классу через параметры.

СР 31.10.18

Создал новый класс (структуру) для хранения сущностей типа Renter.  
Добавлены поля для создания полноценной сущности арендатора Renter (не только имя)  
В RenterModel добавлена логика для добавления сущности Renter в БД, а также удаление её по id.  
RenterController. RenterFormController связывается с полями Renter.fxml.

ПТ 02.11.18

Renter.toString метод для отображения имен в списке арендаторов вместо идентификаторов. Используется в ComboBox создания договора.

Удалены все обращения и перессылки объекта БД из класса в класс.  
Теперь объект БД находится лишь в классах Model в виде статического свойства. Инициализатор {} не используется в виду отсутствия необходимости, он лишь продублирует и увеличит код, не обеспечивая лучшую читаемость, т.к. в классах Model, пока что, лишь 1 статическое свойство, которому сразу же при компиляции и присваивается значение.

Добавлен контроллер и представление для зданий Building.

Ожидается создание таблицы БД, модели, связи её с контроллером, а также добавление контроллера формы создания зданий и его представления.

ВС 04.11.18

Добавлена таблица Building с помощью h2 в БД и в памятку запросов query.

Добавлена сущность Building для оперирования зданиями как отдельным типом данных.  
Реализована BuildingModel для работы с данными.  
Но тут появилась идея, в виду большого количества повторных кусков кода – а что если сделать для всех сущностей один интерфейс, один базовый класс, и таким образом минимизировать количество повторяющихся частей кода, и использовать полиморфизм, возможно, с каким либо паттернами фабрик.

Контроллер зданий связан с контроллером сайдбара – теперь можно заходить на страницу зданий с помощью боковой панели.

Добавлено представление BuildingForm.fxml для создания зданий.

Запланировано: создать контроллер для представления создания зданий. Подумать о вышеозвученной идее сейчас, либо разобрать её после ещё созданий одной сущности.

ПН 05.11.18

Разработан BuildingFormController для обработки представления формы BuildingForm.

Добавлен метод в BuildingController для отображения формы.

Организован вывод списка зданий в форме создания договора

Создан метод получения одного арендатора getRenter

Запланировано: сделать полноценное редактирование профиля, кнопки для возврата назад или типо того. Перенести создание договора в форму. Начать делать форму для договора полноценную с оплатой. Сделать профиль для арендатора по нажатию на кнопку, а получение данных из модели уже перенести в новое окно, ибо тут ни к чему получать арендатора, и еще и передавать потом его в качестве аргумента.

ВТ 06.11.18

Создание представления для профиля RenterProfile, его контроллера и связи между ними.

Добавлен метод для обновления арендатора.

Подключение формы для создания арендаторов к странице профиля, чтобы можно было обновить любого арендатора. И, чтобы не плодить одинаковые формы – использована форма для создания. Но реализовано криво все, поэтому надо будет доводить до ума. Но работает.

СР 07.11.18

Создание модели для договоров.

Вынесение из Contract.fxml в представление для создания договора

Создание логики и обработка представления в ContractFormController

Фиксация изменений в ContractController

Запланировано: доработать модель договоров. Создать простую таблицу договоров.

Приступить к рефакторингу кода и профилю.

ПТ 09.11.18

Добавлена таблица для договоров, примитивная.

Создана сущность для договора.

Модель договора доработана. Теперь правильно работает с таблицей договоров из БД.

Доработан контроллер формы для создания договоров. Реализовано создание договора через форму.

ПН 12.11.18

Добавлено представление и контроллер для профиля договора. Но столкнулся с тем, как объединить все расчеты по аренде и оплаты? Нужно выделить новую сущность, определяющую один месяц для конкретного договора. Назовем её Month, т.к. ничего более обобщенного в голову не лезет. Расчет? Счет? Это все более узкие направления. Коммуналка, аренда помещения? Тоже. Более обобщенное, которое захватывает все вышеперечисленное – Month.

Также столкнулся с проблемой запросов. Можно использовать несколько готовых методов для каждой модели, чтобы выделить данные в профиль договора – модель договора, арендатора и здания.  
Иной путь – сделать лишь 1 запрос, но придется в методе для получения договора расширить запрос с нескольких таблиц сразу.

Воспользуемся пока готовыми запросами – тремя.

И для полноценного создания профиля договора – нужен список месяцев.

Итак, нужна сущность Month, представление и контроллер. А также модель Month, работающая с таблицами аренды помещений и коммуналкой.

Добавлен метод в модель зданий для получения одного здания.

Важно: в поисках лучшего решения для оптимизации таблиц. Мы не можем выделить таблицы «Аренда» и «Комм. Услуги» в отдельные сущности, как таблицы «Арендаторы» и «Здание», т.к. предыдущие не могут «жить» вне договора, тогда как последние – могут. Но так как нужно выделить что-то, что будет сущностью, для работы с ними обоими – создадим Month, который будет содержать в себе и аренду и комуналку, во избежании связи 1 к 1.  
Конечно можно оставить и без месяца – просто будут 2 сущности, 2 таблицы месяцев в профиле, 2 формы для работой с каждым типом расчетов.

Или сущность Month будет работать просто с разными расчетами в приложении, её модель будет работать с разными таблицами из БД. Например, Month->GetRent.   
Хоть и можно объединить все в 1 таблицу Month, но декомпозиция облегчит понимание.

Добавлена сущность Rent

13.11.18

Создано представление для расчета ежемесячных платежей.

14.11.18

Пока что для облегчения задачи и комуналка и арендная плата обьединены в 1 сущность Month. Потому что пока что не могу решить, а нужны ли отдельные сущности. Ведь это придется делать запросы из 2 таблиц.

Добавлена модель Month.

В модель контрактов при создании контракта реализован базовое добавление 12 месяцев.

Осуществлен вывод списка месяцев в профиле договора ContractProfileController.

15.11.18

Пофикшены баги с выводом. Работоспособная версия

Реализовано чтение месяца из договора

Реализовано полное обновление месяца через форму.

Использование Datepicker для создания догвора.

Получение разницы количества месяцев договора. Динамическое добавление месяцев для контракта, в зависимости от дат.

18.11.18

Исправлено проставление первой и последней даты для вынесения счета за месячную оплаты.

Добавлена функция продления аренды в независимо от количества месяцев, с изменением последнего месяца.

Подготовлено представление для расчетов аренды.

19.11.18

В сущность и модели для Месяца добавлены новые поля – индексы.

Обработчики TextField для вывода итогов вычисления расчетов за оплату на Label’ы.

Создание кнопок для вывод данных о договоре – счета и расчетов.

Сохраняю ворд файл и запросы к БД обновленные.

Пока что использую принцип «Не сохраняй в БД то, что можно вычислить на основании имеющихся в БД данных». Поэтому не сохраняю итоги расчетов и не добавляю итоговое поля в сущность Month, она уже и так раздута.

Надо: оптимизировать обработчики изменения TextField – просто метод выделить отедьный в этом же классе, который будет сам навешивать обработчик в зависимости от 2 аргументов – объекта, на который навешивается, и того, который будет участвовать в умножении.  
Таким образом очень сократится код при инициализации.

20.11.18

Обновлены доки и модель БД