Техническое задание

Ювелирная мастерская

Список ответственных исполнителей

Организация, должность	**Фамилия, имя, отчество**
УО «БГУИР», Студент	Юревич А. А.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Администрирование данных — управление информационными ресурсами, включая планирование базы данных, разработку и внедрение стандартов, определение ограничений и процедур, а также концептуальное и логическое проектирование баз данных.

Администратор данных отвечает за корпоративные информационные ресурсы, включая и некомпьютеризированные данные. На практике это часто связано с управлением данными, которые являются совместно используемым ресурсом для различных пользователей и прикладных программ организации. В одних случаях администрирование данных может представлять собой отдельную функциональную задачу, а в других — совмещаться с администрированием базы данных.

В настоящее время при обдумывании стратегии планирования информационной системы все больший акцент делается на важности администрирования данных. Организации все в большей и большей степени склонны уделять внимание значению данных, используемых или собранных в их информационной системе, как средству достижения более высокой конкурентоспособности. В результате возникает обязательное требование слияния стратегии построения информационных систем с бизнес-стратегиями организации. Это позволяет создать организацию с более гибкой структурой, способную адаптироваться к резким изменениям, имеющую более творческую и инновационную внутреннюю среду, обеспечивающую эффективную перестройку бизнес-процессов в случае необходимости. Упомянутый перенос акцентов означает, что администрирование данных во все большей мере должен понимать идеологию развития не только информационных систем, но и бизнес-процессов, и играть ключевую роль в разработке стратегии развития информационной системы, поддерживая её соответствие деловым стратегиям организации.

В рамках данного проекта предлагается разработать непосредственно систему администрирования данных на тему «Ювелирная мастерская».

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Десктопные приложения – это программы, логика работы которых требует наличия оператора (человека работающего с программой), содержащие в себе всю полную функциональность и способные работать отдельно на любой машине изолированно от других приложений. Microsoft Word, Excel, Блокнот, однопользовательские игры – всё это примеры десктопных приложений. Для их работы необходимы лишь достаточные аппаратные ресурсы компьютера, само приложение и набор библиотек, содержащих функции для работы с приложением.

Десктопные приложения могут быть также и многопользовательскими. Например, редактор файлов который в зависимости от логина и пароля, введенных при запуске, будет давать доступ к различным файлам. И программа и файлы находятся на одном компьютере, просто производится локальное разграничение доступа для разных пользователей.

2 ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

2.1 Архитектурные паттерны

Паттерн проектирования Repository

«Repository» — один из наиболее часто используемых паттернов проектирования при работе с базами данных (БД). Он позволяет отделить программную логику, работающую непосредственно с БД, от всей остальной программы выступая посредником меду ними с помощью интерфейса во многом схожего с коллекциями.

Преимущество использования паттерна «Repository»:

- Отделение программной логики, работающей непосредственно с БД от всей остальной программы. Если требуется изменить логику работы с БД или даже тип БД (например, перенос БД с MS SQL Server на MySQL), то внесение изменений осуществляется локально и централизованно. Нет необходимости вносить многочисленные правки по всей программе с вероятностью что-либо пропустить и тем самым спровоцировать ошибку в работе программы.
- С помощью «Repository» можно значительно упростить и алгоритмы по работе с БД в остальной программе. Чаще всего для выполнения какой-либо операции с БД достаточно вызвать один из методов класса, реализующего этот паттерн.

2.2 Инструменты разработки

IntelliJ IDEA

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Первая версия появилась в январе 2001 года и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными свободными инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit. В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке Ruby.

Начиная с версии 9.0, среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0, в ней реализована полная поддержка Java SE, Kotlin, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. В редакции Ultimate Edition, доступной под коммерческой лицензией, реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков.

Git

Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день его поддерживает Джунио Хамано.

Примерами проектов, использующих Git, являются Ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt и некоторые дистрибутивы Linux.

Программа является свободной и выпущена под лицензией GNU GPL 2.

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

На проекте используется в связке с GitHub (GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки).

2.3 Язык программирования

Java

Java — сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). В настоящее время проект принадлежит OpenSource и распространяется по лицензии GPL.

В OpenJDK вносят вклад крупные компании, такие как — Oracle, RedHat, IBM, Google, JetBrains. Так же на основе OpenJDK эти компании разрабатывают свои сборки JDK. Как утверждает компания Oracle — отличия между OpenJDK и OracleJDK практически отсутствуют за исключением лицензии, отрисовки шрифтов в Swing и некоторых библиотек, на которые лицензия GPL не распространяется.

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре с помощью виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года. На 2019 год Java — один из самых популярных языков программирования.

2.4 База данных

MySQL

На сегодняшний день СУБД MySQL является одной из самых известных, надежных и быстрых из всего семейства существующих СУБД. Одной из причин являются правила ее распространения — за нее не надо платить деньги и распространяется она вместе со своими исходными текстами, другая причина – это то, что MySQL относительно быстрая СУБД.

MySQL написан под десятки видов операционных систем. Это и FreeBSD, OpenBSD, MacOS, OS/2, SunOS, Win9x/00/NT и Linux. Сегодня MySQL особенно распространена на платформах Linux и Windows. Причем на последней встречается гораздо реже.

Принцип работы СУБД MySQL аналогичен принципу работы любой СУБД, использующей SQL (Structured Query Language, язык структурированных запросов) в качестве командного языка для создания/удаления баз данных, таблиц, для пополнения таблиц данными, для осуществления выборки данных.

Приведем очень краткий список возможностей MySQL:

- ACID-совместимые транзакции;
- кроссплатформенную поддержку;
- репликации;
- поддержку огромных таблиц и баз данных;
- полнотекстовый поиск;

- поддержку подзапросов;
- поддержку большинства требований синтаксиса SQL 92.
- представления;
- использование сохраненных процедур;
- триггеры.

3. ОБЩИЕ ПОЛОЖЕНИЯ

3.1. Назначение документа

В настоящем документе приводится полный набор требований системе, необходимых для реализации.

3.2. Цели создания системы

Создать приложение для управления процессами в ювелирной мастерской: прием заказов от клиентов, заказа материалов для изделий, оплата заказа клиентом.

3.3. Основные функциональные возможности системы

Приложение должно соответствовать следующим требованиям:

- возможность регистрации и авторизации пользователя;
- иметь дружественный, интуитивно понятный интерфейс;
- редактирование личных пользовательских данных;
- управление списком пользователей через интерфейс администратора;
- заказ новых материалов у поставщиков;
- проверка материалов, хранящихся на складе;
- проверка актуального баланса мастерской;
- количество заказов клиента за последний год по каждому месяцу;
- заработная плата мастера за последний год по каждому месяцу;
- принятие на работу нового мастера;
- оформление заказа клиентом;
- проверка статуса заказа;
- отображение информации о заказе определенного клиента или мастера через интерфейс администратора;
- оплата заказа клиентом;

3.4. Плановые сроки окончания работ

Проект планируется сдать до 20 декабря 2019 года.

3.5. Использование технического задания

Для составления общей картины проекта лабораторных работ.

4 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

4.1 Диаграмма базы данных

Результат предполагаемого каркаса базы данных, в виде диаграммы представлен на рисунке 1.

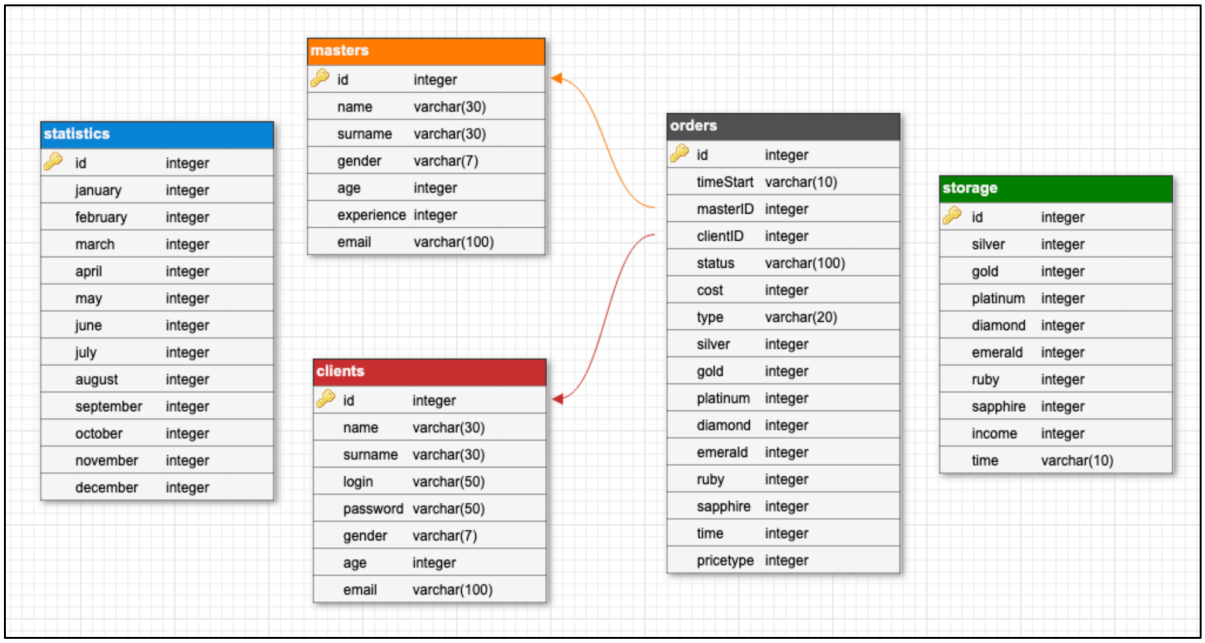

Рисунок 1. Схема базы данных

4.2 Описание вариантов использования

4.2.1 ВИ «Оформить заказ»

Описание ВИ:
Пользователь должен иметь возможность осуществить заказ с выбором изделия и материалов, которые будут использоваться для его изготовления.

Предусловия:

- Пользователь авторизировался как клиент.

Основной поток действий:

- Пользователь нажимает кнопку «Сделать заказ»;
- В появившемся окне пользователь выбирает тип продукта из списка предлагаемых изделий и отмечает материалы, которые должны использоваться для изготовления ювелирного украшения;
- Пользователь нажимает кнопку «Показать заказ»;
- После отображения окна подтверждения, где пользователь видит тип

ювелирного изделия, материалы, которые будут использоваться для его изготовления и суммы заказа, пользователь нажимает кнопку «Подтвердить заказ».

- Система отправляет заказ свободному мастеру и сохраняет информацию о заказе в базе данных.

4.2.2 ВИ «Редактировать личную информацию»

Описание ВИ:
Пользователь должен иметь возможность редактировать личную информацию, находящуюся в его личном кабинете.

Предусловия:

- Пользователь авторизировался как клиент.

Основной поток действий:

- Пользователь нажимает кнопку «Подробная информация»;
- В появившемся окне пользователь изменяет интересующие его данные;
- Пользователь нажимает кнопку «Изменить»;
- Система сохраняет все введенную информацию в базе данных.

4.2.3 ВИ «Принять на работу мастера»

Описание ВИ:
Администратор системы должен иметь возможность нанимать на работу новых мастеров и добавлять их в систему.

Предусловия:

- Пользователь авторизировался как администратор.

Основной поток действий:

- Пользователь кликает на пункт «Меню», находящийся в шапке;
- В появившемся списке пользователь выбирает пункт «Добавить мастера»;
- В появившемся окне пользователь заполняет личную информацию нового мастера;
- Пользователь нажимает кнопку «Добавить»;
- Система добавляет нового мастера в базу данных.

4.2.4 ВИ «Заказ новых материалов»

Описание ВИ:
Администратор системы должен иметь возможность заказывать материалы, необходимые для изготовления ювелирных изделий.

Предусловия:

- Пользователь авторизировался как администратор.

Основной поток действий:

- Пользователь нажимает кнопку «Заказать материалы»;
- В появившемся окне пользователь выбирает нужное количество материалов каждого вида;
- Пользователь нажимает кнопку «Подтвердить»;
- Система производит заказ материалов у поставщика, перевод ему деньги со счета мастерской и добавляет новые материалы в базу данных.

4.2.5 ВИ «Проверить динамику заработной платы мастера»

Описание ВИ:
Администратор системы должен иметь возможность просмотра статистики о заработной плате каждого мастера за последний год.

Предусловия:

- Пользователь авторизировался как администратор.

Основной поток действий:

- Пользователь нажимает кнопку «Мастера»;
- В появившемся списке пользователь выбирает интересующего его мастера;
- Пользователь нажимает кнопку «Статистика мастера»;
- Система отображает статистику заработной платы выбранного мастера по месяцам за последний год в виде диаграммы.

4.2.6 ВИ «Проверить количество заказов клиента»

Описание ВИ:
Администратор системы должен иметь возможность просмотра количества сделанных заказов каждым клиентов в течении последнего года.

Предусловия:

- Пользователь авторизировался как администратор.

Основной поток действий:

- Пользователь нажимает кнопку «Клиенты»;
- В появившемся списке пользователь выбирает интересующего его клиента;
- Пользователь нажимает кнопку «Статистика клиента»;
- Система отображает статистику количества заказов выбранного пользователя по месяцам за последний год в виде диаграммы.

4.2.7 ВИ «Проверить заказ»

Описание ВИ:
Администратор системы должен иметь возможность просмотра любого заказа, выполняемого в данный момент в мастерской.

Предусловия:

- Пользователь авторизировался как администратор.

Основной поток действий:

- Пользователь нажимает кнопку «Мастера»;
- В появившемся списке пользователь выбирает интересующего его мастера;
- Пользователь нажимает кнопку «Заказ»;
- Система отображает подробную информацию о заказе, который выполняет данный мастер.

Альтернативный поток действий:

- Пользователь нажимает кнопку «Клиенты»;
- В появившемся списке пользователь выбирает интересующего его клиента;
- Пользователь нажимает кнопку «Заказ»;
- Система отображает подробную информацию о заказе, который сделал данный клиент.

4.3 Система ролей

Рассмотрим подробнее возможности пользователей приложения. Всех пользователей можно разделить на 2 категории:

- Обычный авторизированный пользователь – клиент ювелирной

мастерской, имеющий следующие операции: сделать заказ, редактировать свою личную информацию, проверить статус выполнения заказа, оплатить заказ, после его получения.

- Администратор – пользователь, имеющий возможность управлять всеми процессами мастерской. Доступные операции: прием на работу нового мастера, просмотр и редактирование личной информации мастеров и клиентов, просмотр информации о выполняемых заказах, проверка материалов, хранящихся на складе, заказ новых материалов у поставщика, проверка счета мастерской, просмотр статистики мастеров и клиентов.

5 НЕФУНКЦИОНВЛЬНЫЕ ТРЕБОВАНИЯ

5.1. Интерфейс пользователя

Требования к пользовательскому интерфейсу:

- Система должна отображать корректно интерфейс пользователя с разрешением от 1024x600 пикселей.

5.2. Отображение на различных ОС

Приложение должно быть кроссплатформенным.

5.3. Документация

Документация должна соответствовать следующим требованиям:

- Разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии;
- В состав сопровождающей документации должна входить UML-диаграммы.

5.4. Порядок разработки

Требования к процессу разработки программного продукта:

- Вся разработка должна сопровождаться гранулярными коммитами (на английском языке) в системе GIT. Коммиты должны быть понятны и содержать лишь основную суть изменений кода программы. При разработке должны быть активно использованы соответствующие ветки (весь процесс разработки не может производиться только в ветке master);
- Разрабатываемая программа должна соответствовать руководству по стилю кода.

6. ПЕРСПЕКТИВЫ РАЗВИТИЯ

В дальнейшем планируется улучшение интерфейса приложения, повышение удобства использования, предоставление интерфейса для мастеров, добавление функционала для клиентов, повышение вовлеченности пользователей.