МИНОБРНАУКИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«**САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО**»

Кафедра математического и компьютерного моделирования

Разработка информационной системы «Ветеринарная фармакологическая компания»

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ (ДИПЛОМНОЙ) РАБОТЫ

студентки 4 курса 451 группы

направления 38.03.05 - Бизнес -информатика
код и наименование направления

механико-математического факультета
наименование факультета, института, колледжа

Минаковой Дарьи Сергеевны
фамилия, имя, отчество

Научный руководитель
доцент, к.э.н. _____ Мельникова Ю.В
должность, уч. степень, уч. звание дата, подпись инициалы, фамилия

Заведующий кафедрой
зав.каф., д. ф-м. доцент _____ Блинков Ю.А.
должность, уч. степень, уч. звание дата, подпись инициалы, фамилия

Саратов 2020

Введение. Большинство современных компаний каждый день сталкиваются с новыми требованиями своих клиентов, заинтересованных сторон, работающих сотрудников и внешнего рынка. Эффективно работающие организации отличаются от иных тем, что они постоянно находятся в поиске инновационных методов работы и занимаются совершенствованием своих бизнес-процессов. Очень быстро растущий спрос на улучшение бизнес-деятельности заставляет современные компании обращаться к рынку информационных технологий.

Информационные технологии на данный момент - неотъемлемая часть современных реалий. Информационные системы, связанные с предоставлением и обработкой информации для всех уровней управления объектами, приобретают особую важность в общественной жизни. Сейчас уже невозможно представить какую-либо организацию, не применяющую компьютерных технологий. Это обусловлено также и тем, что государственные структуры требуют обязательных отчетов в электронном виде, следовательно, необходима систематизированная информация.

На сегодняшний день для динамичных, развивающихся предприятий стало просто необходимым использование разнообразных информационных систем. С их помощью организации имеют возможность автоматизировать работу и значительно сэкономить время. Самое трудное и важное выстроить единую систему, которая будет отвечать всем запросам всех пользователей, то есть сотрудников всей организации. В последнее время потребность в ветеринарной фармакологии возрастает, так как отношение к домашним питомцам меняется, люди стали более ответственно и эмоционально относиться к своим питомцам. В современном мире существует огромное количество компаний, которые производят лекарственные препараты для животных. Для обеспечения конкурентноспособности компании на рынке ветеринарных услуг для нее необходимо разработать информационную систему. Исходя из этого тема данной бакалаврской работы является актуальной.

Целью данной бакалаврской работы является разработка информационной системы, обеспечивающей работу бизнес-процессов предметной области «Ветеринарная фармакологическая компания».

Для достижения поставленной цели, необходимо будет решить несколько задач:

- провести анализ данной предметной области;
- построить диаграммы прецедентов, деятельности и последовательности для предметной области;
- построить ER-диаграмму для предметной области;
- разработать графический интерфейс.

Реализация поставленных целей и задач позволит получить информационную систему для ветеринарной фармакологической компании, которая будет решать актуальную задачу.

Цель дипломной работы будет считаться достигнутой в том случае, если после выполнения поставленных задач мы получим функционирующую информационную систему, предназначенную для эксплуатации в ветеринарной фармацевтической компании.

Выпускная квалификационная работа состоит из введения, пяти глав, заключения, списка используемых источников и приложений. В первой главе рассматриваются теоретичские основы информационных систем и их проектирования. Вторая глава посвящена ознакомлению с графическим языком для создания диаграмм UML. В данной главе были построены диаграммы прецедентов, последовательности и деятельности, был описан поток событий для диаграмм последовательности и деятельности. Третья глава посвящена средствам разработки для создание базы данных. В частности, рассматривается методология проектирования реалиционных баз данных, а также СУБД SQLite. Также в данной главе была создана ER-диаграмма базы данных и создана непосредственно сама база данных. В четвертой главе рассказывается о средствах разработки интерфейса- языке программирования Python и библиотеки PySide.

Основное содержание работы. Информационная система — система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию (ISO/IEC 2382:2015).

Информационная система предназначена для своевременного обеспечения надлежащих людей надлежащей информацией, то есть для удовлетворения конкретных информационных потребностей в рамках определенной предметной области, при этом результатом функционирования информационных систем является информационная продукция — документы, информационные массивы, базы данных и информационные услуги.

Свойствами информационных систем являются:

- любая информационная система может быть подвергнута анализу, построена и управляема на основе общих принципов построения сложных систем;
- при построении информационной системы необходимо использовать системный подход;
- информационная система является динамичной и развивающейся системой;
- информационную систему следует воспринимать как систему обработки информации, состоящую из компьютерных и телекоммуникационных устройств, реализованную на базе современных технологий;
- выходной продукцией информационной системы является информация, на основе которой принимаются решения или производятся автоматическое выполнение рутинных операций;
- участие человека зависит от сложности системы, типов и наборов данных, степени формализации решаемых задач.

В настоящее время известны две парадигмы проектирования информационных систем, использующие два различных подхода к описанию систем:

- структурная (процессно-ориентированная), основанная на каскадной (водопадной) модели жизненного цикла информационной системы;
- объектно-ориентированная, основанная на поэтапной модели жизненного цикла информационной системы.

Объектно-ориентированный подход базируется на концепциях:

- объекта и идентификатора объекта;
- атрибутов и методов;
- классов;
- иерархии и наследования классов.

Выбор обязательных характеристик ООБД основан на двух критериях: система ООБД должна быть объектно-ориентирована и должна представлять собой базу данных.

UML является языком широкого профиля, это - открытый стандарт, который использует графические обозначения для создания абстрактной модели системы, которая называется UML-моделью. UML был создан для определения, визуализации, проектирования, документирования программных систем. UML не является языком программирования. Для демонстрации функциональности и поведения системы были рассмотрены три диаграммы:

1. диаграмма вариантов использования;
2. диаграмма деятельности;
3. диаграмма последовательности.

Диаграммы прецедентов (вариантов использования) описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе. Важно понимать, что диаграммы вариантов использования не предназначены для отображения проекта и не могут описывать внутреннее устройство системы. Диаграммы вариантов использования предназначены для упрощения взаимодействия с будущими пользователями системы, с клиентами, и особенно пригодятся для определения необходимых характеристик системы. Другими словами, диаграммы вариантов использования говорят о том, что система должна делать, не указывая сами применяемые методы. Вариант использования описывает, с точки зрения действующего лица, группу действий в системе, которые приводят к конкретному результату. Варианты использования являются описаниями типичных взаимодействий между пользователями системы и самой системой. Они отображают внешний интерфейс системы и указывают форму того, что система должна сделать. Действующее лицо является внешним источником (не элементом системы), который взаимодействует с системой через вариант использования. Действующие лица могут быть как реальными людьми (например, пользователями системы), так и другими компьютерными системами или внешними событиями. Действующие лица представляют не физических людей или системы, а их роли. Это означает, что когда человек взаимодействует с системой различными способами (предполагая различные роли), он

отображается несколькими действующими лицами. Например, человек, работающий в службе поддержки и принимающий от клиентов заказы, будет отображаться в системе как «участник отдела поддержки» и «участник отдела продаж».

Диаграмма последовательностей (Sequence diagram) - диаграмма поведения, на которой показано взаимодействие и подчеркнута временная последовательность событий. В UML взаимодействие объектов понимается как обмен информацией между ними. При этом информация принимает вид сообщений. Кроме того, что сообщение несет какую-то информацию, оно некоторым образом также влияет на получателя. В этом плане язык UML полностью соответствует основным принципам ООП, в соответствии с которыми информационное взаимодействие между объектами сводится к отправке и приему сообщений.

Диаграмма последовательностей относится к диаграммам взаимодействия UML, описывающим поведенческие аспекты системы, но рассматривает взаимодействие объектов во времени. Другими словами, диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами. Диаграммы последовательностей необходимы для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями.

Диаграмма деятельности - диаграмма UML, выглядящая наиболее простой, поскольку напоминает привычную всем блок-схему. На самом же деле диаграмма активности - это нечто большее, чем блок-схема, хотя цели у них похожи: обе они отображают некий алгоритм. Диаграммы деятельности позволяют моделировать сложный жизненный цикл объекта, с переходами из одного состояния (деятельности) в другое. Но этот вид диаграмм может быть использован и для описания динамики совокупности объектов. Они применимы и для детализации некоторой конкретной операции, причем, предоставляют для этого больше возможностей, чем sклассическаяї блок-схема. Диаграммы деятельности описывают переход от одной деятельности к другой, в отличие от диаграмм взаимодействия, где акцент делается на перехо-

дах потока управления от объекта к объекту. На диаграмме деятельностей можно не только показать параллельно выполняемые действия, но и указать состояния объектов (так же, как и на представлениях конечных автоматов, о которых нам так много говорили в университетах), также есть возможность показывать распределение ролей и т. д.

На практике диаграммы деятельности применяются в основном двумя способами:

- Для моделирования процессов; В этом случае внимание фокусируется на деятельности с точки зрения «Actor», которые работают с системой.
- Для моделирования операций.

В этом случае диаграммы деятельности играют роль «продвинутых» блок-схем и применяются для подробного моделирования вычислений. На первое место при таком использовании выходят конструкции принятия решения, а также разделения и слияния потоков управления (синхронизации).

Реляционная база данных это совокупность взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Методология проектирования реляционных баз данных состоит из 7 этапов:

1. Определение основных типов сущностей, присутствующих в представлении данного пользователя о предметной области приложения. Документирование выделенных типов сущностей.
2. Определение важнейших типов связей, существующих между сущностями, выделенными на предыдущем этапе. Определение кардинальности связей и ограничений участия его членов. Документирование типов связей. При необходимости могут использоваться диаграммы «сущность-связь» (ER- диаграммы).
3. Связывание атрибутов с соответствующими типами сущностей или связей. Идентификация простых и составных атрибутов, простых и множественных атрибутов, а также производных атрибутов. Документирование сведений об атрибутах.
4. Определение доменов для всех атрибутов в каждой локальной концептуальной модели данных. Документирование сведений о доменах атрибутов.

5. Определение потенциального ключа для каждого типа сущности, если таких ключей окажется несколько, выбор среди них первичного ключа. Специализация или генерация типов сущностей (необязательный этап). Определение суперклассов и подклассов для типов сущностей.
6. Разработка диаграмм scущность - связьї (ER- диаграмм), содержащих концептуальное отражение представлений пользователя о предметной области приложения.
7. Обсуждение локальных концептуальных моделей данных с конечными пользователя для получения подтверждения того, что данная модель корректно отражает представления пользователя о приложении и предприятии.

SQLite компактная встраиваемая система управления базы данных, которая поддерживает динамическое типизирование данных.

Основными преимуществами SQLite являются:

1. Файловая: вся база данных хранится в одном файле, что облегчает перемещение.
2. Стандартизированная: SQLite использует SQL; некоторые функции опущены (например, RIGHT OUTER JOIN или FOR EACH STATEMENT), однако, есть и некоторые новые.
3. Отлично подходит для разработки и даже тестирования: во время этапа разработки большинству требуется масштабируемое решение. SQLite, со своим богатым набором функций, может предоставить более чем достаточный функционал, при этом будучи достаточно простой для работы с одним файлом и связанной сишной библиотекой.

SQLite имеет и ряд недостатков:

1. Отсутствие пользовательского управления: продвинутые БД предоставляют пользователям возможность управлять связями в таблицах в соответствии с привилегиями, но у SQLite такой функции нет.
2. Невозможность дополнительной настройки: опять-таки, SQLite нельзя сделать более производительной, поковырявшись в настройках так уж она устроена.

Python высокоуровневый язык программирования с динамической типизацией, поддерживающий объектно-ориентированный, функциональный и

императивный стили программирования. Это язык общего назначения, на котором можно одинаково успешно разрабатывать системные приложения с графическим интерфейсом, утилиты командной строки, научные приложения, игры, приложения для web и много другое. Он является одним из наиболее популярных современных языков программирования. Возможности и особенности:

- классы являются одновременно объектами со всеми ниже приведjнными возможностями;
- наследование, в том числе множественное;
- полиморфизм (все функции виртуальные);
- инкапсуляция (два уровня общедоступные и скрытые методы и поля). Особенность скрытые члены доступны для использования и помечены как скрытые лишь особыми именами;
- специальные методы, управляющие жизненным циклом объекта: конструкторы, деструкторы, распределители памяти;
- перегрузка операторов (всех, кроме is, '.', '=' и символьных логических);
- свойства (имитация поля с помощью функций);
- управление доступом к полям (эмуляция полей и методов, частичный доступ, и т. п.);
- методы для управления наиболее распространjнными операциями (истинностное значение, len(), глубокое копирование, сериализация, итерация по объекту, и. т.п);
- метапрограммирование (управление созданием классов, триггеры на создание классов, и др.);
- полная интроспекция;
- классовые и статические методы, классовые поля;
- классы, вложенные в функции и классы.

Однако, как и все остальные языки программирования, Python имеет ряд минусов, таких как:

- низкое быстродействие;
- невозможность модификации встроенных классов;

- глобальная блокировка интерпретатора (GIL) - GIL (Global Interpreter Lock) особенность, присущая CPython, Stackless и PyPy, но отсутствующая в Jython и IronPython.

PySide привязка языка Python к инструментарию Qt, совместимая на уровне API с PyQt [12]. Проект возник в результате нежелания создателей PyQt менять лицензионную политику для своего проекта. PySide появился в августе 2009 года. Основными разработчиками PySide являются программисты Digia.

Qt является кроссплатформенным фреймворком для приложений и пользовательских интерфейсов [13]. Он позволяет разработчикам писать приложения однократно и разворачивать их на операционных системах, не переписывая исходный код, в то время как Python является современным, динамическим языком программирования с большим сообществом разработчиков. PySide обеспечивает богатство фреймворка Qt-разработчикам благодаря объединению мощностей Qt и Python. Таким образом, Qt представляет собой первоклассную платформу для быстрой разботки программного обеспечения на языке Python, доступную на всех основных операционных системах.

PySide направлен на реализацию поддержки всех возможностей Qt, в том числе:

- QtQuick современной технологией создания пользовательского интерфейса, особенностью которой является разделение декларативного описания дизайна интерфейса и императивной логики программирования;
- QtMobility набора API (интерфейса программирования приложений) и фреймворков (программных платформ, определяющих структуру программной системы), ориентированных на разработку приложений для мобильных платформ.

В отличие от PyQt, PySide доступна для свободного использования как в открытых, так и закрытых, в частности, коммерческих проектах, поскольку лицензирована по LGPL.

GNU Lesser General Public License (стандартная общественная лицензия ограниченного применения GNU) лицензия на свободное программное обеспечение, одобренная Фондом свободного программного обеспечения и раз-

работанная как компромисс между GNU General Public License и простыми разрешительными лицензиями, такими, как BSD License и MIT License.

Заключение. Благодаря информационной системе, разработанной с учетом специфики конкретной предметной области, администраторы получают возможность осуществлять учет и хранение информации о работе компании, а также правильно выстроенная информационная система способствует достижению максимальной эффективности его работы.

В процессе выполнения работы был проведен анализ бизнес-процессов предметной области «Ветеринарная фармакологическая компания». В процессе выполнения работы были приобретены навыки разработки диаграмм UML, ER-диаграмм, баз данных и способов взаимодействия с ними. На основе полученных знаний при помощи высокоуровневого языка программирования Python был разработан интерфейс для взаимодействия с базой данных.

Для достижения целей, поставленных во введнии, было сделано следующее:

- Изучена методика проектирования бизнес-процессов предметной области «Ветеринарная фармакологическая компания».
- Построены диаграммы прецедентов и диаграммы деятельности. Данные диаграммы были построены с помощью PlantUML.
- Построена ER-диаграмма, и на ее основе разработана база данных в SQLite.
- Разработан графический интерфейс пользователя с помощью языка программирования Python.

Таким образом, в ходе работы были исследованы требования к информационной системе «Ветеринарная фармакологическая компания» и спроектирована сама система.