**Содержание**

[ВВЕДЕНИЕ 5](#_Toc37169151)

[1 АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧИ 6](#_Toc37169152)

[1.1 Постановка задач 6](#_Toc37169153)

[1.2 Описание предметной области 6](#_Toc37169154)

[1.3 Выбор и обоснование средств и методов решения задач 7](#_Toc37169155)

[2 СОЗДАНИЕ ФУНКЦИОНАЛЬНОЙ МОДЕЛИ ПО 8](#_Toc37169156)

[2.1 Разработка диаграммы вариантов использования 8](#_Toc37169157)

[2.2 Оценка трудозатрат на разработку ПО на основе диаграммы вариантов использования 9](#_Toc37169158)

[2.2.1 Определение весовых показателей действующих лиц 9](#_Toc37169159)

[2.2.2 Определение весовых показателей вариантов использования 9](#_Toc37169160)

[2.2.3 Определение технической сложности проекта 9](#_Toc37169161)

[2.2.4 Определение уровня квалификации разработчиков 10](#_Toc37169162)

[2.2.5 Оценка трудозатрат проекта 11](#_Toc37169163)

[2.3 Создание макета графического интерфейса пользователя 11](#_Toc37169164)

[3 СОЗДАНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ ПО 12](#_Toc37169165)

[3.1 Разработка диаграммы классов 12](#_Toc37169166)

[3.1.1 Описание модели 12](#_Toc37169167)

[3.1.2 Проектирование уровня данных 12](#_Toc37169168)

[3.1.3 Проектирование уровня интерфейса и бизнес– логики 13](#_Toc37169169)

[3.2 Разработка диаграмм последовательностей 13](#_Toc37169170)

[3.3 Разработка диаграмм деятельности 14](#_Toc37169171)

[3.4 Разработка диаграмм состояний 15](#_Toc37169172)

[4 СОЗДАНИЕ ФИЗИЧЕСКОЙ МОДЕЛИ ПО 17](#_Toc37169173)

[4.1 Разработка диаграммы компонентов 17](#_Toc37169174)

[4.2 Разработка диаграммы развертывания 18](#_Toc37169175)

[5 ОПИСАНИЕ РЕАЛИЗАЦИИ И ТЕСТИРОВАНИЯ ПО 19](#_Toc37169176)

[5.1 Детальная реализация функциональных частей ПО 19](#_Toc37169177)

[5.2 Тестирование ПО 22](#_Toc37169178)

[ЗАКЛЮЧЕНИЕ 23](#_Toc37169179)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc37169180)

[ПРИЛОЖЕНИЯ 25](#_Toc37169181)

ВВЕДЕНИЕ

Тема курсовой работы – проектирование информационной системы почтового отделения.

В соответствии с предметной областью работы, будущая база данных должна хранить сведения о сотрудниках, изданиях, регионах, подписчиках, и обеспечивать целостность этих данных.

Для обеспечения функционала, а также для удобства пользования информационной системой необходимо разработать приложение, которое позволит добавлять, удалять, редактировать и выводить подробную информацию по каждой возможной сущности. Приложение должно быть простым в использовании, которым могли бы пользоваться даже неквалифицированные сотрудники.

Актуальность данной работы определяется тем, что в настоящее время в связи с развитием и распространением почтовых отделений, то можно потом установить связь между почтовым отделением и базой данных, в которых и будет храниться вся информация. В таких целях возможно использование данного программного продукта. Он во много раз облегчит управление этим видом бизнеса.

Аналогов данной программы немало и не только в данной области. Они обычно разрабатываются под конкретную организацию, занимающуюся определенным видом деятельности. Создаваемая информационная система не привязывается к какому-либо почтовому отделению, и создается на основе некой абстрактной информационной системы.

Для создания информационной базы данных будет использоваться СУБД PostgreSql. Для создания приложения – среда Pycharm Professional.

1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

1.1 Анализ предметной области и выявление необходимого набора сущностей

Для построения информационной системы требуется для начала выделить, необходимый набор сущностей, которые описывают эту систему. Данный набор должен удовлетворять всем условиям на проектирование системы.

Определим минимальный набор сущностей, необходимый для проектирования информационной системы почтового отделения. Для определения первичного набора сущностей будет проведён анализ технического задания и предметной области.

Данной информационной системой может пользоваться не только сотрудники, но также и администратор, который имеет возможность управлять любым почтовым отделением данной ИС. Отсюда следует необходимость выделения сущности – *Group* которая будет хранить всевозможные роли*.*

Так как данный проект является веб-приложением, то должно предусматривается авторизация для каждой из ролей. Отсюда следует необходимость выделения сущности – *User* которая будет хранить идентификационные данные каждой из ролей.

Сущность post office описывает все возможные отделения, которые в свою очередь включают сотрудников, издания, подписки и регионы, которые охватывает ПО.

Сущность publishing house описывает издательские дома, которые в свою очередь печатают издания*.*

Сущность publication описывает все издания. Характеризуется именем и кодом издательского дома.

Сущность release описывает подробную информацию об издании. Характеризуется стоимостью, количеством изданий, кодом издания и кодом почтового отделения.

Сущность subscription описывает все подписки. Характеризуется датой подписки, описанием, сроком подписки (в месяцах), датой завершения подписки и кодом подробной информации об издании.

Сущность region описывает все регионы. Характеризуется индексом региона, кодом почтового отделения и кодом почтальона.

Сущность house описывает все дома. Характеризуется адресом и кодом региона.

Сущность follower описывает всех подписчиков. Характеризуется ФИО, почтой, мобильным телефоном и кодом дома.

1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Для построения инфологической концептуальной модели необходимо для каждой сущности, выявленной в предыдущем пункте, определить требуемый набор атрибутов. Атрибутом является поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

Ниже представлены сущности и определенные для них атрибуты, а также ключи (подчеркнуты). Имена сущностей и атрибутов указываются, как они будут определены в созданной базе данных, в скобках указывается перевод либо описание:

1. Post\_office (Почтовое отделение):
   * id (код почтового отделения);
   * name (название);
   * address (адрес);
2. Position (Роль сотрудника):
   * id (код позиции);
   * name (название);
   * salary (зарплата);
3. Employee (сотрудник):
   * id (код сотрудника);
   * first\_name (имя);
   * last\_name (фамилия);
   * middle\_name (отчество);
   * email (почта);
   * phone (мобильный телефон);
   * position\_id (код позиции);
   * post\_office\_id (код почтового отделения);
4. Region (Регион):
   * id (код региона);
   * index (индекс);
   * post\_office\_id (код почтового отделения);
   * postman\_id (код почтальона);
5. House (Дом):
   * id (код дома);
   * address (адрес);
   * region\_id (код региона);
6. Follower (Подписчик):
   * id (код подписчика);
   * first\_name (имя);
   * last\_name (фамилия);
   * middle\_name (отчество);
   * email (почта);
   * phone (мобильный телефон);
   * house\_id (код дома);
7. Subscription (Подписка):
   * id (код подписки);
   * start\_date (дата начала подписки);
   * description (описание);
   * term (срок подписки в месяцах);
   * end\_date (дата окончания подписки);
   * release\_id (код релиза);
8. Release (Релиз):
   * id (код релиза);
   * price (стоимость);
   * count (количество);
   * publication\_id (код публикации);
   * post\_office\_id (код почтового отделения);
9. Publication (Издание):
   * id (код издания);
   * name (название);
   * publishing\_house\_id (код издательского дома);
10. Publishing\_House (Издательский дом):
    * id (код издательского дома);
    * name (имя);
    * address (адрес);

1.3 Определение связей между объектами

Кроме атрибутов каждой сущности модель данных должна определять связи между сущностями. На концептуальном уровне связи представляют собой простые ассоциации между сущностями.

Связь – это ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако, одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся десятки или даже сотни сущностей, то между ними может быть установлено великое множество связей. Наличие такого множества связей и определяет сложность инфологических моделей.

Для реализации информационной системы почтового отделения необходимо установить все связи между объектами. А именно, нужно рассмотреть всю информационную систему и определить взаимное влияние объектов, составляющих систему.

Этот процесс изображен на рис. 1.1

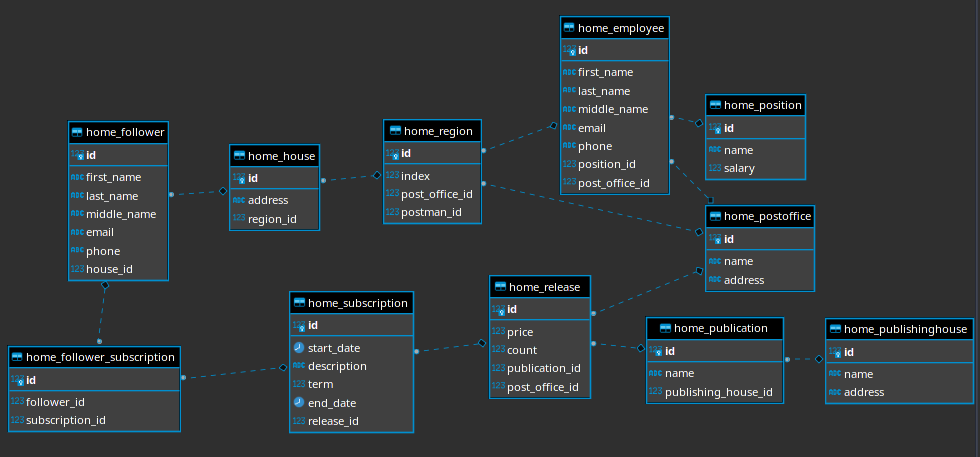


Рисунок 1.1 – Концептуальная схема базы данных

Проследить отношения, в которых состоят таблицы базы данных можно по схеме, изображенной на рис. A.1 в приложении A.

1.4 Описание полученной модели на языке инфологического проектирования

Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели (*англ.*entity-relationship model).

По правилам построения ER-диаграмм сущность изображается в виде прямоугольника. Связь изображается линией, которая связывает две сущности, участвующие в отношении. Степень конца связи указывается графически, множественность связи изображается в виде «вилки» на конце связи. Модальность связи так же изображается графически — необязательность связи помечается кружком на конце связи. Атрибуты сущности записываются внутри прямоугольника, изображающего сущность.

Таким образом, на основании результатов предыдущих пунктов, получим ER-диаграмму, проектируемой базы данных, представленную в приложении A, рисунок А.1.

2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

2.1 Построение набора необходимых отношений базы данных

Чтобы построить схему реляционной базы данных необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

В предыдущем пункте мы создали инфологическую концептуальную модель базы данных почтового отделения, построенной с помощью языка «Таблицы-связи». На основе полученной концептуальной модели можно определить набор необходимых отношений базы данных. На рисунке 2.1 представлены отношения для базы данных.

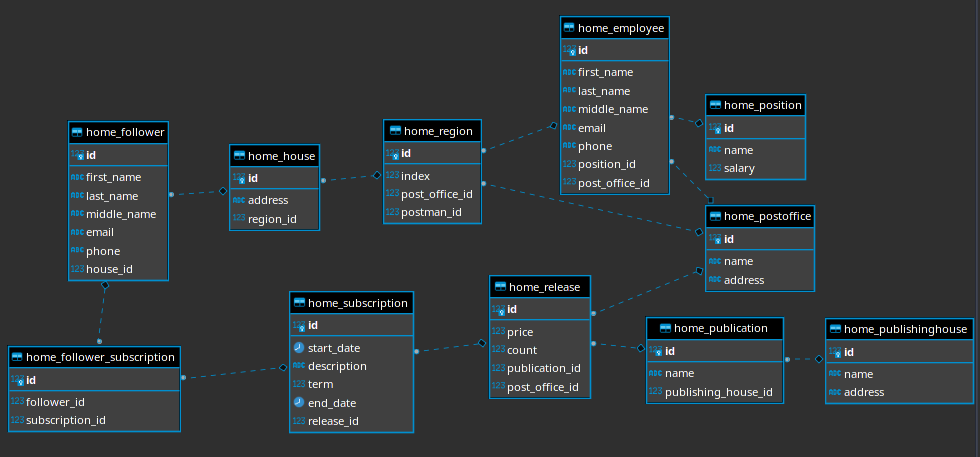


Рисунок 2.1 – Набор необходимых отношений базы данных

2.2 Задание первичных и внешних ключей определенных отношений

Поле первичного ключа служит уникальным определением записи. Оно также служит для связи таблиц. В связанных таблицах первичный ключ родительской таблицы становится внешним ключом в дочерней таблице. Внешний ключ дочерней таблицы отсылает к сведениям родительской таблицы.

Первичные ключи будут иметь постфикс PK, вторичные – FK. Также в подавляющем числе случаев названия вторичных ключей будут соответствовать названию связанных таблиц (без множественного числа), при несоответствии укажем название связанной таблицы во избежание путаницы. Также следует упомянуть, что в контексте данной работы первичные ключи будут являться единственным полем в таблице. В таблице 2.1 определены первичные и внешние ключи для отношений.

**Таблица 2.1** – Первичные и внешние ключи отношений

|  |  |  |
| --- | --- | --- |
| **Таблица** | **Первичный ключ** | **Внешние ключи** |
| Publishing\_house | id |  |
| Publication | id | publishing\_house\_id ( код издательского дома); |
| Release | id | publication\_id (код издания);  post\_office\_id (код почтового отделения); |
| Subscription | id | release\_id (код релиза); |
| Post\_office | id |  |
| Position | id |  |
| Employee | id | position\_id (код позиции);  post\_office\_id (код почтового отделения); |
| Region | id | post\_office\_id (код почтового отделения);  postman\_id (код почтальона); |
| House | id | region\_id (код региона); |
| Follower | id | house\_id (код дома); |

В дальнейшем построении схемы реляционной базы данных ключи будут служить для организации связей между отношениями.

Таким образом, ненормализованная схема базы данных представлена в приложении Bна рисунке B.1.

2.3 Третья нормальная форма

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение объёма БД. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации.

Нормальная форма — свойство  в, характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией.

Так как все атрибуты наших отношений атомарны, а каждое отношение имеет первичный ключ, то это означает, что отношения базы находятся в первой нормальной форме (1НФ).

Так как зависимости неключевых атрибутов от части составного ключа отсутствуют (все ключи в вышеописанных отношениях несоставные), а отношения базы находятся в 1НФ, то можно утверждать, что отношения базы удовлетворяют требованиям второй нормальной формы (2НФ).

Так как в отношениях базы отсутствуют зависимости неключевых атрибутов от других неключевых атрибутов, а присутствие 2НФ указано выше, можно сказать, что отношения базы находятся в третьей нормальной форме (3НФ).

2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений. У проектируемой базы все таблицы имеют первичный ключ.

2.5 Графическое представление связей между внешними и первичными ключами

По результатам нормализации, определении первичных и внешних ключей, связей между сущностями, была получена схема реляционной базы данных, представленная в приложении B на рисунке B.2. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами. Первичные ключи обозначаются буквами PK(от англ.PrimaryKey – первичный ключ), внешние ключи обозначаются FK (от англ.ForeignKey – внешний ключ).

3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Для реализации спроектированной базы данных была выбрана система управления базами данных PostgreSql 9.6. Это обусловлено тем, что, во-первых, данная СУБД получилась широкое распространение, а во-вторых, имеются свободно распространяемые сборки.

Произведем описание структуру каждой из таблиц с описанием типа полей.

Таблица PostOfficeхранит данные почтовых отделений. Её структура приведена в таблице 3.1.

**Таблица 3.1** – Характеристика атрибутов таблицы PostOffice.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "PostOffice". |
| name | varchar | Название почтового отделения. |
| address | varchar | Адрес почтового отделения. |

Таблица Positionхранит роли сотрудников почтового отделения. Её структура приведена в таблице 3.2.

**Таблица 3.2** – Характеристика атрибутов таблицы Position.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "Position". |
| name | varchar | Название роли. |
| salary | numeric | Заработная плата роли. |

Таблица Employeeхранит информацию о сотрудниках почтового отделения. Её структура приведена в таблице 3.3.

**Таблица 3.3** – Характеристика атрибутов таблицы Employee.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "Employee". |
| first\_name | varchar | Имя сотрудника. |
| last\_name | varchar | Фамилия сотрудника. |
| middle\_name | varchar | Отчество сотрудника. |
| email | varchar | Почта сотрудника. |
| phone | varchar | Телефон сотрудника. |
| position\_id | int4 | Идентификационный номер таблицы "Position". |
| post\_office\_id | int4 | Идентификационный номер таблицы "PostOffice". |

Таблица Regionхранит данные регионов. Её структура приведена в таблице 3.4.

**Таблица 3.4** – Характеристика атрибутов таблицы Region.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "Region". |
| index | int4 | Индекс региона. |
| post\_office\_id | int4 | Идентификационный номер таблицы "PostOffice". |
| postman\_id | int4 | Идентификационный номер таблицы "Employee". |

Таблица Houseхранит информацию о домах. Её структура приведена в таблице 3.5.

**Таблица 3.5** – Характеристика атрибутов таблицы House.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "House". |
| address | varchar | Адрес дома. |
| region\_id | int4 | Идентификационный номер таблицы "Region". |

Таблица Followerхранит данные о подписчиках. Её структура приведена в таблице 3.6.

**Таблица 3.6** – Характеристика атрибутов таблицы Follower.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "Follower". |
| first\_name | varchar | Имя подписчика. |
| last\_name | varchar | Фамилия подписчика. |
| middle\_name | varchar | Отчество подписчика. |
| email | varchar | Почта подписчика. |
| phone | varchar | Телефон подписчика. |
| house\_id | int4 | Идентификационный номер таблицы "House". |

Таблица PublishingHouseхранит информацию об издательских домах . Её структура приведена в таблице 3.7.

**Таблица 3.7** – Характеристика атрибутов таблицы PublishingHouse.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы " PublishingHouse ". |
| name | varchar | Название издательского дома. |
| address | varchar | Адрес издательского дома. |

Таблица Publicationхранит данные о публикациях издательского дома. Её структура приведена в таблице 3.8.

**Таблица 3.8** – Характеристика атрибутов таблицы Publication.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "Publication". |
| name | varchar | Название публикации. |
| publishing\_house\_id | int4 | Идентификационный номер таблицы "PublishingHouse". |

Таблица Releaseхранит данные о релизах почтового отделения. Её структура приведена в таблице 3.9.

**Таблица 3.9** – Характеристика атрибутов таблицы Release.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "Release". |
| price | numeric | Стоимость одного издания. |
| count | int4 | Количество выпущенных изданий. |
| publication\_id | int4 | Идентификационный номер таблицы "Publication". |
| post\_office\_id | int4 | Идентификационный номер таблицы "PostOffice". |

Таблица Subscriptionхранит данные почтовых отделений. Её структура приведена в таблице 3.10.

**Таблица 3.10** – Характеристика атрибутов таблицы Subscription.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "Subscription". |
| start\_date | date | Дата начала подписки. |
| description | text | Описание подписки. |
| term | int2 | Срок подписки(в месяцах). |
| end\_date | date | Дата окончания подписки. |
| release\_id | int4 | Идентификационный номер таблицы "Release". |

Таблица Userхранит данные почтовых отделений. Её структура приведена в таблице 3.11.

**Таблица 3.11** – Характеристика атрибутов таблицы User.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | serial | Идентификационный номер таблицы "User". |
| password | varchar | Пароль пользователя. |
| last\_login | timestampz | Последний вход в аккаунт. |
| is\_superuser | bool | Является ли аккаунт супер-пользователем. |
| username | varchar | Логин пользователя. |
| first\_name | varchar | Имя пользователя. |
| last\_name | varchar | Фамилия пользователя. |
| email | varchar | Почта пользователя. |
| is\_staff | bool | Есть ли возможность входа в админку. |
| is\_active | bool | Принадлежит ли сотрудник почтовому отделению. |
| date\_joined | timestampz | Дата создания аккаунта. |

4 ЗАПИСЬ ВЫРАЖЕНИЙ УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL

4.1 Запросы используемые в спроектированной базе данных

Все рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления. Диаграмма компонентов описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения. Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов. [11]

Программа состоит из исполняемого файла приложения и базы данных. Для компиляции исполняемого файла требуются файлы исходного кода, для которых в свою очередь требуются библиотеки языка C#. Исходный файл имеет доступ к таким интерфейсам, которые в свою очередь связаны с БД.

Диаграмма компонентов разрабатываемого программного обеспечения, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении З (рисунок З.1).

4.2 Триггеры используемые в спроектированной базе данных

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты – экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывании не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На диаграмме развертывания они не указываются.

Узел (node) представляет собой некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом, а также другие механические или электронные устройства, такие как датчики, принтеры, модемы, цифровые камеры, сканеры и манипуляторы.

Графически на диаграмме развертывания узел изображается в форме трехмерного куба. Узел имеет собственное имя, которое указывается внутри этого графического символа. [12]

На диаграмме развертывания разрабатываемого программного обеспечения показано, что система состоит из двух узлов. Первым узлом является база данных, а второй это непосредственно устройство пользователя управляема операционной системой Windows. Так же на диаграмме показано, что между этими узлами существует связь посредством протокола TCP/IP.

Диаграмма развертывания разрабатываемого программного обеспечения, выполненная в CASE – средстве EnterpriseArchitect8, представлена в приложении И (рисунок И.1)

5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Для реализации задачи по разработке информационной системы туристического клуба была выбрана СУБД PostgreSQL 9.6. Данная СУБД является одной из наиболее популярных систем управления и обслуживанию баз данных. В ней применяются новейшие разработки и последние достижения в области проектирования, построения и обслуживания баз данных.

Для создания приложения используется Pycharm Professional 2020.2, а языком написания приложения был выбран – Python и Фреймворк Django.

Django — это фреймворк, на основе которого можно создавать приложений на основе Python. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода

Django обладает большим функционалом, но его наиболее значимыми особенностями являются: встроенный ORM, автоматическая генерируемая админка, поддержка MTV и высокая скорость работы.

ORM — Object-RelationalMapping или в переводе на русский объектно-реляционное отображение. Это технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

ORM — это, по сути концепция о том, что Python объект можно представить как данные в БД (и наоборот).

**Bootstrap — это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.**

Фреймворк Bootstrap используется по всему миру не только независимыми разработчиками, но иногда и целыми компаниями.

Основная область его применения – это разработка frontend[4].

.

6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ

4.1 Запросы используемые в спроектированной базе данных

4.1 Запросы используемые в спроектированной базе данных

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте были освоены технологии разработки программного продукта, а также спроектирована «Информационная система мастера по маникюру и педикюру». Это приложение представляет собой программу для получения актуальной информации о данной информационной системе.

В ходе выполнения курсового проекта были созданы: функциональная, логическая, физическая модели ПО и следующие диаграммы:

* диаграммы вариантов использования;
* диаграммы классов;
* диаграммы последовательностей;
* диаграмма состояний;
* диаграммы деятельности;
* диаграммы компонентов;
* диаграммы развертывания.

Было использовано CASE – средство EnterpriseArchitect8, в котором содержаться все инструменты для проектирования, и отлично адаптировано под пользователя.

Разработанное приложение удовлетворяет всем требованиям, предназначенным для комфортной работы, и позволяет без проблем хранить и взаимодействовать с требуемыми данными.

Проведено полное тестирование приложения с разбором основных возможностей данного приложения.

В процессе выполнения курсового проекта были закреплены навыки по созданию различных диаграмм и макетов оконного интерфейса на языке UML 2.1, разработке приложений в Microsoft VS 2019, разработке целостной базы данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org. Дата обращения: 01.02.2020.
* Википедия. Свободная энциклопедия. [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/GNU\_Compiler\_Collection – Дата доступа: 2.03.2020.
* Официальный сайт справочной информации " Wikipedia ", [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/MySQL. Дата доступа – 28.04.2020 г.
* Руководство по языку программирования С. – Режим доступа: https://metanit.com/с/tutorial/. Дата доступа: 16.03.2020
* ClassServiceLoader <S> [Электронный ресурс] – Режим доступа: https://docs.oracle.com/с/6/docs/api/с/util/ServiceLoader.html. Дата обращения: 01.03.2020.
* Википедия. Свободная энциклопедия. [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki /Диаграмма\_состояний\_(UML) – Дата доступа: 2.03.2020.
* Наблюдатель[Электронный ресурс] – Режим доступа: https://refactoring.guru/ru/design– patterns/observer. Дата обращения 20.04.2020.
* Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма\_классов. Дата доступа – 28.04.2020 г.
* Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма\_последовательности. Дата доступа – 28.04.2020 г.
* Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма\_деятельности. Дата доступа – 28.04.2020 г.
* Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма\_состояний. Дата доступа – 28.04.2020 г.
* Официальный сайт справочной информации "Wikipedia" [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма\_компонентов. Дата доступа – 28.04.2020 г.

ПРИЛОЖЕНИЯ