



Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский государственный технический
университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет: «Информатика и системы управления»

Кафедра: «Программное обеспечение ЭВМ и информационные технологии»

Расчетно-пояснительная записка

к курсовой работе на тему:

«Web-приложение, содержащее информацию о вирусах»

Студент: Левушкин И. К.

Группа: ИУ7-62Б

Научный руководитель: Гаврилова Ю.М.

Москва, 2020 г.

Задание на курсовой проект

Разработать Web-приложение на языке программирования Python с использованием фреймворка Django. Основная задача данного приложения - хранение и показ информации о скорости распространения вируса, количестве зараженных, симптомов, вызываемых заболеванием, статистике выздоровевших и умерших от этого вируса в удобной для пользователя форме. Также, будет реализована возможность сравнивать графики (статистику) тех или вирусов.

В приложении можно будет осуществлять поиск вирусов по базе данных. Пользователь имеет возможность авторизоваться или зарегистрироваться в приложении, после этого пользователь сможет добавлять в избранное различную информацию про тот или иной вирус, что он просмотрел.

Содержание

| | |
|--|-----------|
| Введение | 5 |
| 1 Аналитический раздел | 6 |
| 1.1 Формализация задачи | 6 |
| 1.2 Разбор различных СУБД | 6 |
| 1.2.1 SQLite | 7 |
| 1.2.2 MySQL | 8 |
| 1.2.3 PostgreSQL | 9 |
| 1.3 Django | 10 |
| 2 Конструкторский раздел | 13 |
| 2.1 USE-CASE диаграмма | 13 |
| 2.2 Структура базы данных | 14 |
| 2.2.1 ER-диаграмма | 14 |
| 2.2.2 Диаграмма базы данных | 14 |
| 2.2.3 UML диаграмма данных | 15 |
| 2.3 Менеджеры приложения | 16 |
| 3 Технологический раздел | 21 |
| 3.1 Технологический стек | 21 |
| 3.2 Реализация хранения данных | 22 |
| 3.3 URL-адреса | 25 |
| 3.4 View | 27 |
| 3.5 HTML | 30 |
| 3.6 Заполнение базы данных | 33 |
| 3.7 Примеры работы программы | 34 |

Заключение 40

Список используемой литературы 40

Введение

Целью данной работы является закрепление навыков проектирования и создания реляционной базы данных, изучение особенностей разработки веб-интерфейса для работы с базой данных. Для выполнения поставленной задачи было спроектировано Web-приложение, в котором пользователи могут просматривать, редактировать и анализировать информацию о вирусах, регистрироваться и авторизоваться, добавлять в избранное понравившуюся информацию. Это позволит исследовать и изучить операции, которые присутствуют почти на многих современных проектах, а именно: чтение, редактирование и отображение информации из базы данных.

Результатом выполнения данной работы является веб-сайт с вышеописанным интерфейсом, а также замечания по поводу достоинств и недостатков тех или иных решений в разработке. Кроме того, будет приведена исследовательская информация по проекту, касающаяся работы базы данных.

1 Аналитический раздел

1.1 Формализация задачи

В соответствии с техническим заданием необходимо разработать систему, позволяющую осуществлять регистрацию учетной записи, вход в нее, выход из нее и ее редактирование.

Неавторизованный пользователь может просматривать информацию о вирусах, эпидемиях и местах, где происходило распространение вируса.

Авторизованный пользователь может добавлять понравившуюся информацию в избранное и анализировать ее. Анализ может осуществляться путем просмотра краткой информации о вирусе, а также построением графиков распространения вирусов.

Также, программный комплекс будет предоставлять особые полномочия администраторам веб-приложения, а именно - редактирование информации о вирусах, эпидемиях и местах, где происходило распространение вируса.

1.2 Разбор различных СУБД

Несмотря на то, что все системы управления базами данных выполняют одну и ту же основную задачу (т.е. дают возможность пользователям создавать, редактировать и получать доступ к информации, хранящейся в базах данных), сам процесс выполнения этой задачи варьируется в широких пределах. Кроме того, функции и возможности каждой СУБД могут существенно отличаться. Различные СУБД документированы по-разному: более или менее тщательно. По-разному предоставляется и техническая поддержка. При сравнении различных популярных баз данных, следует учитывать, удобна ли для пользователя и масштабируема ли данная конкретная СУБД, а также убедиться, что она будет хорошо интегрироваться с другими продуктами, которые будут использоваться

в системе. Для разработки была выбрана реляционная модель базы данных, поэтому рассмотрим три наиболее важные и популярные СУБД с открытым исходным кодом: SQLite[17], MySQL[18], PostgreSQL[6][7].

1.2.1 SQLite

База данных SQLite накладывает минимальное количество ограничений на пользователя. База не предоставляет сетевого интерфейса и использует для своей работы единственный файл. Первоначально была разработана как облегченная версия MySQL без модулей, которые не используются во многих проектах.

Преимущества:

- Главное преимущество заключается в том, что во встраиваемых или распределенных системах каждая машина несет полную реализацию базы данных. Это может значительно повысить производительность БД, поскольку снижает потребность в межпроцессных вызовах. Например, база встроена в ОС Android и предоставляет разработчикам приложений удобный интерфейс работы с ней.
- База использует для работы единственный файл, что хорошо влияет на ее переносимость.
- Система SQLite основана на языке запросов SQL, что удобно для разработки.
- Подходит для небольших проектов с небольшими БД.

Недостатки:

- Отсутствие встроенного шифрования данных, что стало стандартом для предотвращения наиболее распространенных хакерских атак в интернете.

- База не поддерживает систему учета пользователей, в то время как другие популярные СУБД поддерживают эту возможность.
- Только одна операция записи за транзакцию, что уменьшает производительность системы

1.2.2 MySQL

MySQL – самая популярная СУБД. Это многофункциональное открытое приложение, поддерживающее работу огромного количества сайтов. Система MySQL довольно проста в работе и может хранить большие массивы данных. Учитывая популярность MySQL, для этой системы было разработано большое количество сторонних приложений, инструментов и библиотек. MySQL не реализует полный стандарт SQL. Несмотря на это, MySQL предлагает множество функциональных возможностей для пользователей: автономный сервер баз данных, взаимодействие с приложениями и сайтами и т.п.

Преимущества:

- Простота в работе: MySQL очень просто установить и настроить. Сторонние инструменты, в том числе визуализаторы (интерфейсы) значительно упрощают работу с данными.
- Функциональность: MySQL поддерживает огромное количество функций SQL[3].
- Безопасность: MySQL предоставляет много встроенных продвинутых функций для защиты данных.
- Масштабируемость и производительность: MySQL может работать с большими объёмами данных.

Недостатки:

- Ограничения: структура MySQL накладывает некоторые ограничения, из-за которых не смогут работать продвинутые приложения.
- Уязвимости: метод обработки данных, применяемый в MySQL, делает эту СУБД немного менее надёжной по сравнению с другими СУБД.
- Медленное развитие: хотя MySQL является продуктом с открытым исходным кодом, он очень медленно развивается. Однако тут следует заметить, что на MySQL основано несколько полноценных баз данных (например, MariaDB).

1.2.3 PostgreSQL

PostgreSQL – это продвинутая открытая объектно-ориентированная СУБД. Часто используется при разработке веб-сайтов. Позволяет разработчикам управлять как структурированными, так и неструктуризованными данными. Может быть использована на большинстве основных платформ, включая Linux и MacOS. В отличие от других СУБД, PostgreSQL поддерживает очень важные объектно-ориентированные и реляционные функции баз данных: надежные транзакции ACID (атомарность, согласованность, изолированность, долговечность) и т.п. СУБД PostgreSQL основана на надежной технологии, она может одновременно обрабатывать большое количество задач. Хотя СУБД PostgreSQL не так популярна, как MySQL, для неё тоже разработано большое количество дополнительных инструментов и библиотек, которые упрощают работу с данными и увеличивают производительность СУБД.

Преимущества:

- PostgreSQL имеет открытый исходный код.
- База использует язык запросов SQL.
- PostgreSQL обладает большим сообществом разработчиков, которые помогут найти решение любой проблемы, связанной с

СУБД, в любое время суток.

- Помимо встроенных функций, PostgreSQL поддерживает множество открытых сторонних инструментов для проектирования, управления данными и т.п. Одним из таких инструментов является веб-приложение pgAdmin.
- База очень хорошо масштабируется и расширяется.
- Работает быстро и надежно, база способна обрабатывать террабайты данных.
- Поддерживает формат json.

Недостатки:

- Производительность в некоторых ситуациях ниже, чем у MySQL.
- Невысокая популярность, однако все больше проектов используют PostgreSQL в качестве СУБД.

1.3 Django

Django[8] — фреймворк для веб-приложений на языке Python[10]. Один из основных принципов фреймворка — DRY (don't repeat yourself). Веб-системы на Django строятся из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из заметных архитектурных отличий этого фреймворка от некоторых других, например, Ruby on Rails. Также, в отличие от многих других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений, а не автоматически задаются из структуры контроллеров.

Django проектировался для работы под управлением Apache с модулем mod_python и с использованием PostgreSQL в качестве базы данных. В настоящее время, помимо PostgreSQL, Django может

работать с другими СУБД: MySQL, SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere 9 и Oracle. Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Архитектура Django похожа на MVC[9] (Model–View–Controller). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представлением (View), а презентационная логика Представления реализуется в Django уровне Шаблонов (Templates). Из–за этого уровневую архитектуру Django часто называют MTV (Model-Template-View).

Первоначально разработка Django велась для обеспечения более удобной работы с новостными ресурсами, что достаточно сильно отразилось на архитектуре: фреймворк предоставляет ряд средств, которые помогают в быстрой разработке веб–сайтов информационного характера. Например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершенные действия, и предоставляет интерфейс для управления пользователями и группами.

Веб–фреймворк Django используется в таких крупных и известных сайтах, как Instagram, Disqus, Mozilla, The Washington Times, Pinterest, lamoda.

Преимущества:

- Простота в изучении.
- Чистота и читаемость.
- Разносторонность.

- Быстрота написания.
- Цельный дизайн

Выводы по аналитическому разделу

Исходя из представленной выше информации было принято решение использовать СУБД PostgreSQL, поскольку она обладает множеством открытых сторонних инструментов проектирования и управления данными, такими как pgAdmin, которые значительно упрощают разработку базы данных. Также, при своем проектировании фреймворк Django, используемый в данной работе, использовал PostgreSQL в качестве базы данных. Поэтому чтобы избежать непредвиденных трудностей при разработке ПО было решено использовать PostgreSQL.

2 Конструкторский раздел

2.1 USE-CASE диаграмма

Формулировка требования к возможностям программного обеспечения из раздела 1.1 требует большей конкретики. Диаграмма вариантов использования (рисунок 1) описывает взаимоотношения и зависимости между группами вариантов использования и действующими лицами, участвующими в процессе.

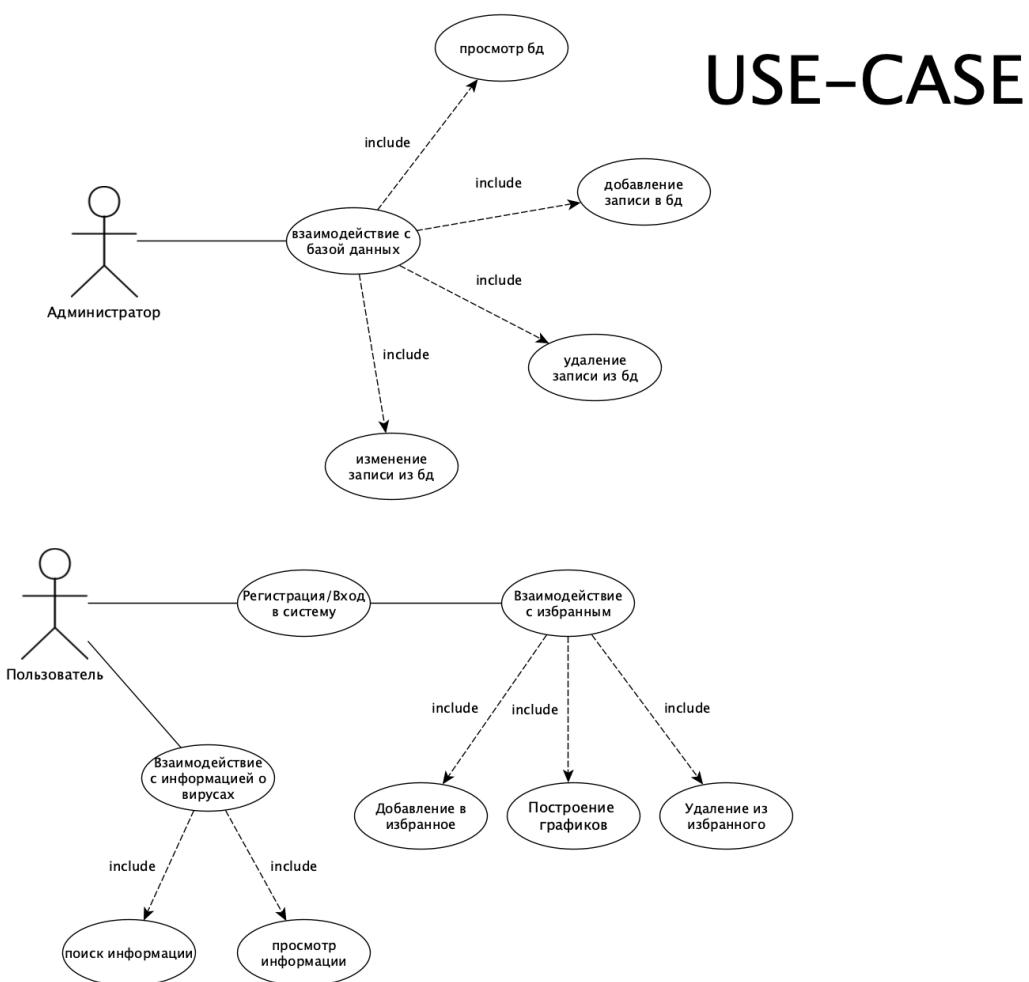


Рисунок 1: USE-CASE диаграмма

Важно понимать, что диаграммы вариантов использования не предназначены для отображения проекта и не могут описывать внутреннее устройство системы. Диаграммы прецедентов 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 [5].

2.2 Структура базы данных

2.2.1 ER-диаграмма

С помощью диаграммы «сущность — связь» (рисунок 2) описывается концептуальная схема предметной области [4]. Графическое отображение дает четкое представление о действующих сущностях и их атрибутах, а также устанавливает связи между различными сущностями.

2.2.2 Диаграмма базы данных

Ниже приведена диаграмма базы данных, показывающая виды связей между таблицами и основной набор их атрибутов.

ER

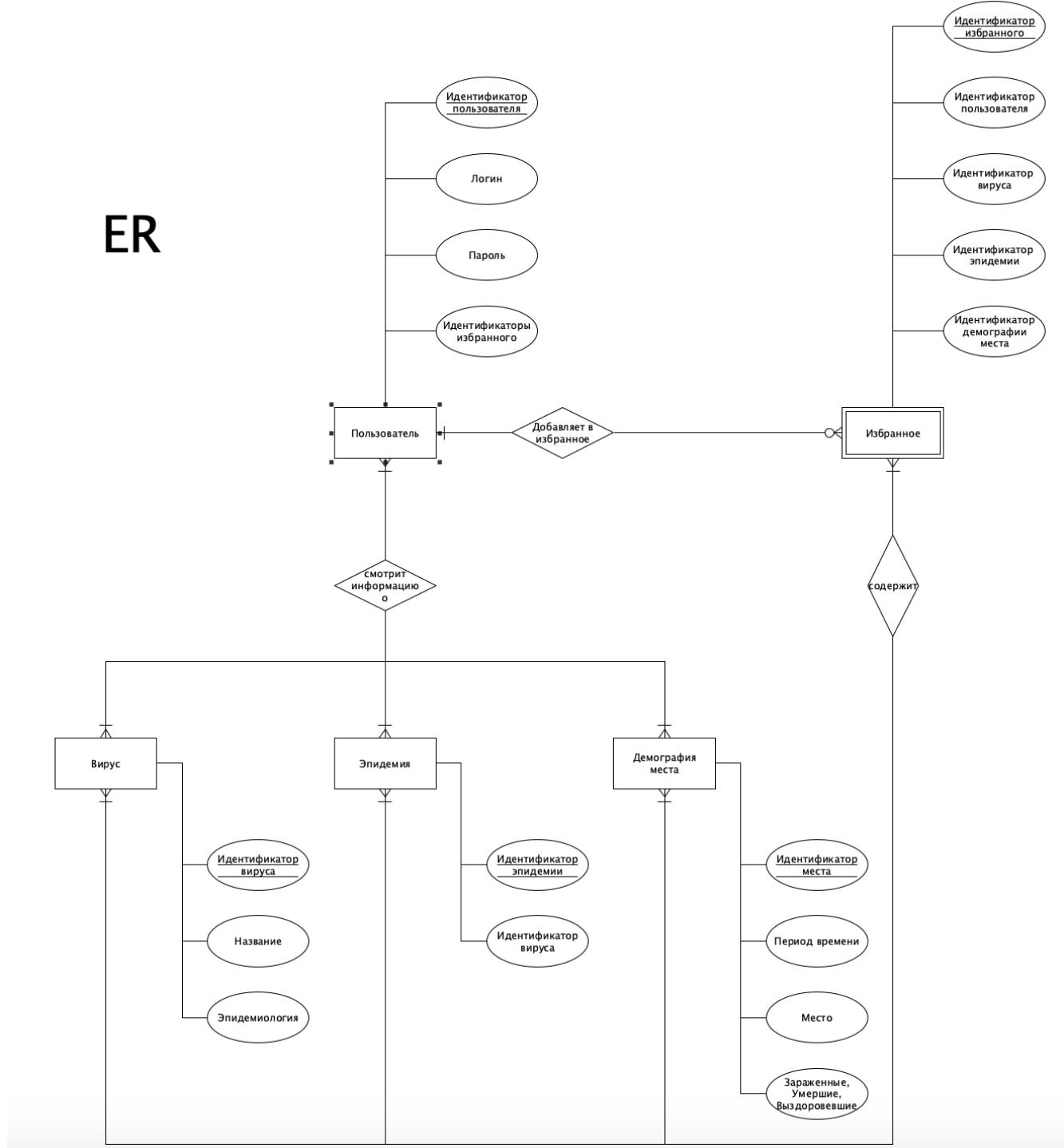


Рисунок 2: Диаграмма «сущность — связь»

2.2.3 UML диаграмма данных

Ниже приведена UML схема ассоциаций между таблицами базы данных и их взаимосвязь.

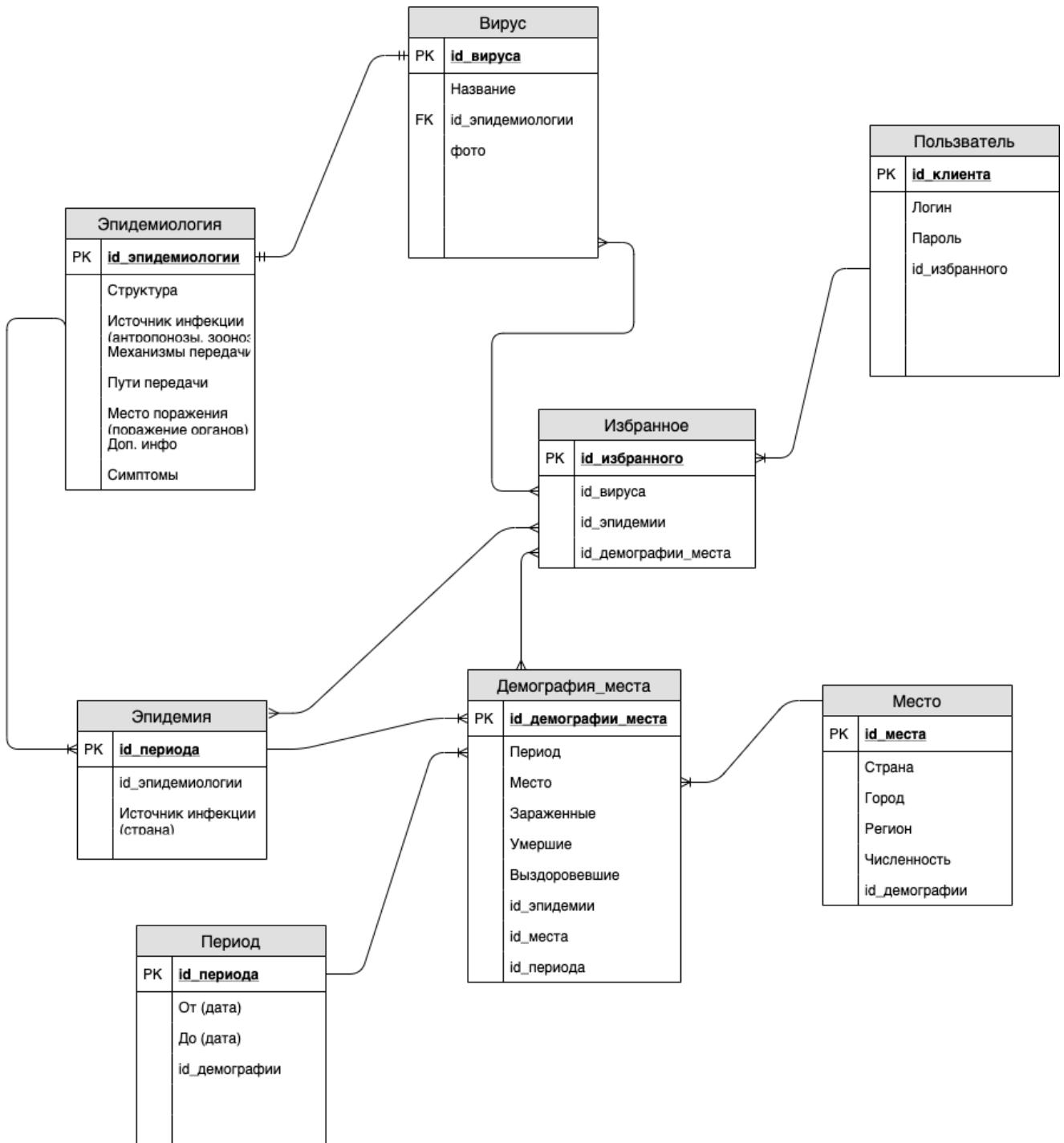


Рисунок 3: Диаграмма «сущность — связь»

2.3 Менеджеры приложения

В приложении реализованы менеджеры, которые позволяют управлять записями базы данных.

Ниже приведена UML схема менеджеров приложения.

Где

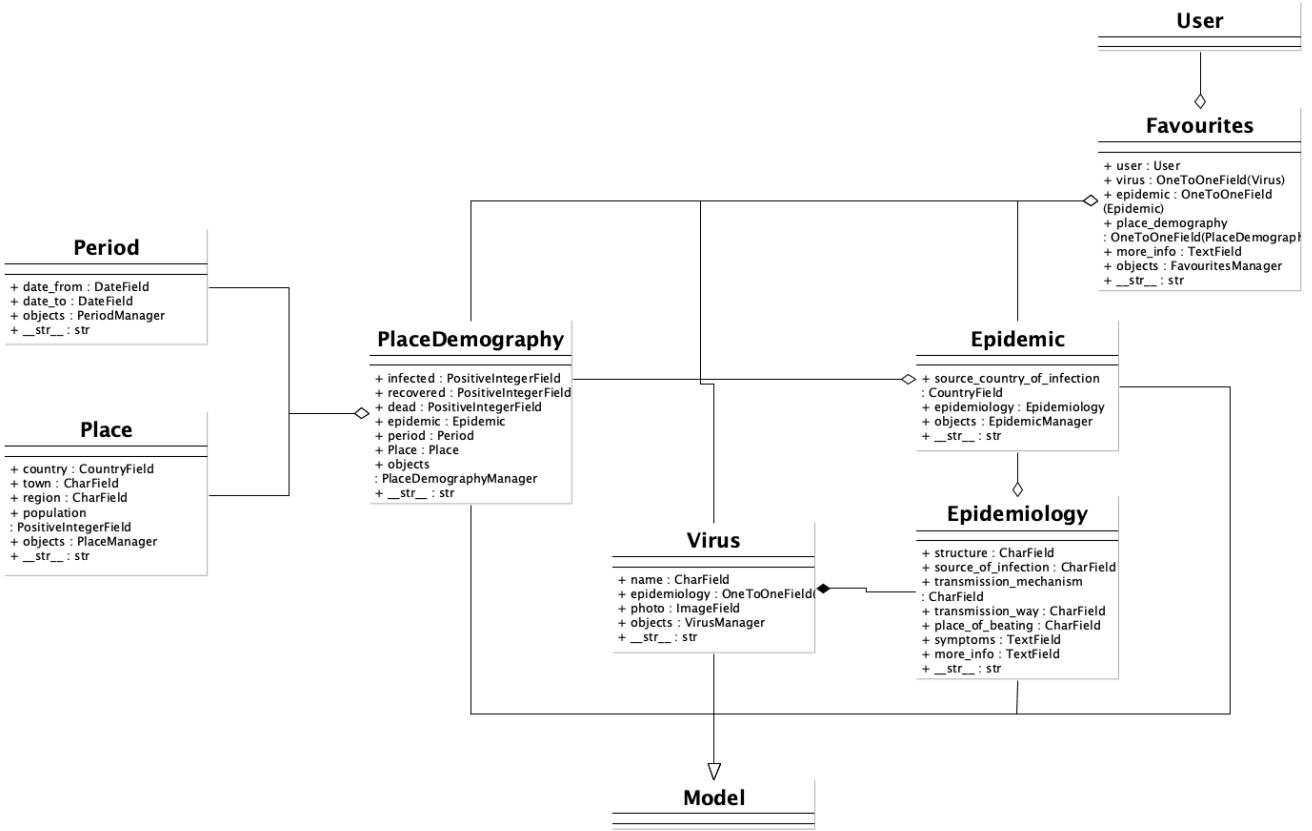


Рисунок 4: Диаграмма «сущность — связь»

Компонента с бизнес-логикой

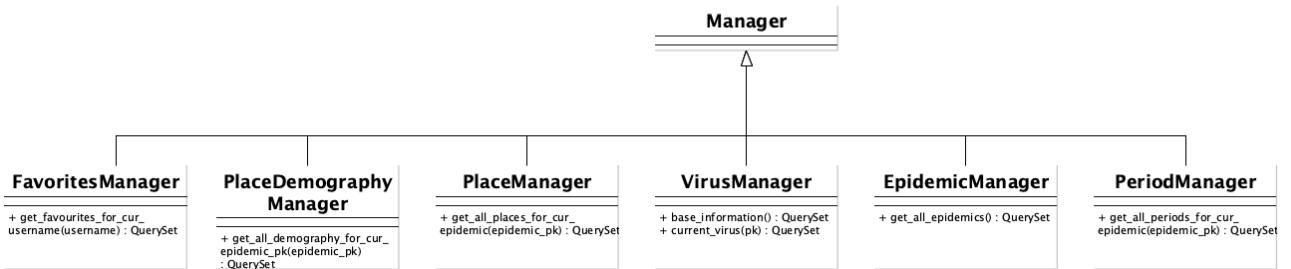


Рисунок 5: Диаграмма «сущность — связь»

- VirusManager имеет 2 метода:

- base_information(self) - сортирует все вирусы по их названию;

- current_virus(self, pk) - получает вирус по его id;
- EpidemicManager имеет 4 метода:
 - current_epidemic(self, pk) - получает эпидемию по ее id;
 - get_all_epidemics(self) - сортирует все эпидемии по названию страны-источника заражения;
 - get_epidemics_by_epidemiology_pk(self, epidemiology_pk)
 - получает все эпидемии по id эпидемиологии и преобразует их к удобной форме вывода на экран;
 - post_changes_in_epidemic(self, pk, request) - получает эпидемию по ее id и применяет к ней изменения из запроса request;
- PlaceDemographyManager имеет 7 методов:
 - get_places_demography_by_epidemic_pk(self, epidemic_pk)
 - получает все демографии места по id эпидемии;
 - get_regions_by_epidemic_pk(self, epidemic_pk),
 get_countries_by_region_pk(self, epidemic_id, region_name),
 get_towns_by_country_pk(self, epidemic_id, region_name, country_name) - функции выполняют сложные sql-запросы (с применением group by), чтобы получить сжатую информацию о регионах, странах и городах, соответственно;
 - get_current_region_sql(self, epidemic_pk, region_name),
 get_current_country_sql(self, epidemic_pk, region_name, country_name),
 get_current_town_sql(self, epidemic_pk, region_name, country_name, town_name) - функции выполняют sql-запросы и получают информацию о конкретном регионе, стране, городе, соответственно;
- PeriodManager имеет 1 метод:
 - get_all_periods_for_cur_epidemic(self, epidemic_pk) - получает все периоды времени по id эпидемии и сортирует их по дате;

- PlaceManager имеет 1 метод:
 - get_all_places_for_cur_epidemic(self, epidemic_pk) - получает все места заражения по id эпидемии;
- FavouritesManager имеет 17 методов:
 - get_all_viruses_by_username(self, username),
get_all_epidemics_by_username(self, username),
get_all_regions_by_username(self, username),
get_all_countries_by_username(self, username),
get_all_towns_by_username(self, username) - возвращает все добавленные конкретным пользователем вирусы, эпидемии, регионы, страны, города, соответственно, в избранное;
 - add_region(self, epidemic_pk, region_name, user),
add_country(self, epidemic_pk, region_name, country_name, user),
add_town(self, epidemic_pk, region_name, country_name, town_name, user) - добавляет регион, страну, город, соответственно, в избранное к конкретному пользователю;
 - delete_region(self, epidemic_pk, region_name, username),
delete_country(self, epidemic_pk, region_name, country_name, username),
delete_town(self, epidemic_pk, region_name, country_name, town_name, username) - удаляет регион, страну, город из избранного конкретного пользователя;
 - get_region_graphics_data(self, regions_info),
get_country_graphics_data(self, countries_info),
get_town_graphics_data(self, towns_info) - получает информацию о регионах, странах, городах, соответственно, в формате, используемым в дальнейшем для рисования графиков;

Выводы по конструкторскому разделу

В данном разделе была приведена диаграмма вариантов использования, описана структура базы данных при помощи ER-диаграммы, диаграммы базы данных и UML диаграммы данных. Также были выделены менеджеры приложения, через которые будут осуществляться запросы к базе данных.

3 Технологический раздел

3.1 Технологический стек

Front-end:

- Язык гипертекстовой разметки HTML5[13]
- Таблицы стилей CSS3[14] + Bootstrap4[11]
- Язык программирования JavaScript[12]

Back-end:

- СУБД PostgreSQL[6]
- Фреймворк Django[8]
- Язык программирования Python3[10]
- Библиотеки:
 - psycopg2[15] - библиотека python для связи с PostgreSQL
 - django-countries [16] - список всех стран мира

3.2 Реализация хранения данных

В листингах 1-3 сообщается о представлении сущностей в Django. Выделенные классы соответствуют схеме базы данных из предыдущего раздела (рисунок 3).

Листинг 1: Представление модели «вирус»

```
from django.db import models

from ..managers.virusmanager import VirusManager
from .epidemiology import Epidemiology

class Virus(models.Model):
    name = models.CharField(max_length=100, unique=True,
                           db_index=True)
    epidemiology = models.OneToOneField(
        Epidemiology, on_delete = models.CASCADE)
    photo = models.ImageField(
        upload_to='img/viruses',
        default='img/viruses/base_virus.jpg')
    )

    objects = VirusManager()

    def __str__(self):
        return self.name
```

Листинг 2: Представление модели «эпидемия»

```
from django.db import models
from ..managers.epidemicmanager import EpidemicManager
from django_countries.fields import CountryField
#official ISO 3166-1 list of countries max_length = 2
#get_FOO_display()
from .epidemiology import Epidemiology

class Epidemic(models.Model):
    source_country_of_infection = CountryField()
    epidemiology = models.ForeignKey(Epidemiology,
```

```

        on_delete = models.PROTECT)

photo = models.ImageField(
    upload_to='img/epidemic_countries',
    default='img/epidemic_countries/base_country.jpg'
)

more_info = models.TextField(
    default='Краткие сведения об эпидемии конкретного вируса.')
)

objects = EpidemicManager()

def __str__(self):
    return '{} {}'.format(self.epidemiology.virus.name,
                          self.source_country_of_infection)

```

Листинг 3: Представление модели «профиль»

```

from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver
from .favourites import Favourites

class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    phone = models.CharField(max_length=20, blank = True)
    photo = models.ImageField(
        upload_to='img/profiles',
        default='img/profiles/base_profile.jpg'
    )
    favourite = models.OneToOneField(Favourites,
                                     on_delete = models.PROTECT,
                                     null=True, blank = True)
    about = models.TextField(blank = True)

@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):

```

```
instance.profile.save()
```

3.3 URL-адреса

В Django используется так называемый URLconf (URL configuration). URLconf — это набор шаблонов, которые Django попробует сравнить с полученным URL, чтобы выбрать правильный метод для представления (view). Django сопоставляет URL-адреса, используя регулярные выражения (regular expressions). Регулярные выражения имеют множество правил, которые формируют поисковый шаблон. Ниже будет представлен список URL-шаблонов в проекте.

Листинг 4: url-шаблоны

```
from django.urls import path
from . import views as my_views
from django.contrib.auth import views as auth_views
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('', my_views.base_info, name='base_info'),
    path('profile/', my_views.profile, name='profile'),
    path('edit_profile/', my_views.edit_profile,
         name='edit_profile'),
    path('register/', my_views.register, name='register'),
    path('favourite/', my_views.favourite, name='favourite'),
    path('login/', auth_views.LoginView.as_view(
        template_name='virus_app/login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(
        template_name='virus_app/logout.html'), name='logout'),
    path('more_virus_info/<int:pk>', my_views.more_virus_info,
         name = 'more_virus_info'),
    path('more_virus_info/<int:pk>/edit_virus_info',
         my_views.edit_virus_info, name='edit_virus_info'),
    path('scroll_epidemic/<int:epidemiology_pk>',
         my_views.scroll_epidemic, name = 'scroll_epidemic'),
    path('scroll_epidemic/<int:epidemiology_pk>/more_epidemic_info/<int:epidemic_pk>',
         my_views.more_epidemic_info,
         name = 'more_epidemic_info'),
    path('scroll_epidemic/<int:epidemiology_pk>/more_epidemic_info/<int:epidemic_pk>/edit_epidemic_info',
         my_views.edit_epidemic_info, name='edit_epidemic_info'),
    path('scroll_epidemic/<int:epidemiology_pk>/scroll_region/<int:epidemic_pk>',
         my_views.scroll_region,
```

```
        name = 'scroll_region'),
    path('scroll_epidemic/<int:epidemiology_pk>/scroll_region/'
          '<int:epidemic_pk>/scroll_country/<str:region_name>/' ,
        my_views.scroll_country, name = 'scroll_country'),
    path('scroll_epidemic/<int:epidemiology_pk>/scroll_region/'
          '<int:epidemic_pk>/scroll_country/<str:region_name>/' 
          'scroll_town/<str:country_name>/' , my_views.scroll_town,
        name = 'scroll_town'),
]
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
                          document_root=settings.MEDIA_ROOT)
```

3.4 View

View или представление — это то место, где размещается логика работы приложения. Оно запрашивает информацию о модели, которую мы создали ранее и передаст её в шаблон и после этого приложение берет из базы данных значения, необходимые для отображения на странице. Ниже будет представлен пример View в проекте - virus.py.

Листинг 5: virus.py

```
from django.shortcuts import render, get_object_or_404
from django.shortcuts import redirect
from django.contrib import messages
from ..models.virus import *
from ..models.favourites import *
from ..models.placedemography import *
from .tools import *
import re

from django.contrib.auth.decorators import login_required

def base_info(request):
    if request.method == 'POST':
        items = request.POST.items()
        delete_from_fav_str = 'delete_from_fav'
        put_in_fav_str = 'put_in_fav'
        show_epidemics = 'show_epidemics'

        for item in items:
            if item[0].startswith(delete_from_fav_str):
                res = re.match(r'delete_from_fav_(\d+)', item[0])
                virus_pk = int(res.group(1))
                virus = Virus.objects.current_virus(virus_pk)
                #удаление вируса из избранного
                request.user.profile.favourite.virus.remove(virus)
                pass
            elif item[0].startswith(put_in_fav_str):
                res = re.match(r'put_in_fav_(\d+)', item[0])
                virus_pk = int(res.group(1))
                virus = Virus.objects.current_virus(virus_pk)
                #добавление вируса в избранное
                request.user.profile.favourite.virus.add(virus)
```

```

        elif item[0].startswith(show_epidemics):
            res = re.match(r'show_epidemics_(\d+)', item[0])
            virus_pk = int(res.group(1))
            virus = Virus.objects.current_virus(virus_pk)
            epidemiology_pk = virus.epidemiology.pk
            return redirect('scroll_epidemic',
                            epidemiology_pk=epidemiology_pk)

viruses = Virus.objects.base_information()
two_viruses_in_row = create_two_elems_in_row(viruses)
fav_viruses = Favourites.objects.get_all_viruses_by_username(
    request.user)
return render(request, 'virus_app/virus/scroll_virus.html',
              {'two_viruses_in_row' : two_viruses_in_row,
               'fav_viruses' : fav_viruses})

def edit_virus_info(request, pk):
    virus = Virus.objects.current_virus(pk)
    if request.method == 'POST':
        Epidemiology.objects.post_changes_in_epidemiology(
            virus.epidemiology.pk, request.POST)
    try:
        virus.photo = request.FILES['photo']
    except:
        pass
    virus.save()
    return redirect('more_virus_info', pk=pk)

epidemiology, other_epidemiology = Epidemiology.objects.\
    get_edit_data(virus.epidemiology.pk)

return render(request, 'virus_app/virus/edit_virus_info.html',
              {'virus' : virus,
               'epidemiology' : epidemiology,
               'other_epidemiology' : other_epidemiology})

def more_virus_info(request, pk):
    virus = Virus.objects.current_virus(pk)
    if request.method == 'POST':
        return redirect('edit_virus_info', pk=pk)

    epidemiology = Epidemiology.objects.get_printable_data(
        virus.epidemiology.pk)

```

```
return render(request, 'virus_app/virus/more_virus_info.html',
             {'virus' : virus,
              'epidemiology' : epidemiology})
```

3.5 HTML

HTML — это код, который может быть интерпретирован браузером, таким как Chrome, Firefox или Opera, чтобы отобразить веб-страницу пользователю. HTML (HyperText Markup Language) — язык гипертекстовой разметки. Гипертекст — это тип текста, поддерживающий гиперссылки между страницами. Под разметкой понимается введение в текст документа кода, который будет говорить браузеру, как интерпретировать веб-страницу. HTML код строится при помощи тегов, каждый из которых должен начинаться с < и заканчиваться >. Эти теги представляют элементы разметки. Ниже будет приведен пример HTML файла проекта - profile.html.

Листинг 6: profile.html

```
{% extends "virus_app/base_info.html" %}  
{% load static %}  
{% block title %}  
    Профиль  
{% endblock %}  
{% block style %}  
    <link type="text/css" href="{% static 'css/profile.css' %}"  
          rel="stylesheet">  
    <link type="text/css"  
          href="{% static 'css/profile_background.css' %}"  
          rel="stylesheet">  
{% endblock %}  
{% block content %}  
<div class="container card emp-profile">  
<form method="post">  
    {% csrf_token %}  
<div class="row">  
        <div class="col-md-4">  
            <div class="profile-img">  
                  
            </div>  
        </div>  
        <div class="col-md-6">  
            <div class="row">  
                <div class="profile-head">
```

```

<h5>
    {{ request.user.username }}
</h5>
<h6>
    {% if request.user.is_staff %}
        Администратор
    {% else %}
        Обычный пользователь
    {% endif %}
</h6>
<ul class="nav nav-tabs" id="myTab"
    role="tablist">
    <li class="nav-item">
        <a class="nav-link active"
            id="home-tab" data-toggle="tab"
            href="#home" role="tab"
            aria-controls="home"
            aria-selected="true">Информация</a>
    </li>
</ul>
</div>
</div>
<div class="row">
    <div class="tab-content profile-tab" id="myTabContent">
        <div class="tab-pane fade show active"
            id="home" role="tabpanel"
            aria-labelledby="home-tab">
            <div class="row" style="width:150%;">
                <div class="col-md-3">
                    <label>Имя</label>
                </div>
                <div class="col-md-9">
                    <p>{{ request.user.username }}</p>
                </div>
            </div>
            <div class="row" style="width:150%;">
                <div class="col-md-3">
                    <label>Email</label>
                </div>
                <div class="col-md-9">
                    <p>{{ request.user.email }}</p>
                </div>
            </div>
            <div class="row" style="width:150%;">

```

```
<div class="col-md-3">
    <label>Phone</label>
</div>
<div class="col-md-9">
    <p>{{ request.user.profile.phone }}</p>
</div>
</div>
<div class="row" style="width:150%;">
    <div class="col-md-3">
        <label>О себе</label>
    </div>
    <div class="col-md-9">
        <p>{{ request.user.profile.about }}</p>
    </div>
    </div>
</div>
</div>
</div>
<div class="col-md-2">
    <input type="submit" class="profile-edit-btn"
           name="btnAddMore" value="Ред." />
</div>
</div>
</form>
</div>
{% endblock %}
```

3.6 Заполнение базы данных

Первое, о чём стоит задуматься при создании базы данных – это её наполнение. В проекте достаточно большое количество таблиц и сущностей, поэтому задача её наполнения – важнейший аспект работы. Был написан скрипт на языке Python, который генерирует необходимую информацию, используя за основу логистическое уравнение[19], которое хорошо описывает скорость размножения популяции:

$$\frac{KP_0e^{rt}}{K + P_0(e^{rt} - 1)}$$

Где

- K - отвечает за растяжение графика вдоль оси y
- P_0 - отвечает за смещение смысловой части графика вдоль оси x
- r - отвечает за растяжение смысловой части графика вдоль оси x

Варьируя различные параметры формулы с помощью модуля random из стандартной библиотеки python, и происходила генерация данных. Далее эти данные записывались в файлы csv формата, откуда позже с помощью еще одного написанного скрипта на языке Python, преобразовывались в формат json и загружались в базу данных командой:

```
python3 manage.py loaddata myapp/data/current_data.json
```

3.7 Примеры работы программы

Ниже будут приведены различные примеры работы программы и комментарии к ним.

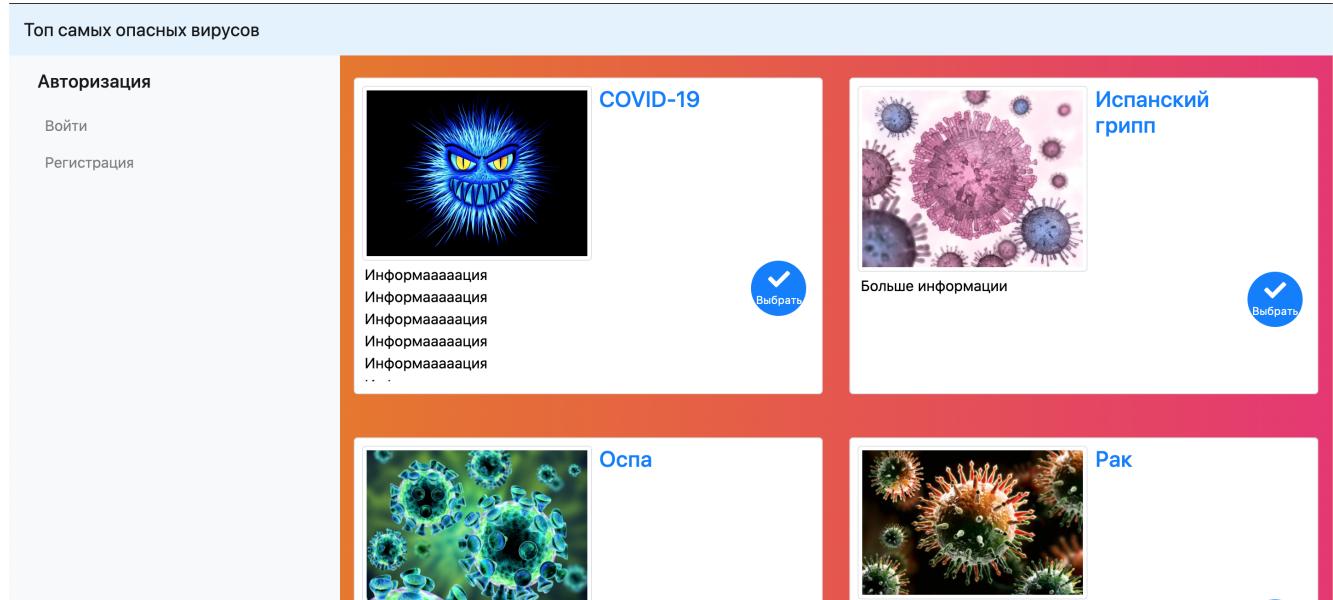


Рисунок 6: Главная страница (список вирусов)

На главной странице сайта можно увидеть список вирусов с краткой информацией о них. Далее можно спуститься на уровень ниже, нажав на кнопку "выбрать и получить список всех эпидемий, связанных с этим вирусом. Аналогично, нажав на кнопку "выбрать можно спуститься еще на уровень ниже и получить список всех регионов, в которых этот вирус распространялся, затем всех стран, находящихся в этом регионе, всех городов, относящихся к данной стране.

Топ самых опасных вирусов

Авторизация
Войти
Регистрация

Страна - источник инфекции Беларусь
Страна - источник инфекции Объединенные Арабские Эмираты

Краткое описание об эпидемии конкретного вируса.
Сообщение

Топ самых опасных вирусов > Список эпидемий

Авторизация
Войти
Регистрация

Регион Африка

| | |
|--------------------------|---------|
| Численность | 9770000 |
| Количество зараженных | 2173100 |
| Количество выздоровевших | 1009424 |
| Количество умерших | 1009424 |

С 7 июля 2019 г.
До 7 июля 2019 г.

Список эпидемий

Топ самых опасных вирусов > Список эпидемий > Список регионов

Авторизация
Войти
Регистрация

Страна Объединенные Арабские Эмираты

| | |
|--------------------------|---------|
| Численность | 9770000 |
| Количество зараженных | 2173104 |
| Количество выздоровевших | 1009424 |
| Количество умерших | 1009424 |

С 7 июля 2019 г.
До 7 июля 2019 г.

Топ самых опасных вирусов > Список эпидемий > Список регионов > Список стран

Авторизация
Войти
Регистрация

Город Town_name589

| | |
|--------------------------|-------|
| Численность | 97700 |
| Количество зараженных | 37731 |
| Количество выздоровевших | 22604 |
| Количество умерших | 22604 |

С 7 июля 2019 г.
До 7 июля 2019 г.

Город Town_name528

| | |
|--------------------------|-------|
| Численность | 97700 |
| Количество зараженных | 11925 |
| Количество выздоровевших | 1508 |
| Количество умерших | 1508 |

С 7 июля 2019 г.
До 7 июля 2019 г.

Город Town_name577

| | |
|-------------|-------|
| Численность | 97700 |
|-------------|-------|

С 7 июля 2019 г.
До 7 июля 2019 г.

Город Town_name535

| | |
|-------------|-------|
| Численность | 97700 |
|-------------|-------|

С 7 июля 2019 г.
До 7 июля 2019 г.

Список стран

Список регионов

Также, нажав на название какого-либо вируса или эпидемии, можно получить более подробную информацию о нем:

Топ самых опасных вирусов

Авторизация
Войти
Регистрация
Оля
Профиль
Избранное
Выйти

COVID-19

Эпидемиология

Структура Продолговатый

Источники Антропоноз

инфекции

Механизмы Вертикальный

передачи

Пути Укус

передачи

Места Покровная система

поражения

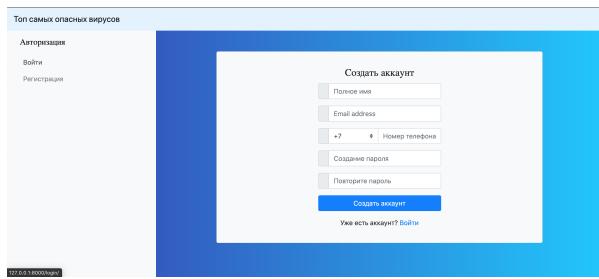
Симптомы Симптомы

Описание Информааааация
Информааааация
Информааааация
Информааааация
Информааааация
Информааааация

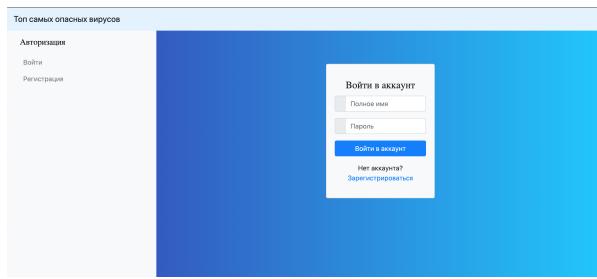
[Назад](#)

Рисунок 7: Подробная информация о вирусе

На любой из перечисленных страниц можно увидеть слева панель, на которой будет предложено либо войти пользователю либо зарегистрироваться:

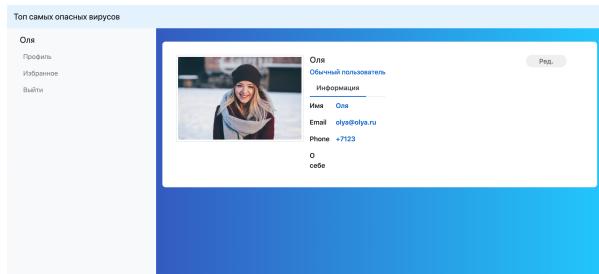


Страница регистрации

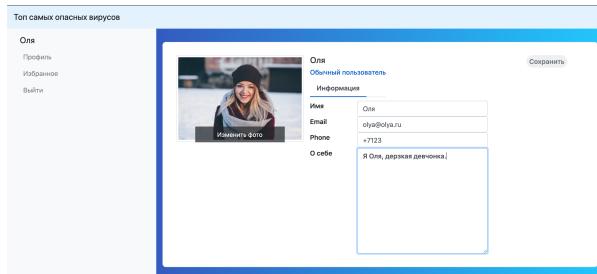


Страница входа в систему

Далее, после прохождения регистрации/входа в систему пользователя перекидывает на страницу его профиля, который можно отредактировать, добавив более подробную информацию о себе.

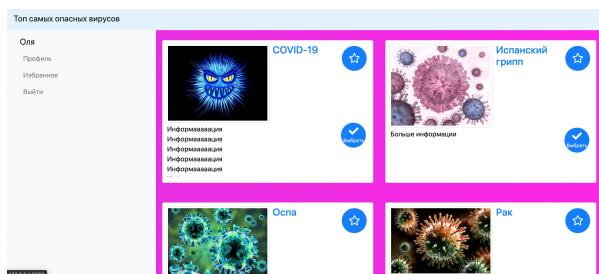


Профиль

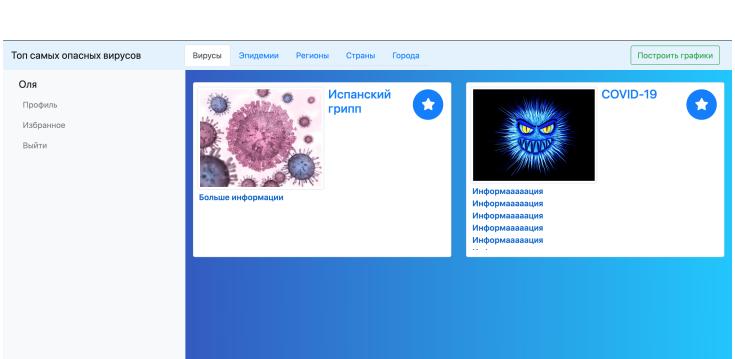


Редактирование профиля

При переходе на главную страницу сайта можно заметить, что на карточках с информацией о вирусах появились новые иконки - звездочки. Именно с помощью них осуществляется добавление и удаление из избранного понравившейся информации конкретному пользователю.



Главная страница (список вирусов) с возможностью добавлять элементы в избранное



Вирусы, попавшие в избранное

Перейдя во вкладку «избранное», пользователь также сможет выбрать из добавленных им регионов, стран, городов те, что захочет проанализировать, и построить графики.

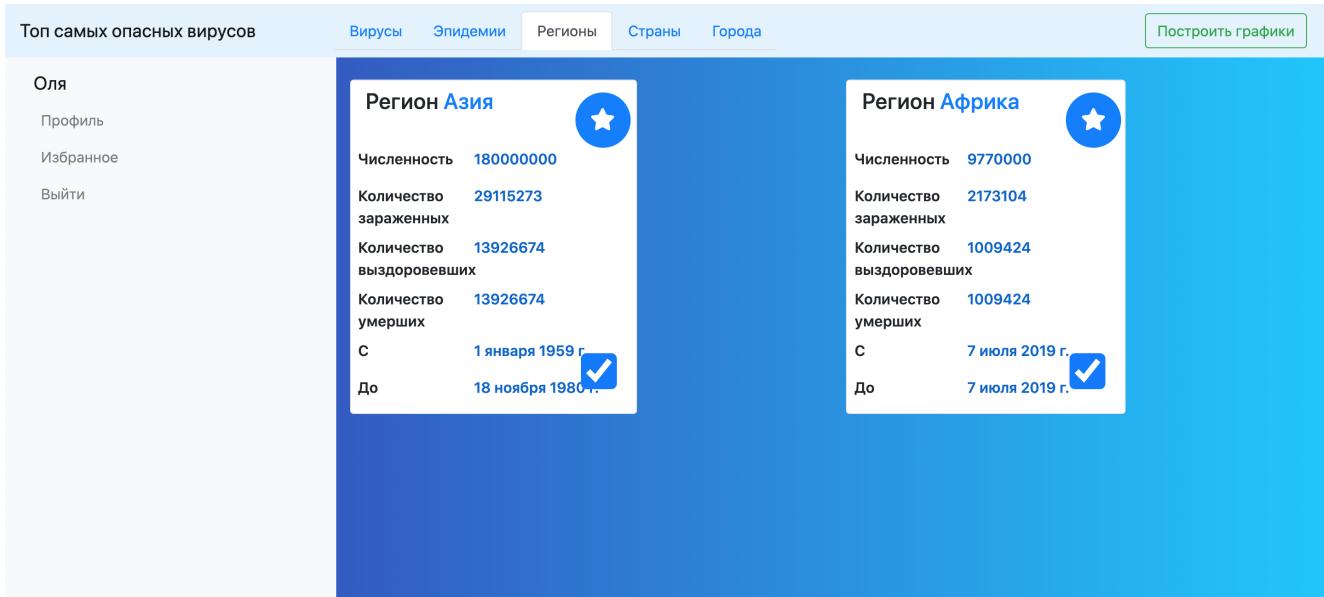


Рисунок 8: Регионы добавленные в избранное

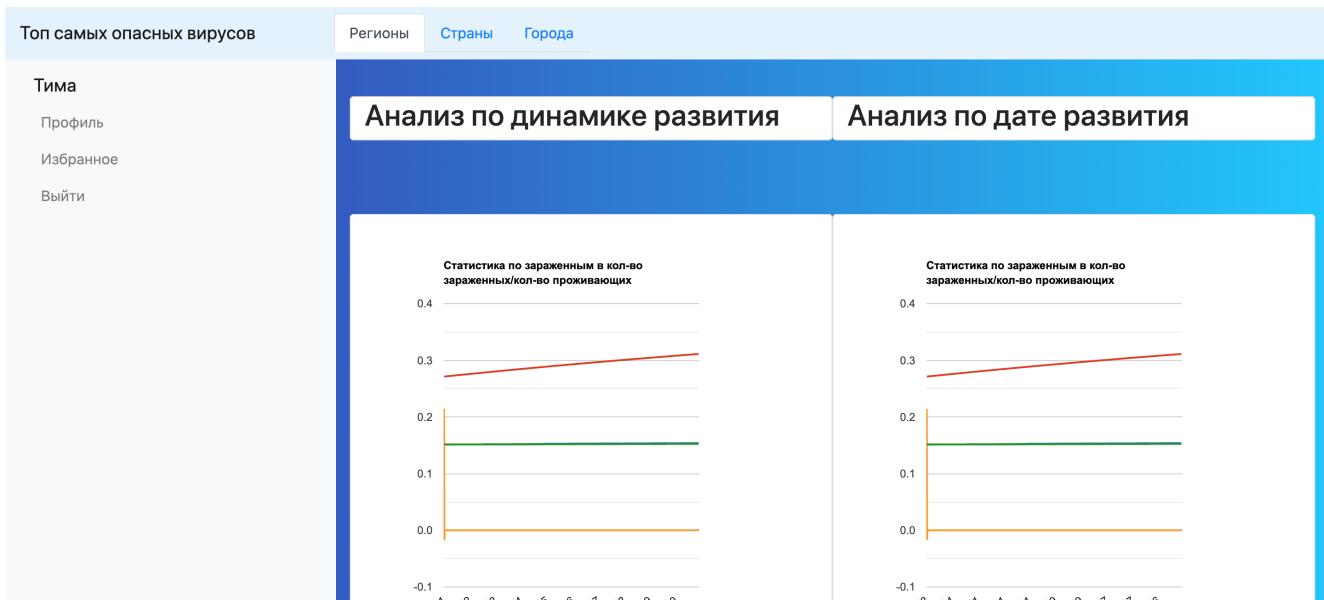


Рисунок 9: Регионы добавленные в избранное

Кроме того, чтобы было проще редактировать информацию о тех или иных вирусах, в системе предусмотрены дополнительные возможности для особых пользователей, администраторов сайта. Так, если мы войдем в систему под логином "ilya" мы увидим, что на странице профиля пользователя надпись «Обычный пользователь» сменилась на «Администратор».

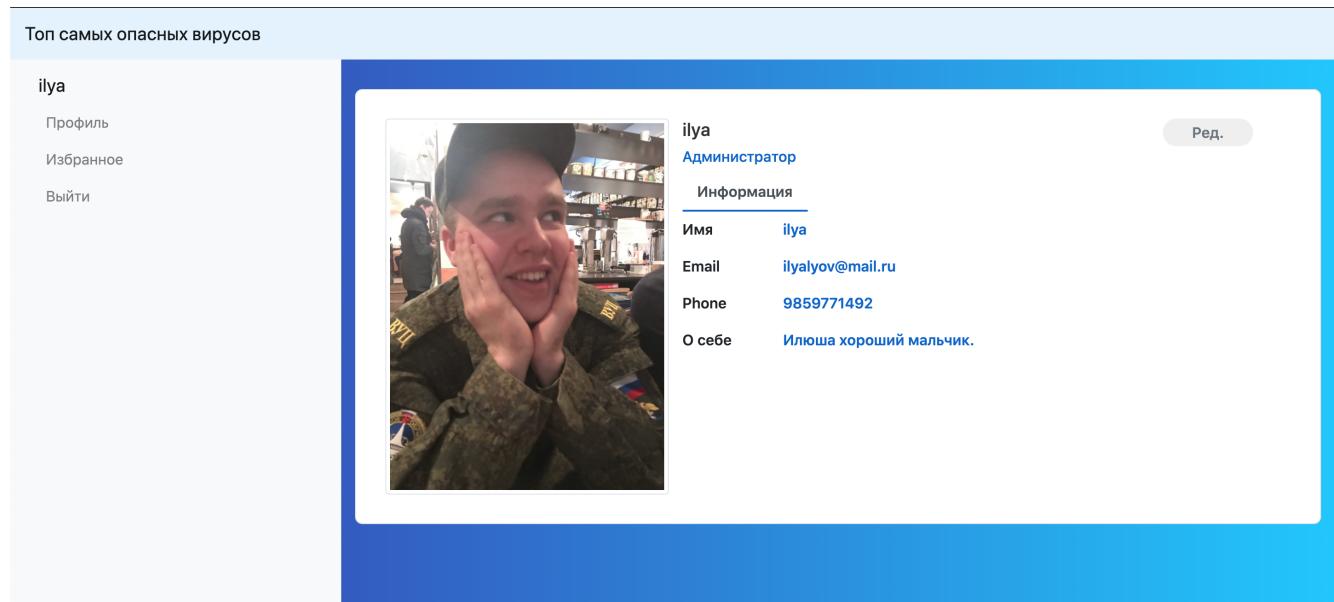


Рисунок 10: Главная страница (список вирусов) с возможностью добавлять элементы в избранное

При просмотре дополнительной информации какого-либо ви- руса или эпидемии появится кнопка «Редактировать информацию».

Дополнительный функционал для
администраторов

Редактирование информации о
вирусе

Выводы по технологическому разделу

В данном разделе были расписан основной технологический стек данной работы, были приведены реализации хранения данных, url-адресов, а также примеры реализаций контроллеров и представлений. Был описан способ заполнения базы данных.

Также, были приведены примеры работы программы, демонстрирующие весь заявленный функционал в техническом задании.

Заключение

В ходе проделанной работы был выполнен сравнительный анализ реляционных баз данных. Выяснилось, что для реализации поставленной задачи лучше всего подходит СУБД PostgreSQL.

Кроме того, была спроектирована архитектура программы на основе шаблона MVC.

В итоге, с помощью СУБД PostgreSQL и фреймворка Django было реализовано веб-приложение для хранения, просмотра и анализа информации о различных вирусах. Программа подходит как для личного, так и для командного использования.

Список литературы

- [1] Основные понятия баз данных [Электронный ресурс]. – Режим доступа: http://inf.susu.ac.ru/Klinachev/lc_sga_26.htm, свободный – (11.07.2020)
- [2] Реляционная модель данных: теоретические основы [Электронный ресурс]. – Режим доступа: https://functionx.ru/sql_relation_data_model.html, свободный – (11.07.2020)
- [3] Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство – М.: Вильямс, 2014
- [4] Построение реляционной структуры из ER-модели [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/50312/>, свободный – (11.07.2020)
- [5] UML — диаграмма вариантов использования (use case diagram) [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/47940/>, свободный – (11.07.2020)
- [6] PostgreSQL [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/>, свободный – (10.07.2020)

- [7] Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/282764/>, свободный – (13.07.2020)
- [8] Django [Электронный ресурс]. – Режим доступа: <https://www.djangoproject.com/>, свободный – (13.07.2020)
- [9] Шаблон проектирования MVC [Электронный ресурс]. – Режим доступа: <https://webformyself.com/shablon-proektirovaniya-mvc/>, свободный – (12.07.2020)
- [10] Python [Электронный ресурс]. – Режим доступа: <https://www.python.org/>, свободный – (12.07.2020)
- [11] Bootstrap [Электронный ресурс]. – Режим доступа: <https://getbootstrap.com/>, свободный – (12.07.2020)
- [12] JavaScript [Электронный ресурс]. – Режим доступа: <https://javascript.ru/manual>, свободный – (12.07.2020)
- [13] HTML5 [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/HTML/HTML5>, свободный – (12.07.2020)
- [14] CSS3 [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/CSS/Reference>, свободный – (12.07.2020)
- [15] Psycopg – PostgreSQL database adapter for Python [Электронный ресурс]. – Режим доступа: <https://www.psycopg.org/docs/>, свободный – (12.07.2020)
- [16] Django Countries [Электронный ресурс]. – Режим доступа: <https://github.com/SmileyChris/django-countries>, свободный – (12.07.2020)
- [17] SQLite Documentation [Электронный ресурс]. – Режим доступа: <https://www.sqlite.org/docs.html>, свободный – (12.07.2020)

[18] MySQL Documentation [Электронный ресурс]. – Режим доступа: <https://dev.mysql.com/doc/>, свободный – (12.07.2020)

[19] Дроздюк А. Логистическая кривая.. – Торонто: Choven 2019