Высшая Школа Экономики
Московский Институт Электроники и Математики

Групповой интернет-проект: Разработка приложения для мониторинга и предсказания статистики по коронавирусу COVID-19

Павел Цветов, Устимов Владислав,
группа БИВ 161

Постановка задачи

Разработка приложения для мониторинга и предсказания статистики по коронавирусу COVID-19 на языке Python с использованием фреймворка Flask, JavaScript, HTML5, CSS3, SASS, SQL.

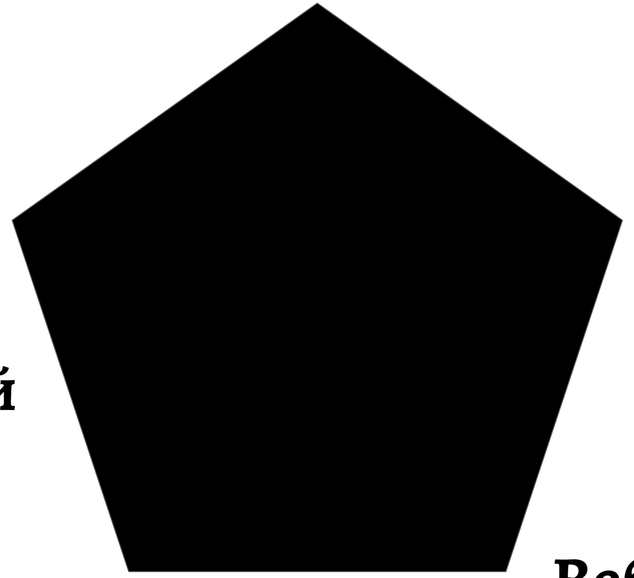
- Актуальная статистика по Коронавирусу
- Возможность предсказаний числа заболевших и умерших
- Разработка API для получения данных
- Реализация базы данных
- Система авторизации

Код проекта:
https://github.com/yaiestura/coronavirus_prediction

Актуальность

COVID-19

Более 390 тыс. случаев заражения, 17000+ смертей



Использование нейросетей для предсказаний

Современные приложения, адаптивность, стиль

Веб-разработка, комплексные страницы, Dashboard, инфографика, разнообразие фреймворков

Новизна

- Веб-приложение для ознакомления со статистикой по Коронавирусу
- Возможность предсказаний числа Коронавирусных заболеваний
- Написание собственного API сервиса для получения данных
- Современный дизайн, собственный стиль, использование SASS
- Приложение с авторизацией
- Google Авторизация

Область применения

- Обычные пользователи интернета, просмотр последней информации и статистики
- Центры сбора статистики и анализа данных
- Возможность платной подписки с расширенными возможностями (напр. полноценный и расширенный PDF-отчет)
- Врачи, ученые, мировое сообщество
- Веб-программисты, Backend/FrontEnd разработчики
- Блог по Коронавирусу, публикация новостей и мнений
- Вклад в открытое сообщество / community (напр. Github)

Метрики проекта

1. 4 недели
2. Более 6800 строк кода
3. 20+ страниц
4. 24 HTML страницы
5. 20+ использованных сторонних JS скриптов, 25+ собственных JS скриптов
6. 15+ ключевых скриптов и файлов Python

Обзор и анализ аналогов

Как таковых комплексных аналогов не существует, т.к Коронавирус начал активно распространяться в последние 3 месяца.

Попытка написания полноценного комплексного Full-Stack Веб-приложения на Python комбинируя лучшее от существующих источников и аналогов.

Большинство аналогов, карт предлагают свежие и актуальные данные по статистике, предлагают интерактивный обзор, карты, инфографику по дням.

Большинство аналогов - зарубежные, все мониторы Коронавируса в основном от известных организаций по статистике и здравоохранению - WHO, John Hopkins University, Stanford, Зарубежных новостных агенств и центров по анализу данных.

Основной сайт-аналог, откуда берутся данные

worldometer Coronavirus Population

https://www.worldometers.info/coronavirus/

COVID-19 CORONAVIRUS PANDEMIC

Last updated: March 24, 2020, 10:24 GMT

Case Graphs - Death Graphs - Countries - Death Rate - Incubation - Age - Symptoms - News

Coronavirus Cases:

386,962

view by country

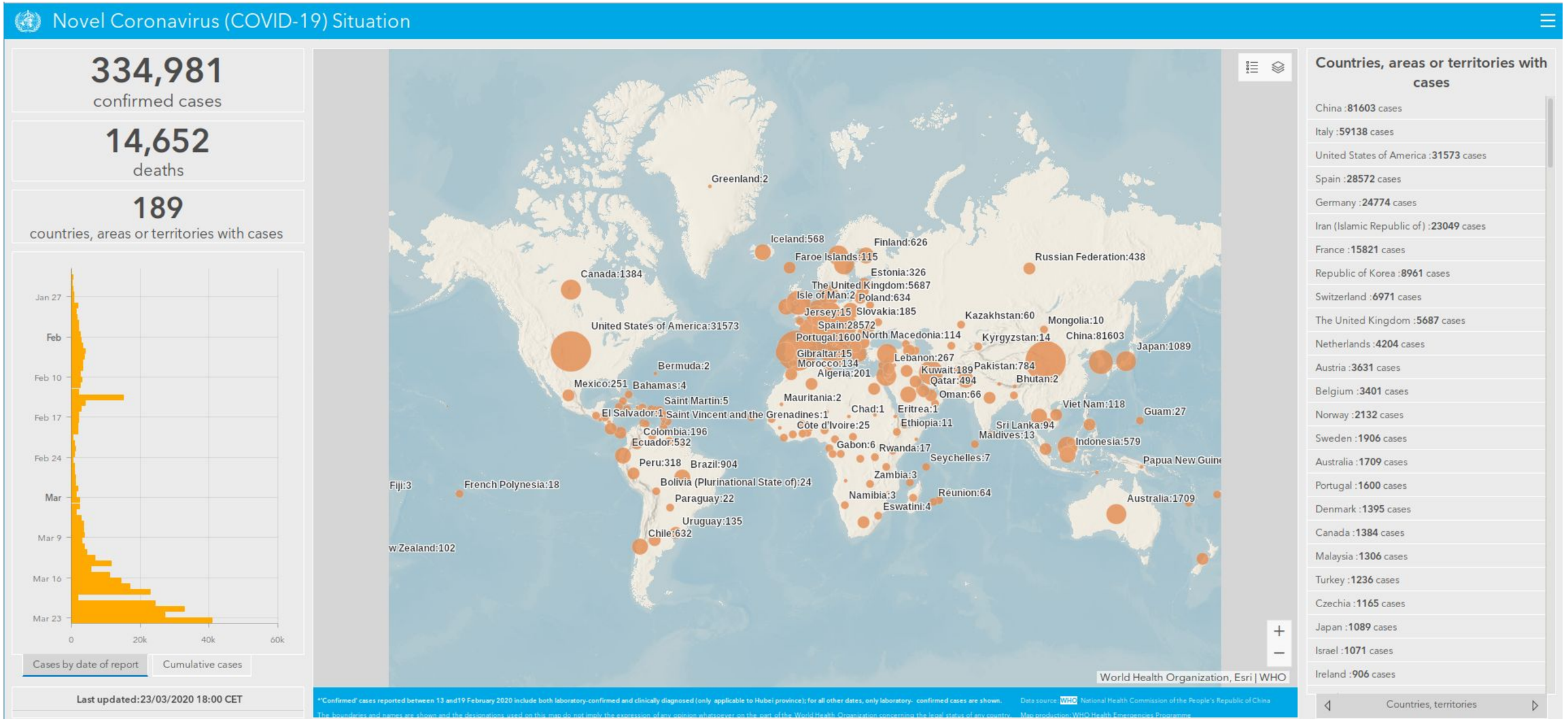
Deaths:

16,751

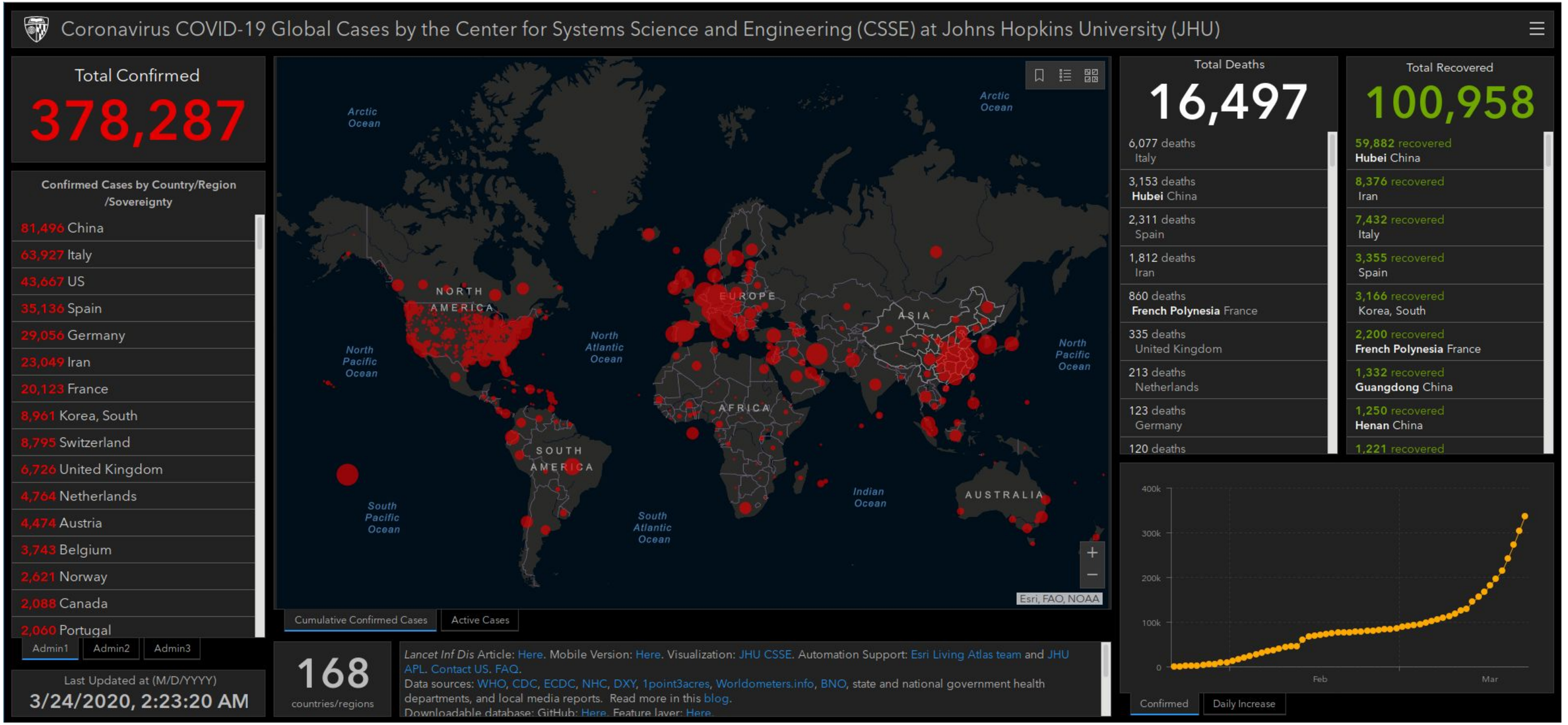
Recovered:

102,404

Примеры страниц аналогов (WHO Dashboard)



Примеры страниц аналогов (JHU University)



План выполнения проекта

1. Инициализация проекта
2. Определение структуры проекта
3. Размышления над дизайном проекта
4. Определение цветовой палитры, создание логотипа
5. Шаблонный HTML-код
6. Написание парсера данных
7. Создание и реализация Базы Данных
8. Инициализация логики приложения, путей
9. Landing страница
10. Dashboard страницы
11. Блог страницы
12. Написание нейросети
13. Обучение нейросети и сохранение моделей
14. Инфографика, карты, графики
15. Страница документации
16. Тестирование проекта
17. Публикация в репозиторий

Обоснование выбора средств разработки

Микрофреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2.

	Flask	**Django**
Тип фреймворка	Минифреймворк	Full Stack Веб-фреймворк
Опыт	Легкость в обучении	Высокий порог входа
Аутентификация	Нет аутентификации, Flask-Login	Встроенная аутентификация
Базы данных	Flask-SQLAlchemy	Встроенная ORM и СУБД
Формы	Не поддерживает формы, WTForms	Встроенные формы
Админ-панель	Нет административного интерфейса, Flask-Admin	Панель администратора - Django Admin

Работа во Flask

HTML-Template (Jinja2)

```
{% extends
"dashboard/coronavirus/layout.html"
%}
{% block scripts %}
   Код CSS
{% endblock %}
{% block content %}
{% for country in countries %}
    <div></div>
{% endfor %}
{% endblock %}
{% block scripts %}
   Код JS скриптов
{% endblock %}
```

Route (логика работы)

```
@main.route("/countries", methods=['GET', 'POST'])
@login_required
def countries():
   countries = CountriesAdvDataParser()
   countries_data = countries.get_countries()

   return
render_template('dashboard/coronavirus/countries.ht
ml', countries=countries_data,
username=current_user.username,
                  avatar=current_user.image_file,)
```

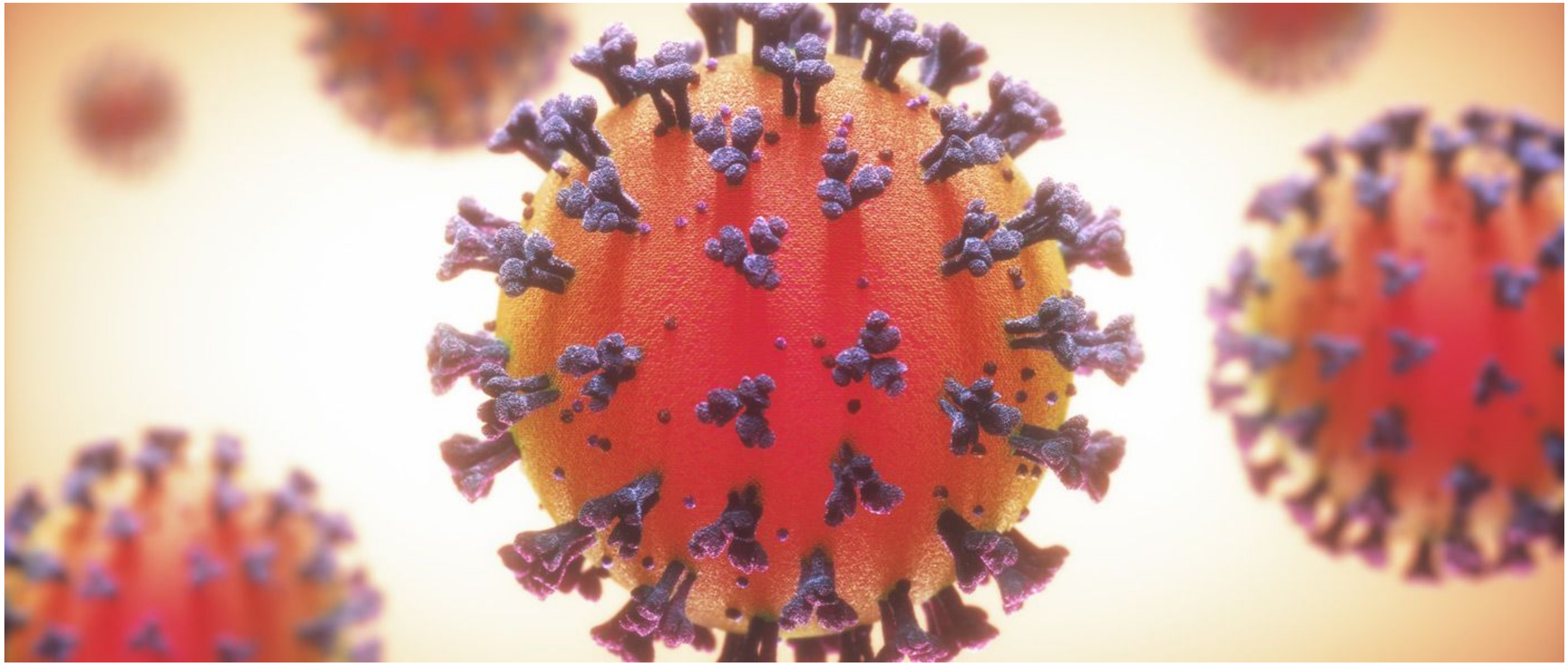
Обоснование выбора средств разработки

1. Bootstrap 4 - Веб фреймворк со встроенной адаптивностью, Grid-системой
2. JQuery - JS фреймворк
3. SASS - расширение над CSS, новые возможности и улучшения
4. AmCharts 4 - библиотека для работы с графикой, визуализацией, картами
5. БД SQLite - реляционная SQL база данных
6. Beautiful Soup 4 - библиотека Python для парсинга данных
7. Flask-SQLAlchemy - библиотека для работы с реляционными БД во Flask
8. Flask-Login - библиотека для работы с авторизацией и логикой Oauth во Flask
9. Pytorch 1.4 - библиотека Python для написания нейросетей
10. VSCode - редактор кода

“Фишки” проекта

1. Данные по COVID-19 в режиме реального времени
2. REST API (API, возможность получить данные в сыром виде - json)
3. Система авторизации пользователей
4. Google Авторизация
5. Система обработки HTTP ошибок (ошибки 403,404, 500) - соотв. страницы
6. Нейросеть, предсказания модели по случаям и смертям (до 10 дней)
7. Возможность переобучить модель по новым данным
8. База данных (реляционная, SQL)
9. Визуализации, инфографика, карты, графики и пр.
10. Блог (возможность написания и публикации постов)
11. Возможность генерации PDF-отчета

Основные цвета



#fd7a6b

#404e67

Белый, серый, оттенки серого

Создание логотипа в фотошоп

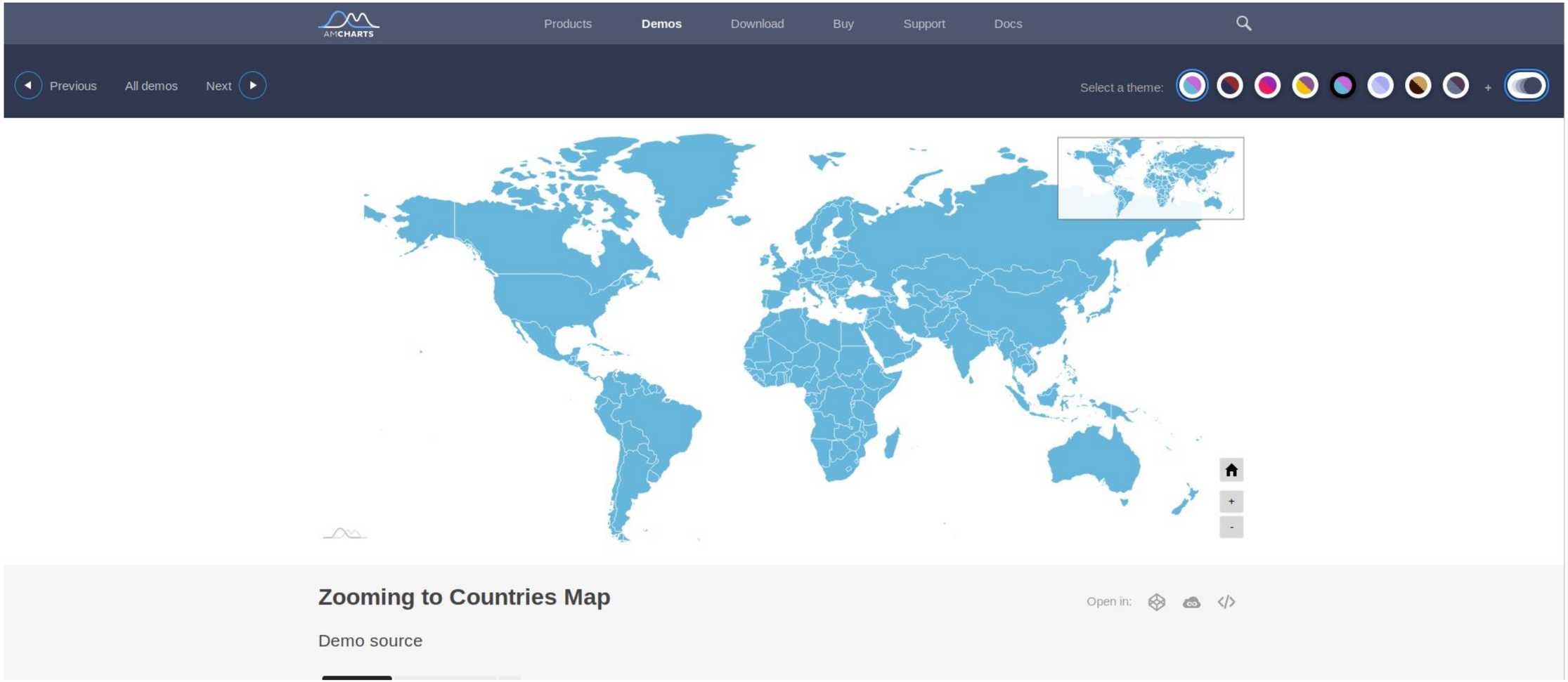
Символ(значок) вируса и текст - covid (референс на коронавирус, COVID-2019)
Сохранение основных цветов в логотипе, на белом фоне используется синяя заливка текста

Обоснование выбора средств разработки

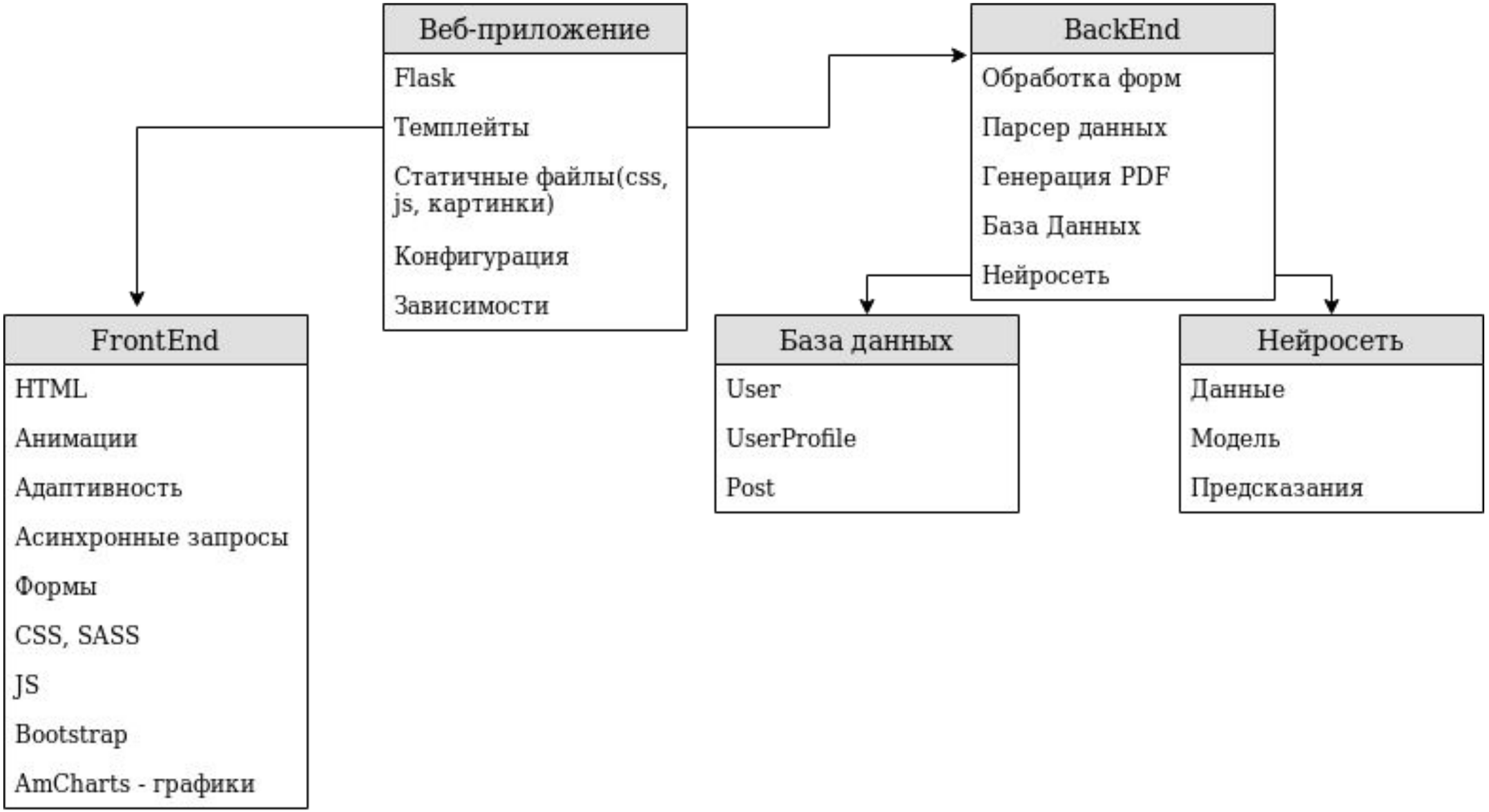
Библиотека JS - AmCharts для визуализации данных, графиков, карт

Обоснование выбора средств разработки

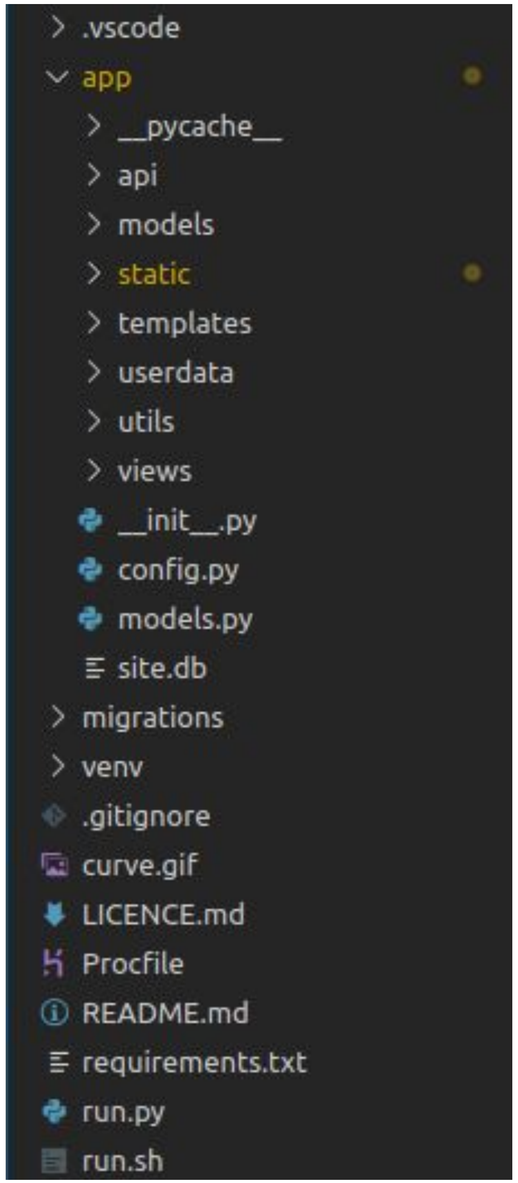
Библиотека JS - AmCharts - пример для работы с картами (Maps)



Структура проекта



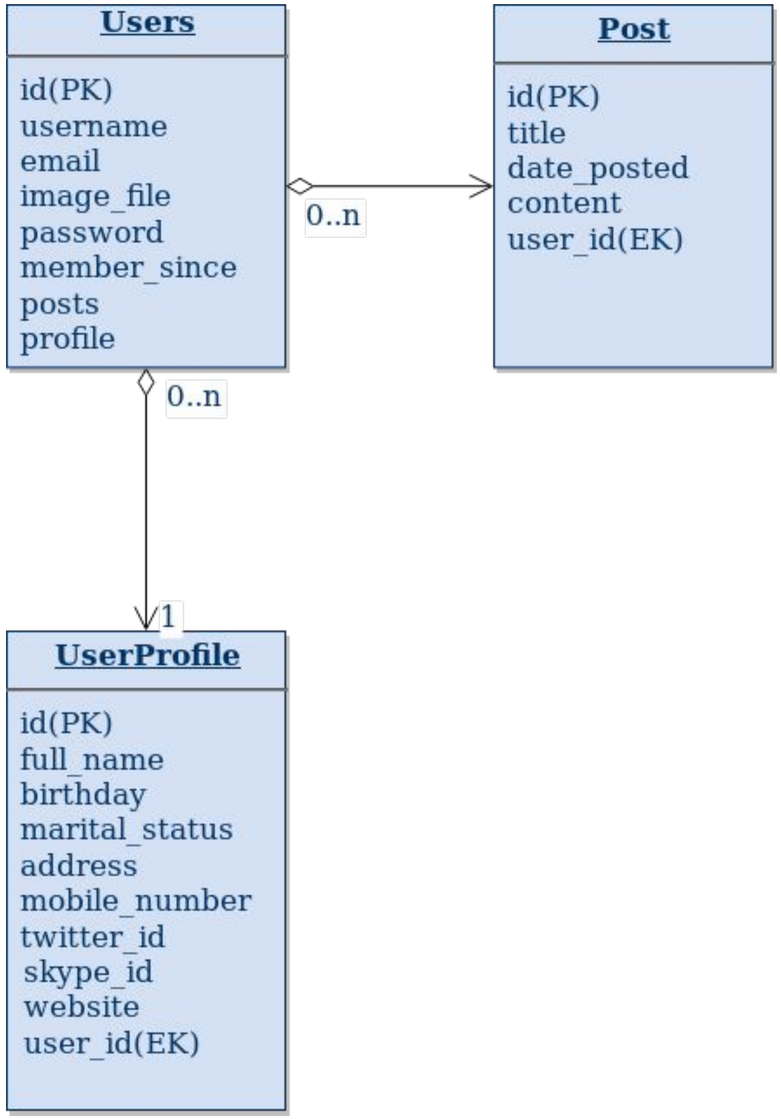
Структура проекта



Папка app - корневая папка с приложением Flask
- Папка api - REST API приложения, парсеры, источники данных
- Папка models - модели cases.pth, deaths.pth Нейронной сети
- Папка static - статичные файлы приложения(js, css, sass, картинки, лого)
- Папка templates - html темплейты и layout приложения
- Папка userdata - пользовательские данные(напр. PDF отчеты)
- Папка utils - вспомогательные Python скрипты и код
- Папка views - Blueprints и Views приложения - формы и вся логика приложения
- config.py - основная конфигурация приложения Flask
- models.py - описание таблиц и создание таблиц БД
- site.db - сама физическая БД, хранимая на диске / сервере

Папка migrations - папка с миграциями Базы Данных
Папка venv -виртуальное окружение Python
requirements.txt - список зависимостей(библиотек Python) приложения
Procfile - файл для деплоймента приложения на Heroku
run.py - Python скрипт для запуска приложения Flask локально
run.sh - bash скрипт для запуска приложения

База данных



База данных - реляционная SQLite 3, site.db
Для создания и управления БД используется Flask-SQLAlchemy

Три таблицы в Базе данных: Юзер, Профиль Юзера, Посты юзера

Добавление:

```
>>> me = User('admin', 'admin@example.com')
>>> db.session.add(me)
>>> db.session.commit()
```

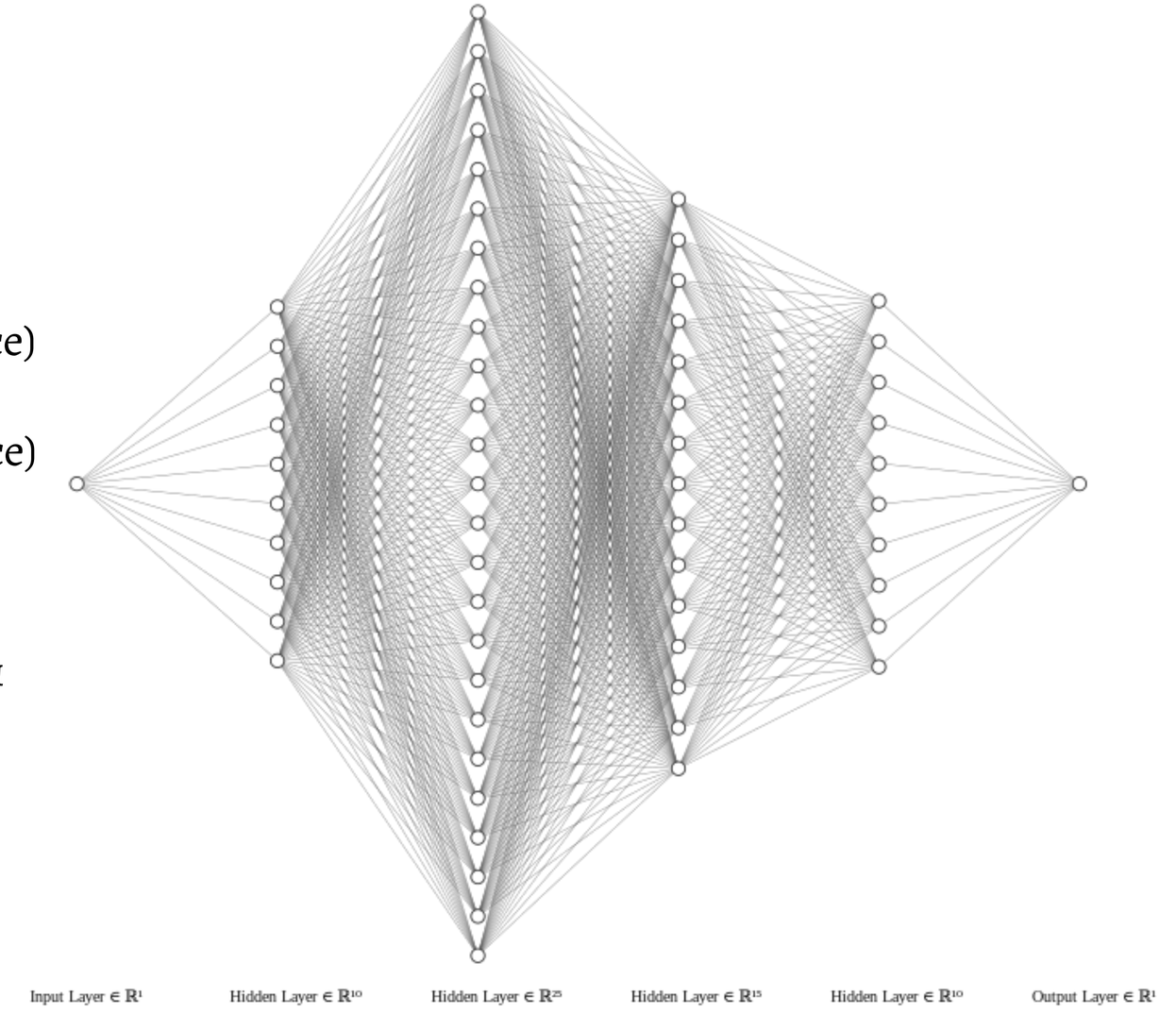
Запросы:

```
>>> user = User.query.filter_by(username='admin').first()
```

Архитектура нейросети

```
class Net(torch.nn.Module):
    def __init__(self, n_feature, n_hidden, n_output):
    super(Net, self).__init__()
    self.hidden = torch.nn.Linear(n_feature,
n_hidden).to(device)
    self.hidden1 = torch.nn.Linear(n_hidden, 800).to(device)
    self.hidden2 = torch.nn.Linear(800, 500).to(device)
    self.hidden3 = torch.nn.Linear(500, n_hidden).to(device)
    self.predict = torch.nn.Linear(n_hidden,
n_output).to(device)
```

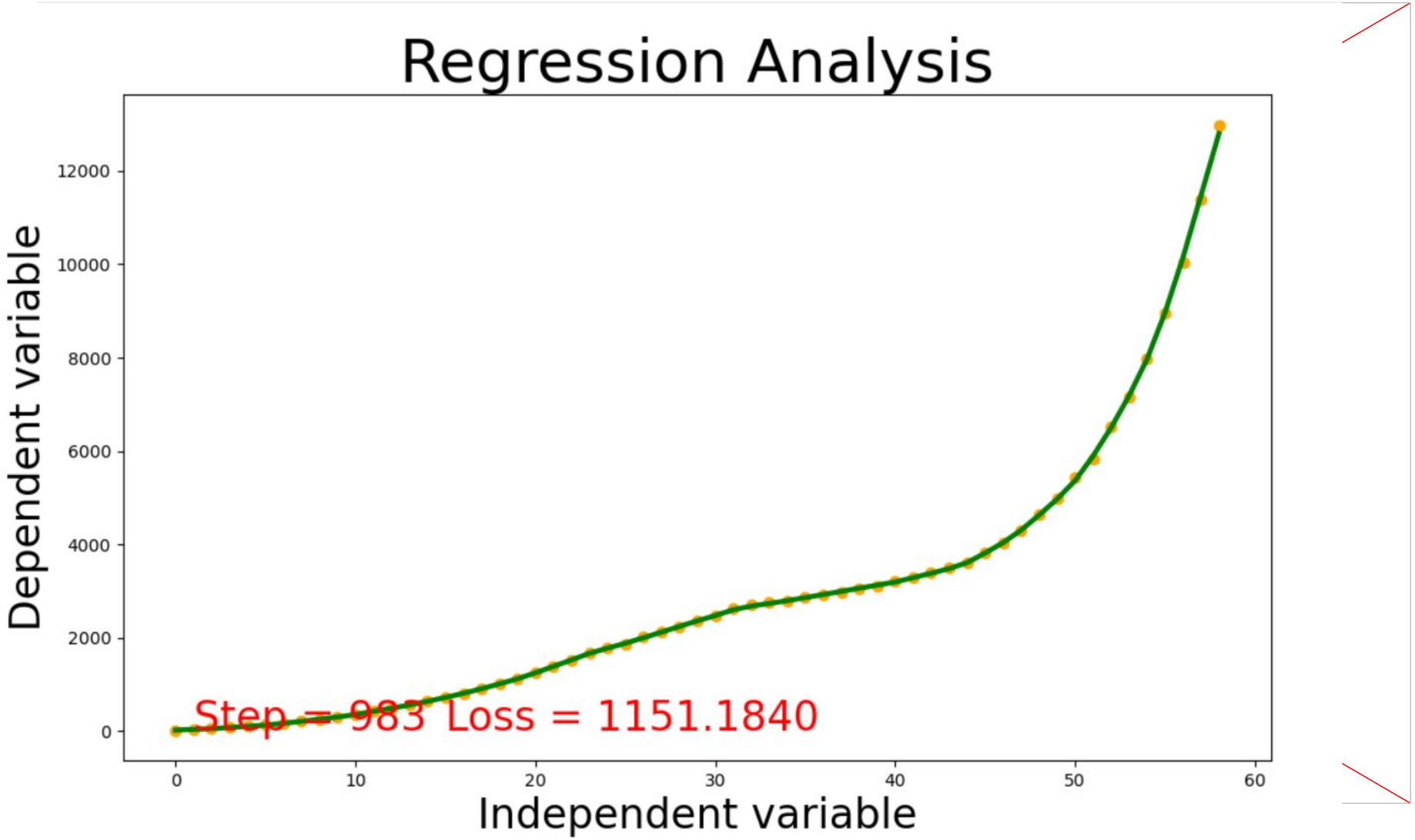
1 входной слой, 4 скрытых линейных слоя, 1 выходной слой



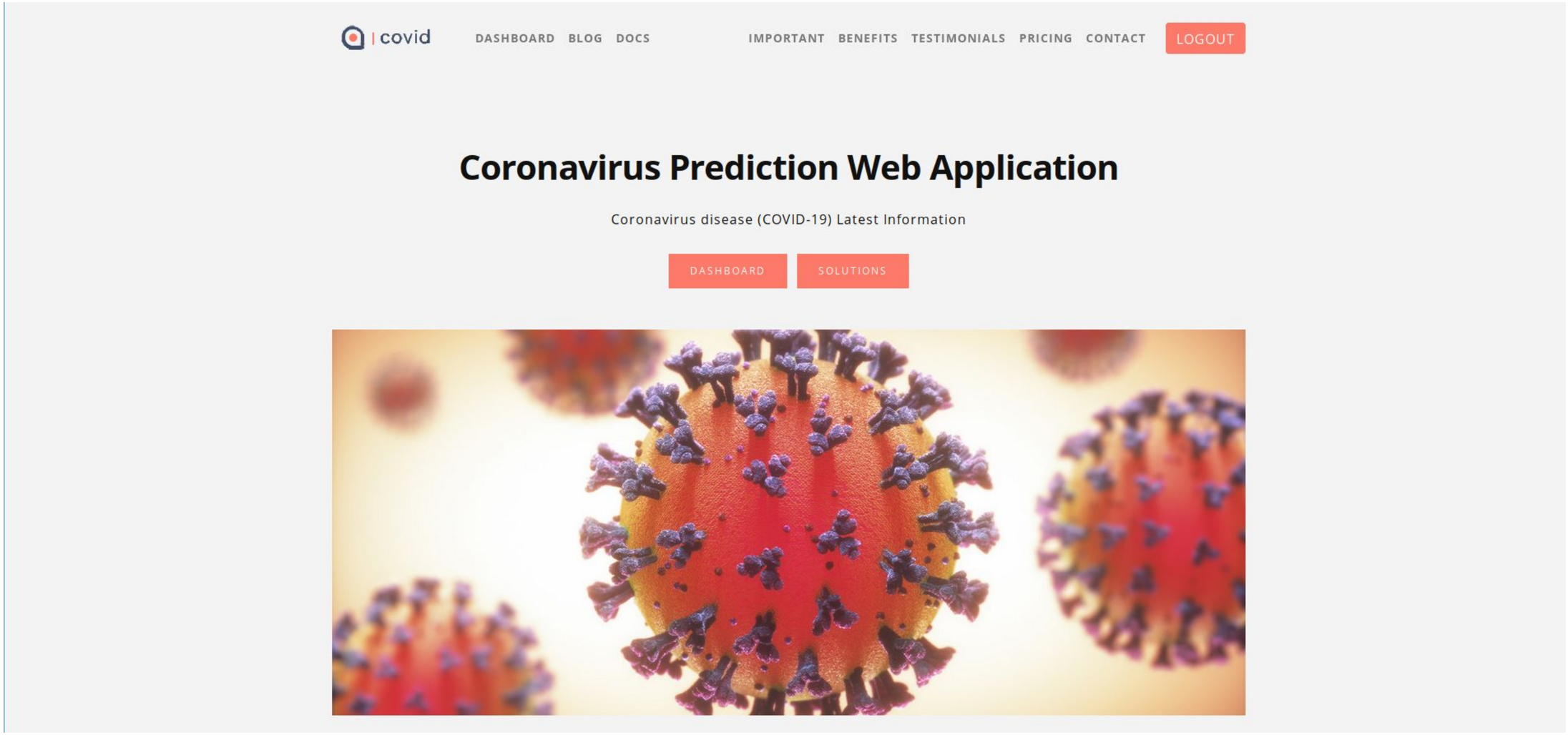
Работа с нейросетью

```
28  class Net(torch.nn.Module):
29      def __init__(self, n_feature, n_hidden, n_output):
30          super(Net, self).__init__()
31          self.hidden = torch.nn.Linear(n_feature, n_hidden).to(device)
32          self.hidden1 = torch.nn.Linear(n_hidden, 800).to(device)
33          self.hidden2 = torch.nn.Linear(800, 500).to(device)
34          self.hidden3 = torch.nn.Linear(500, n_hidden).to(device)  # hidden layer
35          self.predict = torch.nn.Linear(n_hidden, n_output).to(device)   # output layer
36
37      def forward(self, x):
38          x = F.relu(self.hidden(x))      # activation function for hidden layer
39          x = F.relu(self.hidden1(x))
40          x = F.relu(self.hidden2(x))
41          x = F.relu(self.hidden3(x))
42          x = self.predict(x)             # linear output
43          return x
```

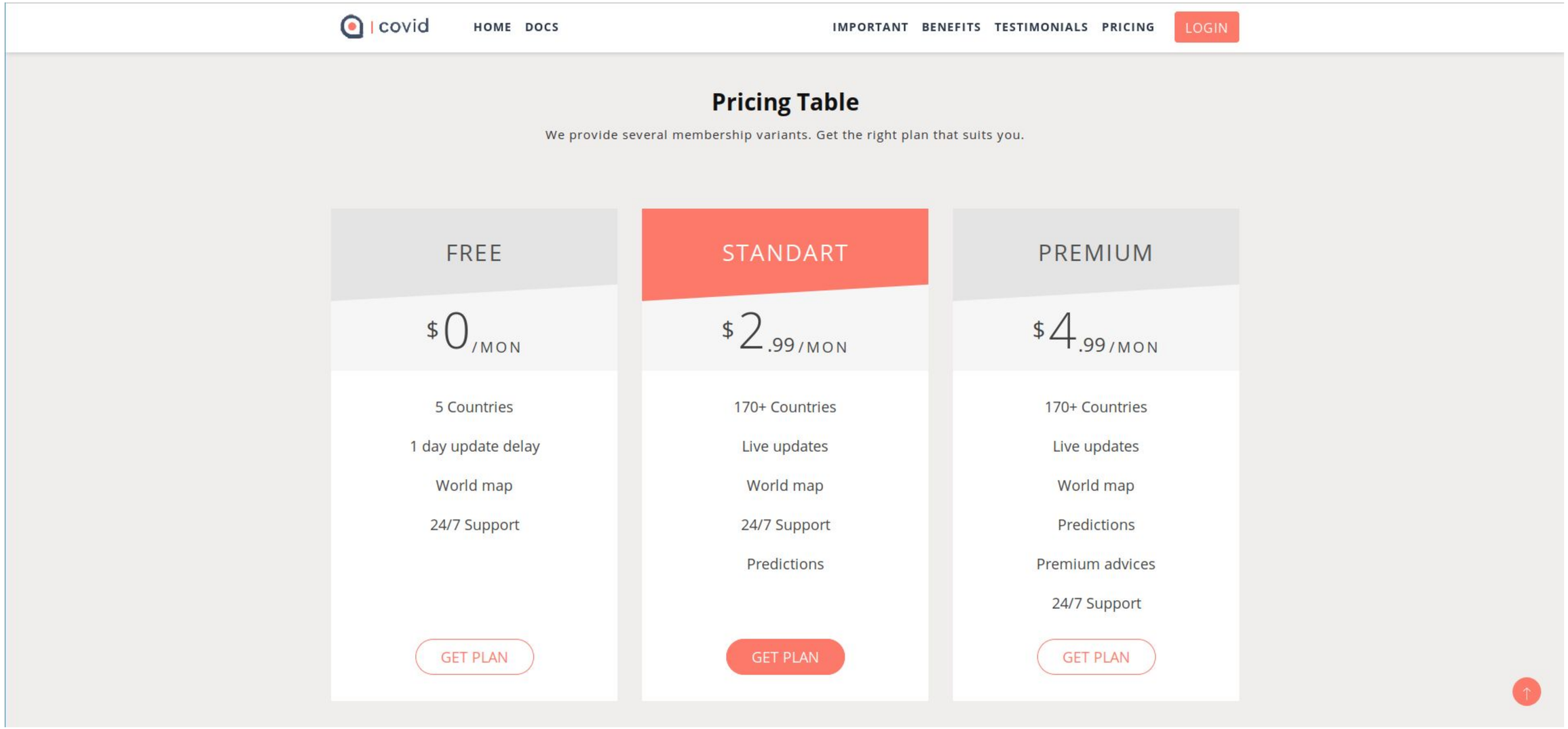
Работа с нейросетью



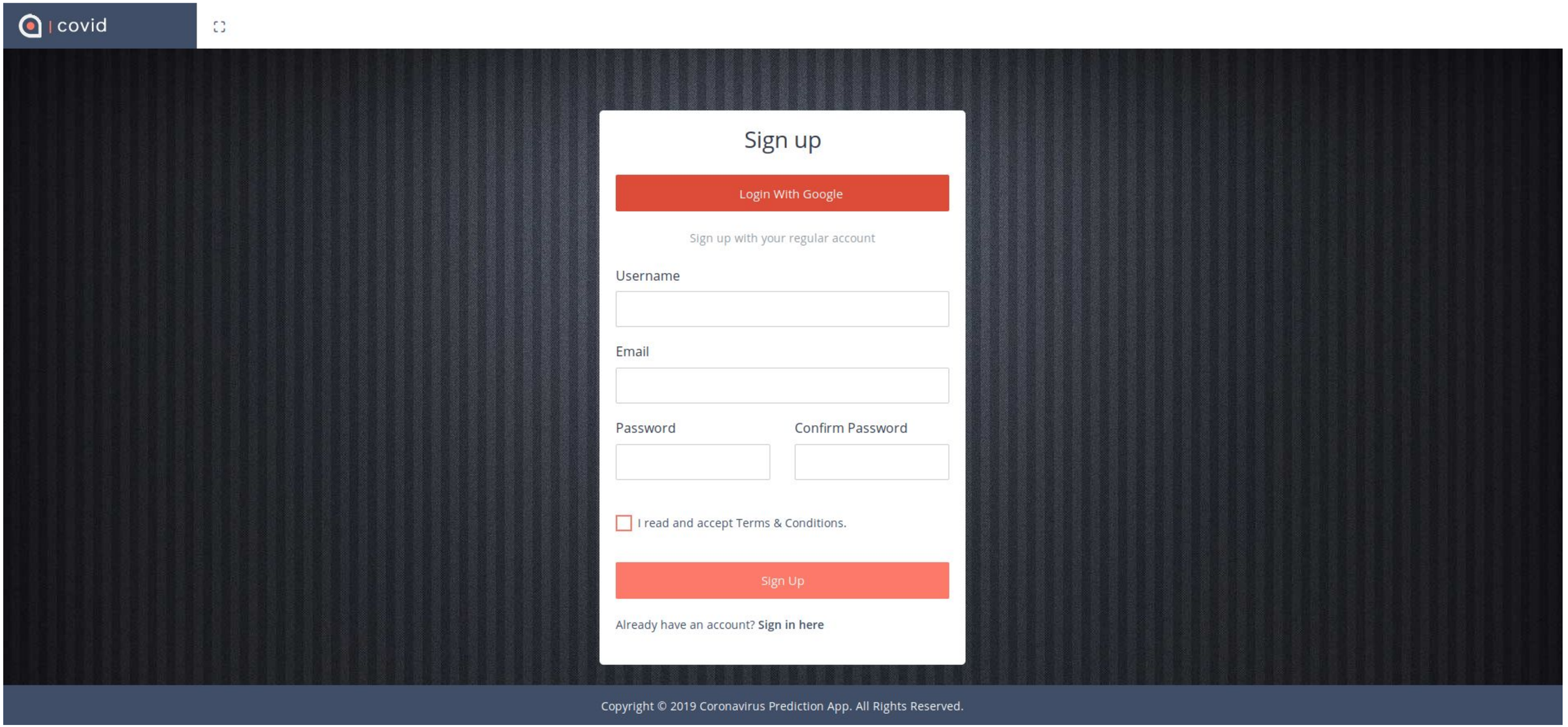
Сценарии работы приложения (Landing)



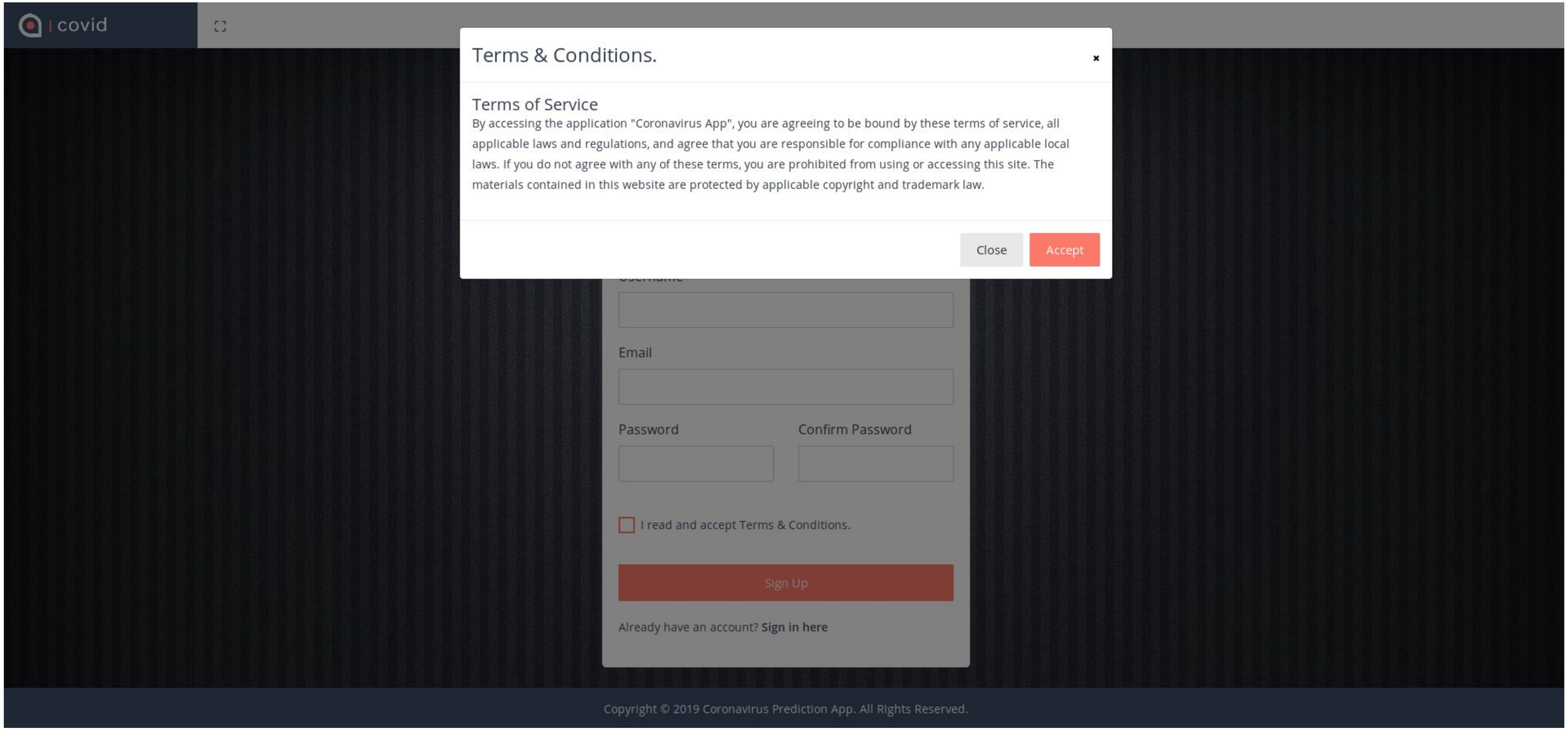
Сценарии работы приложения (Pricing)



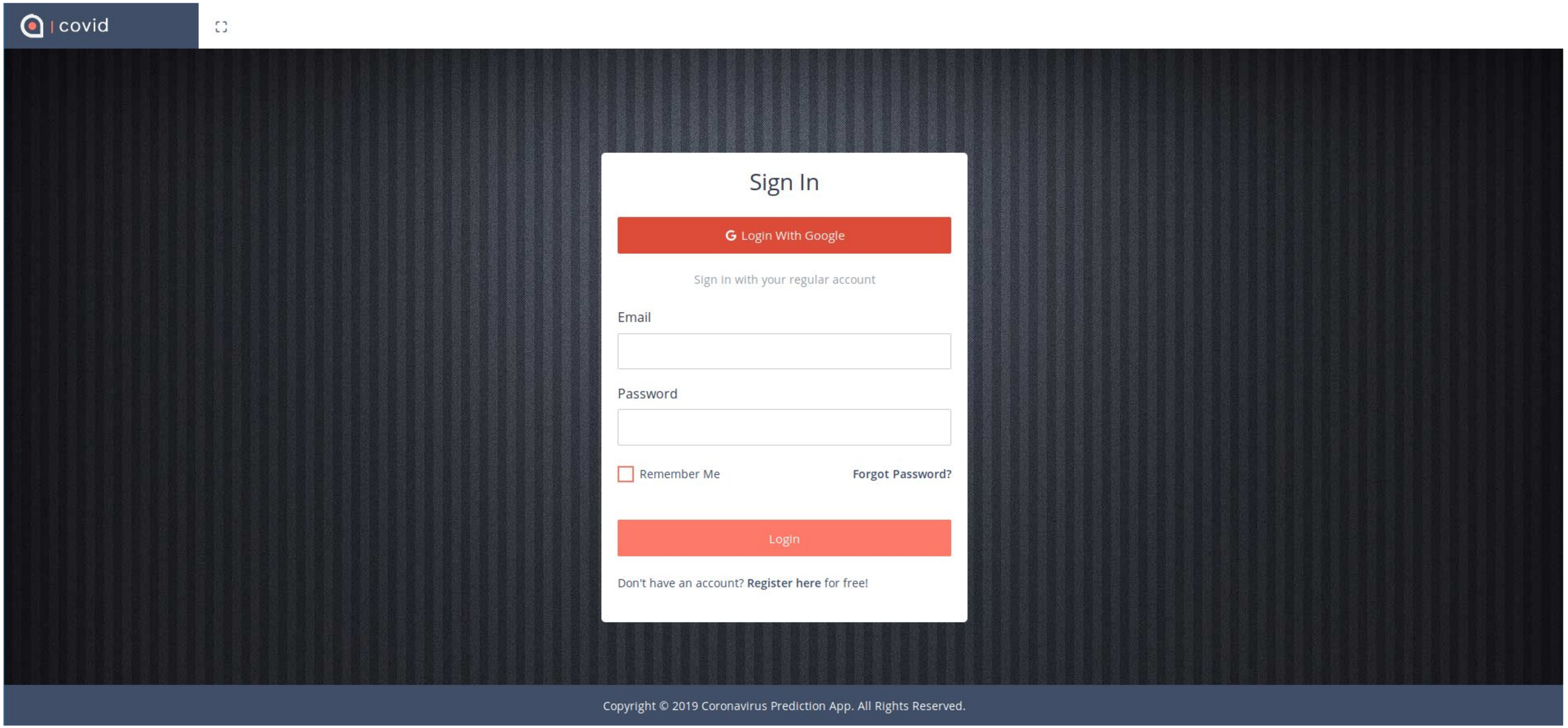
Сценарии работы приложения (Sign Up)



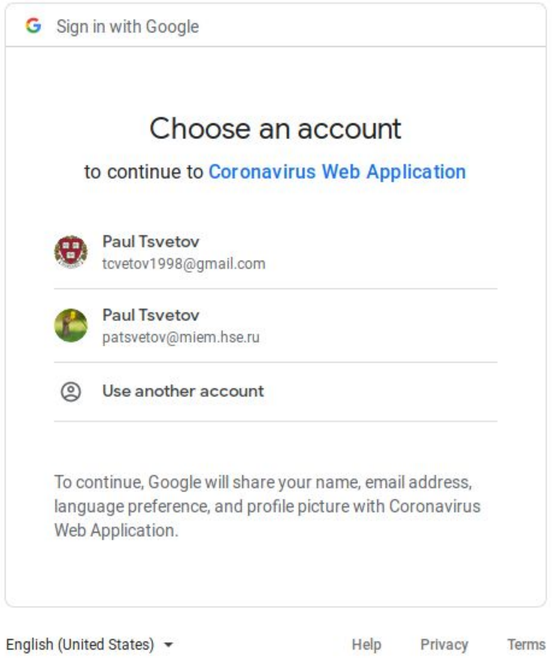
Сценарии работы приложения (Terms & Cond)



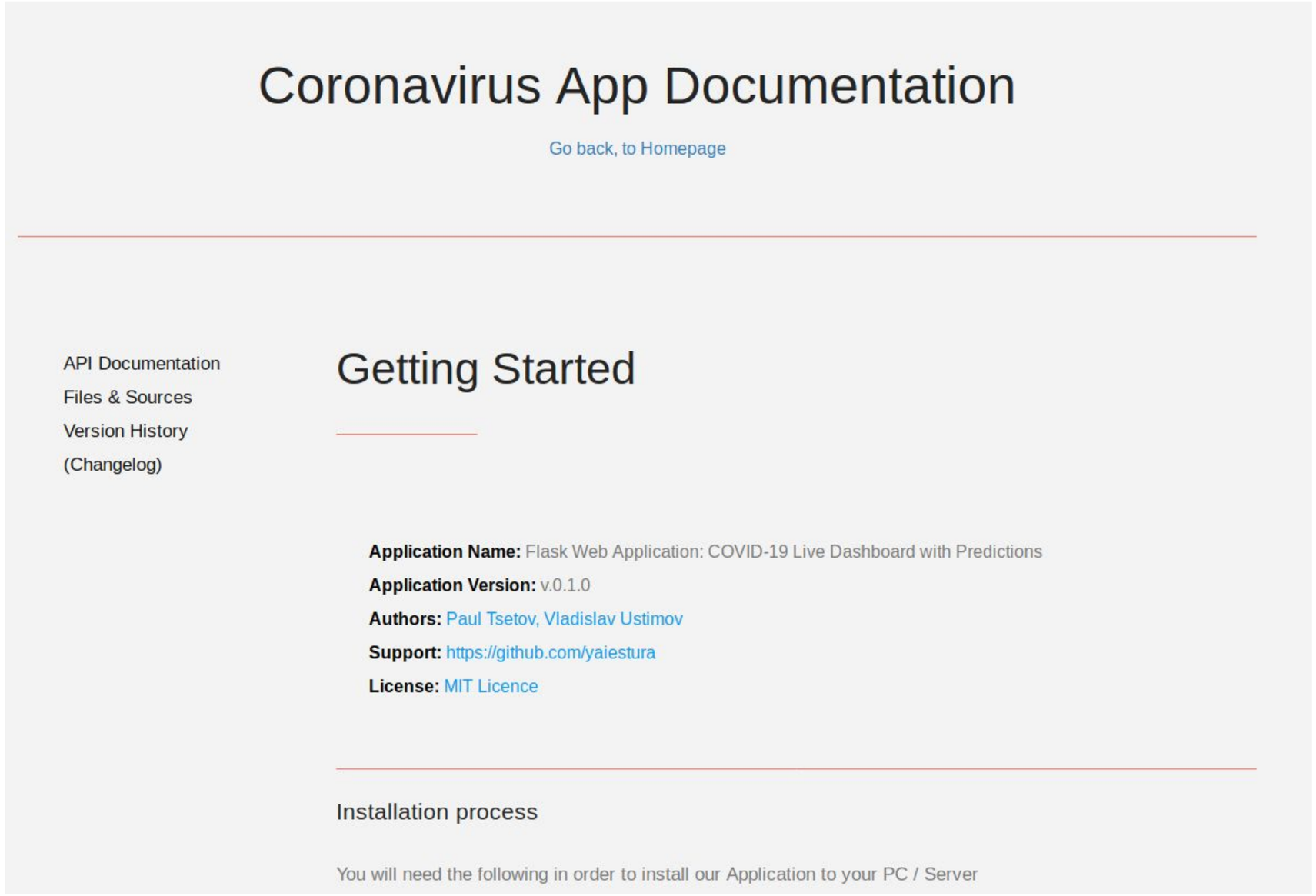
Сценарии работы приложения (Sign In)



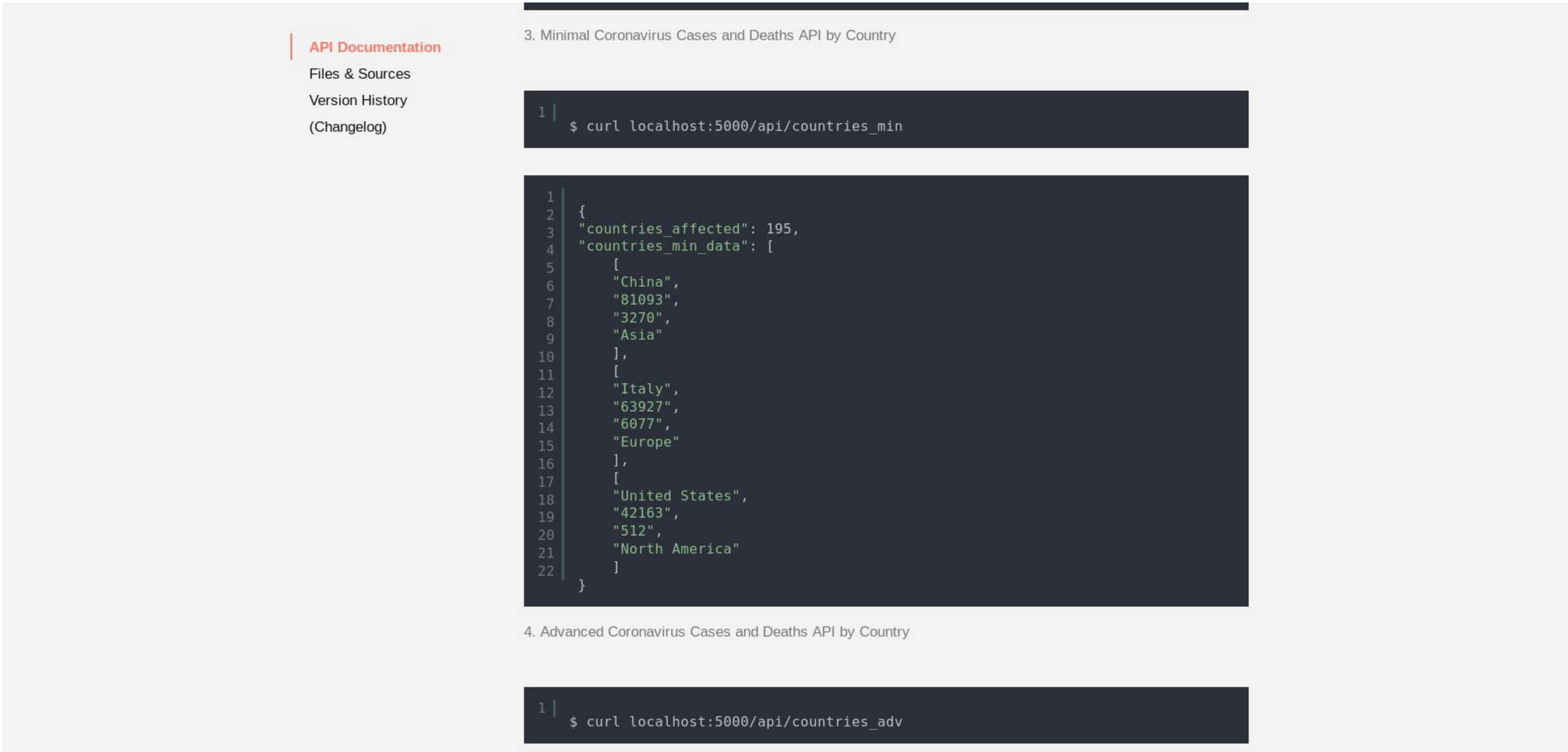
Сценарии работы приложения (Google Sign In)



Сценарии работы приложения (Документация)



Сценарии работы приложения (API)



API Documentation
Files & Sources
Version History
(Changelog)

3. Minimal Coronavirus Cases and Deaths API by Country

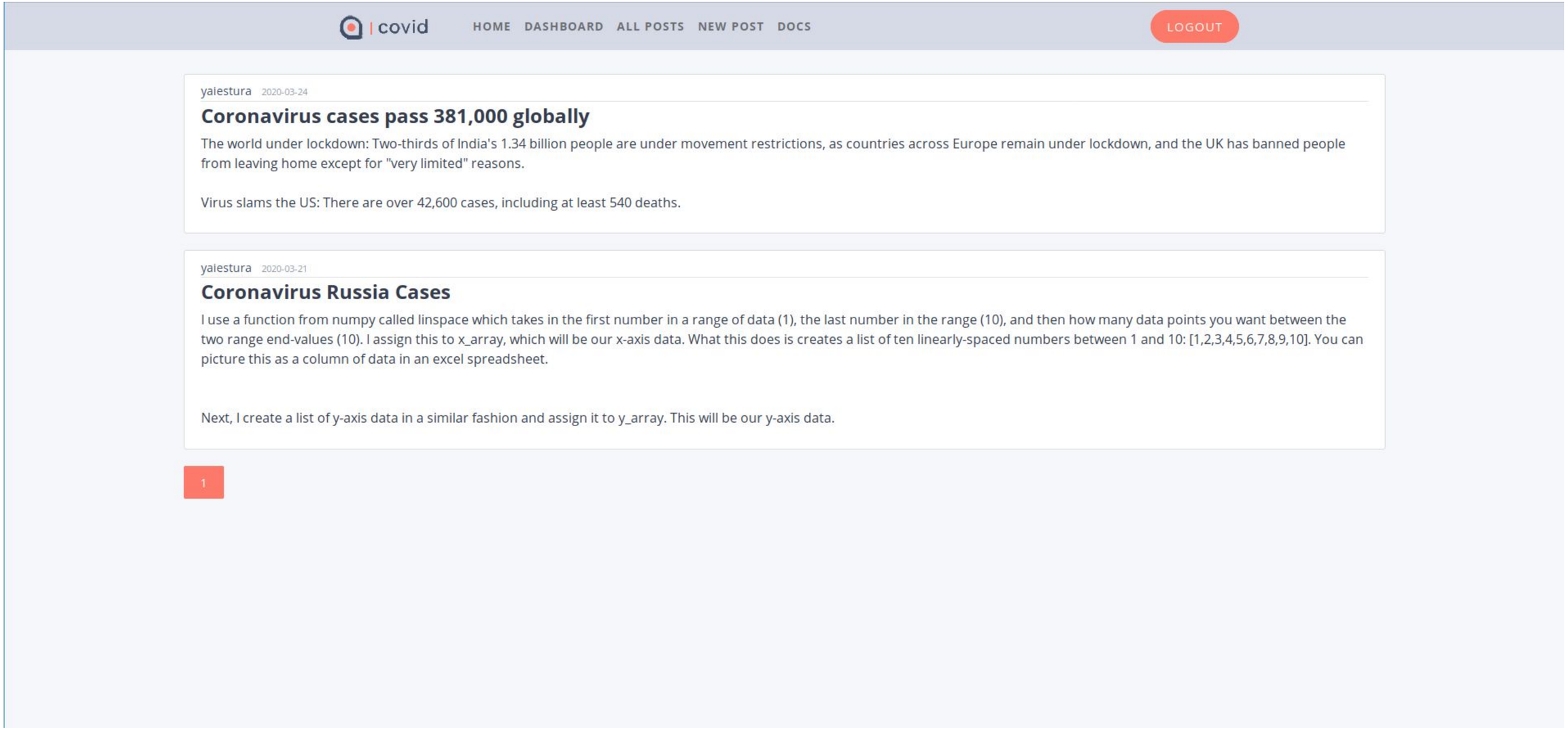
```
$ curl localhost:5000/api/countries_min
```


```
{
"countries_affected": 195,
"countries_min_data": [
    [
    "China",
    "81093",
    "3270",
    "Asia"
    ],
    [
    "Italy",
    "63927",
    "6077",
    "Europe"
    ],
    [
    "United States",
    "42163",
    "512",
    "North America"
    ]
}
```

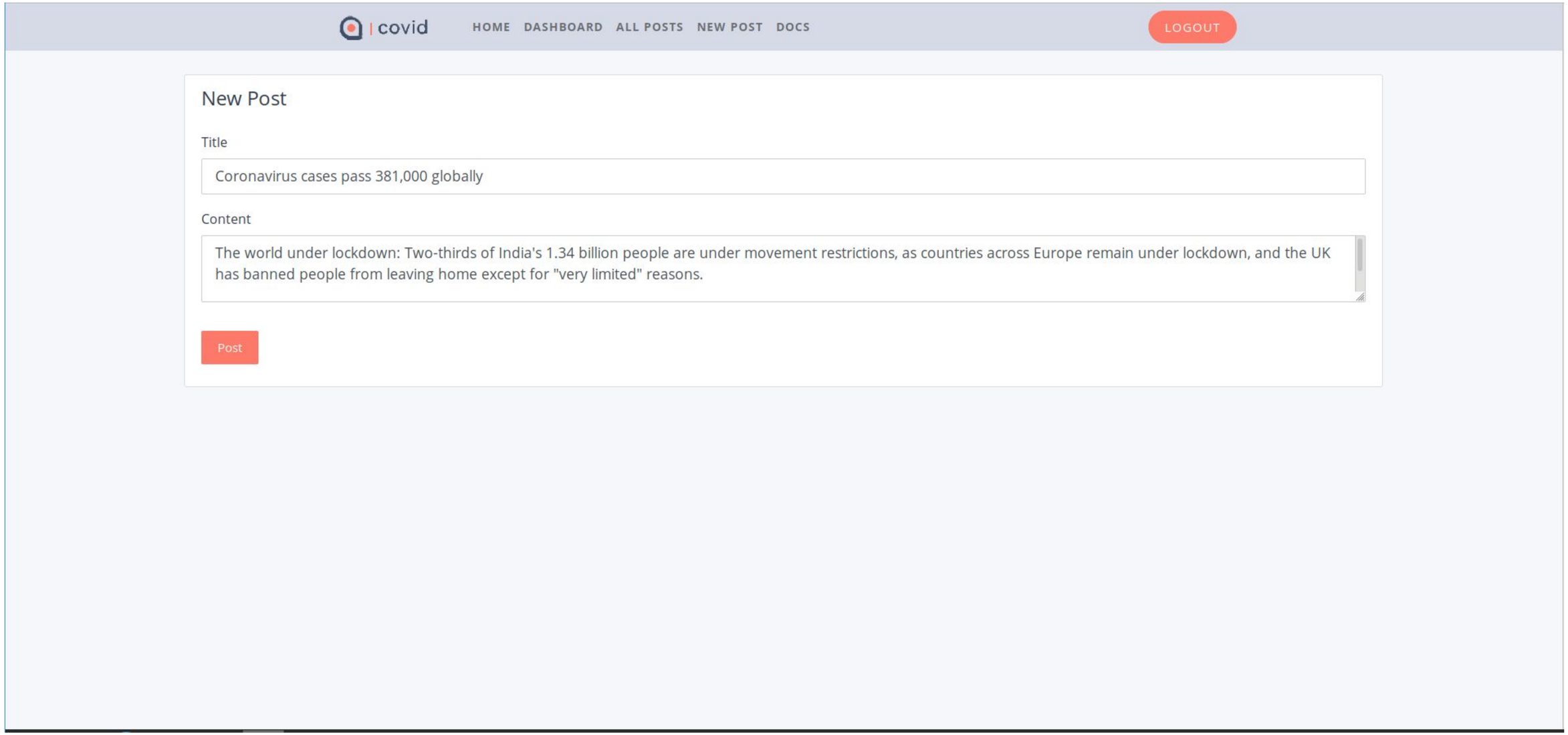

4. Advanced Coronavirus Cases and Deaths API by Country

```
$ curl localhost:5000/api/countries_adv
```

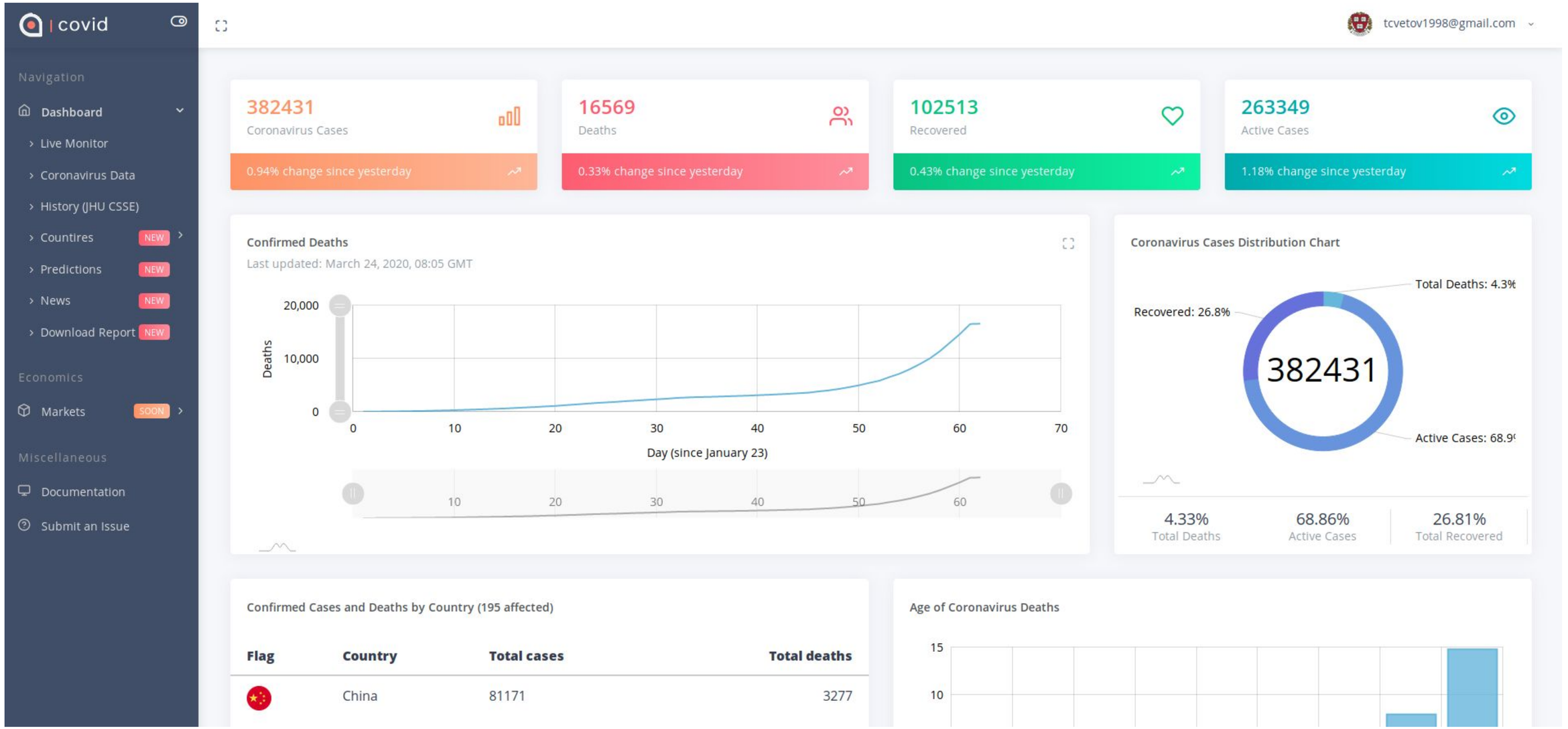
Сценарии работы приложения (COVID-19 Blog)



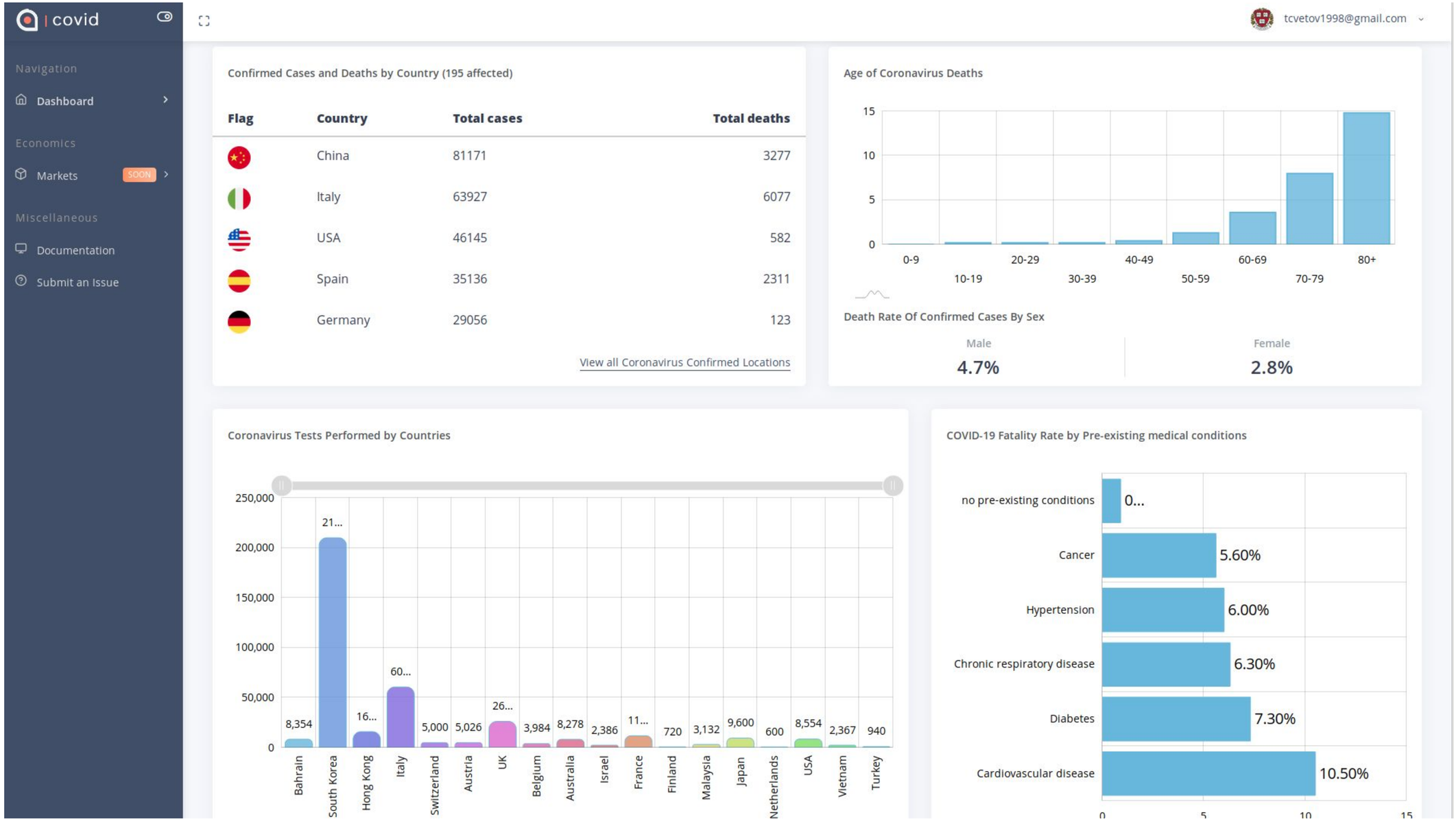
Сценарии работы приложения (Blog - Новый пост)



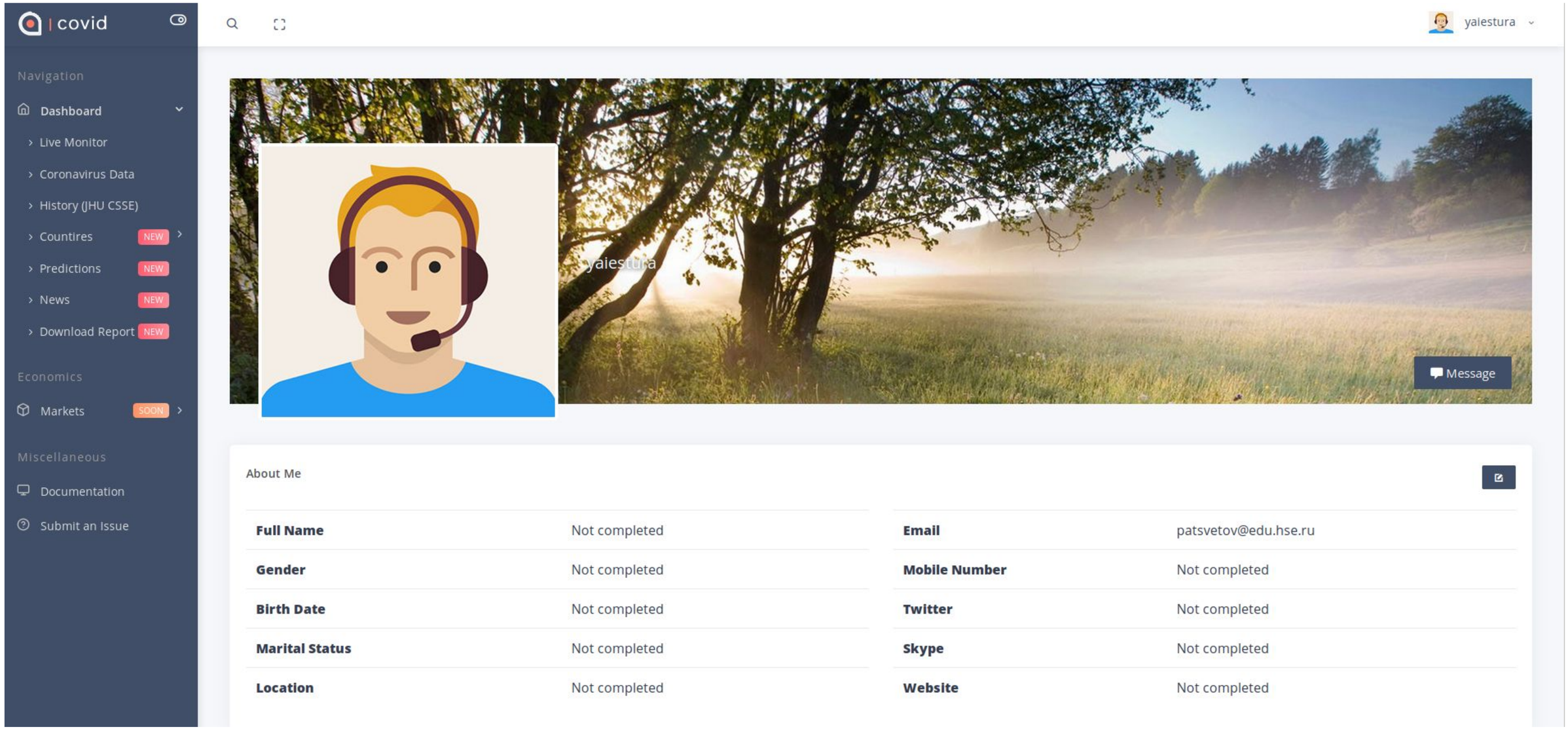
Сценарии работы приложения (Dashboard)



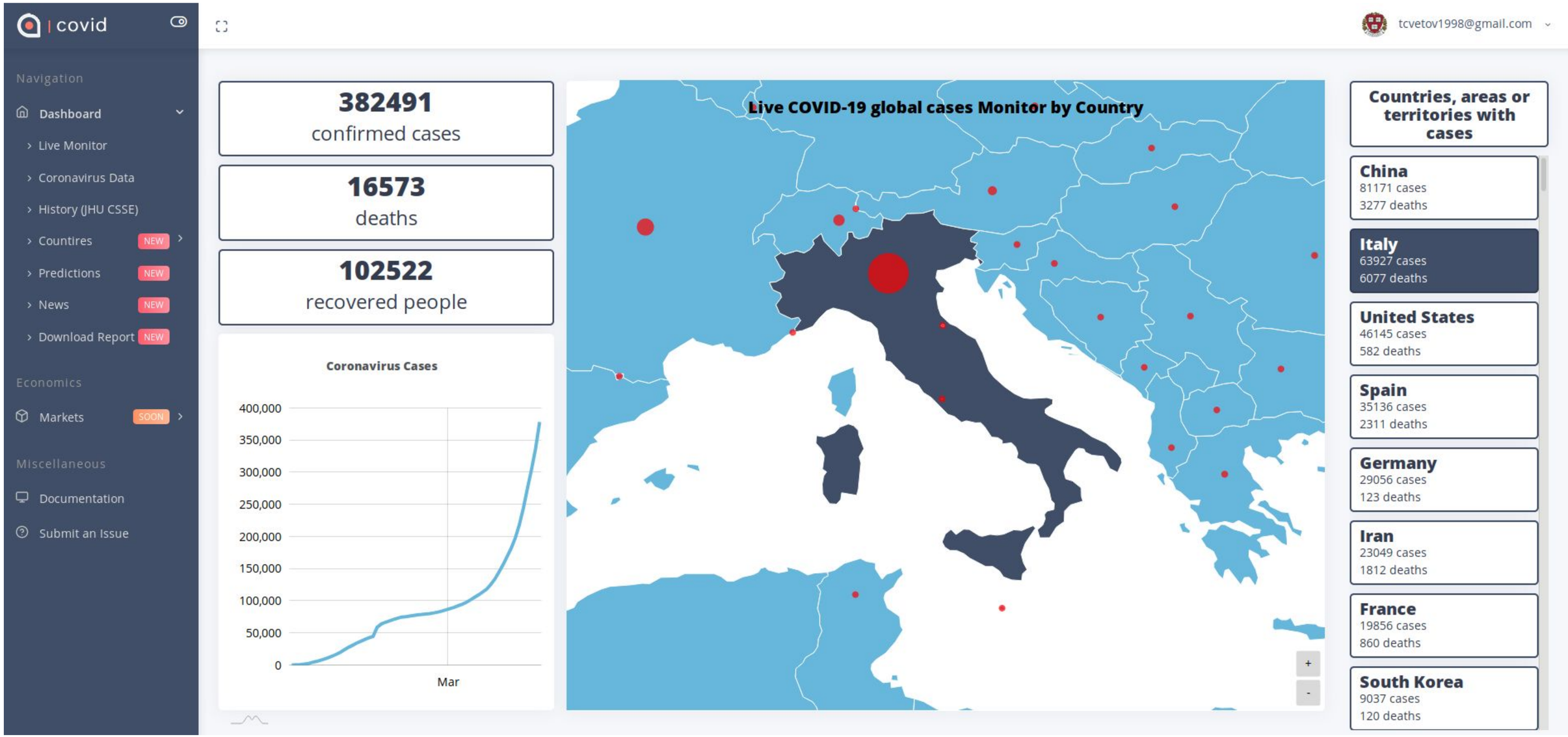
Сценарии работы приложения (Dashboard)



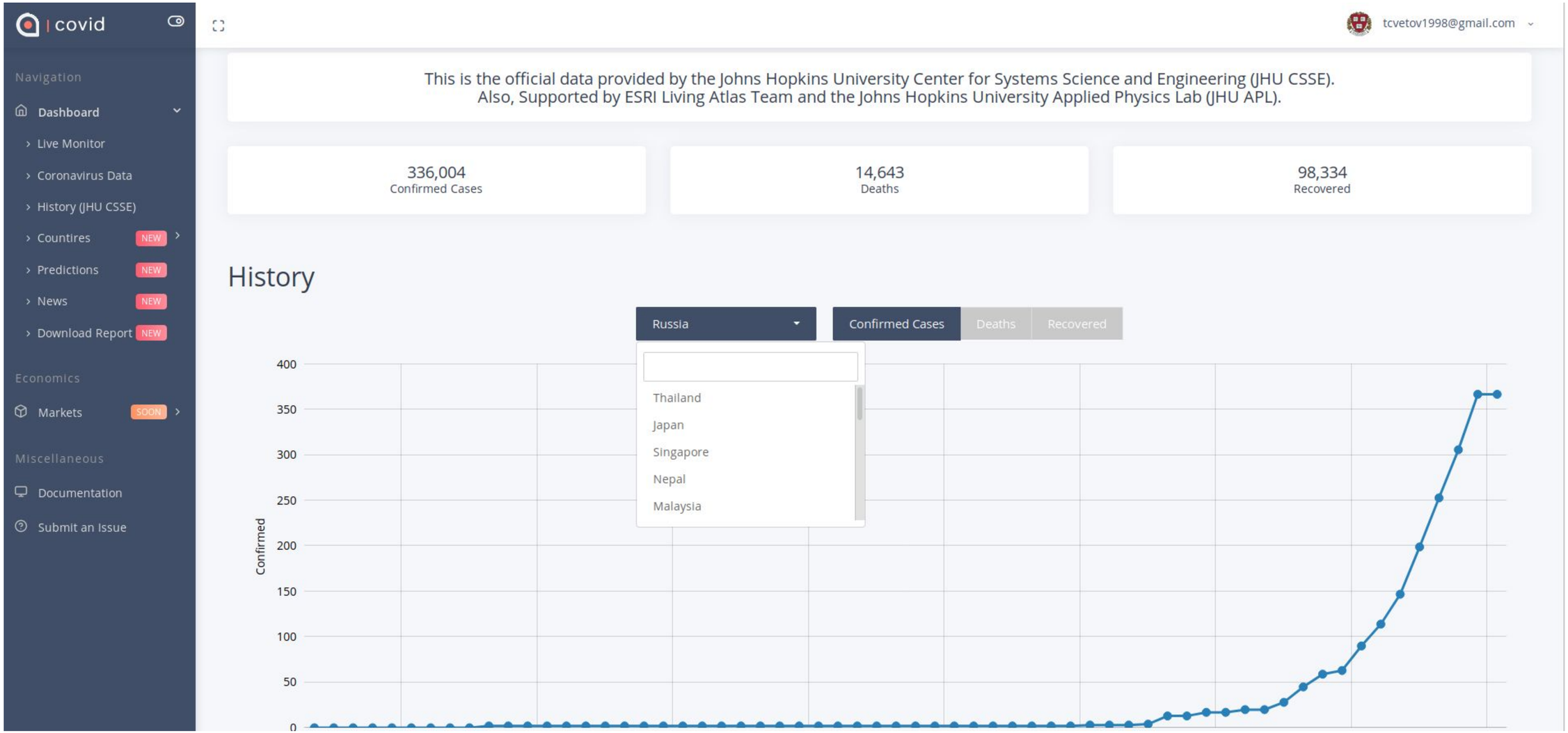
Сценарии работы приложения (Профиль)



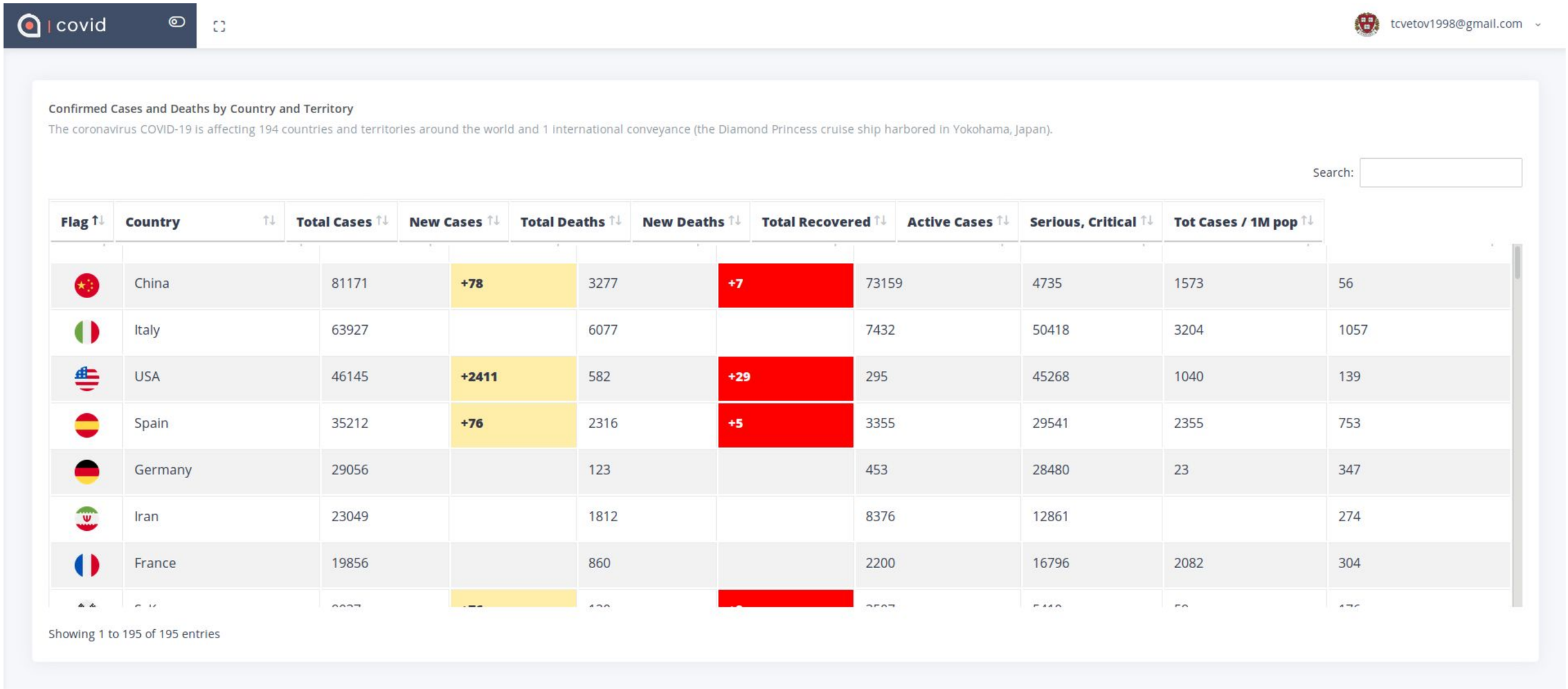
Сценарии работы приложения (Live Monitor)



Сценарии работы приложения (JHU Data History)



Сценарии работы приложения (Countries)



Сценарии работы приложения (Country - Italy)

Сценарии работы приложения (Country - Canada)

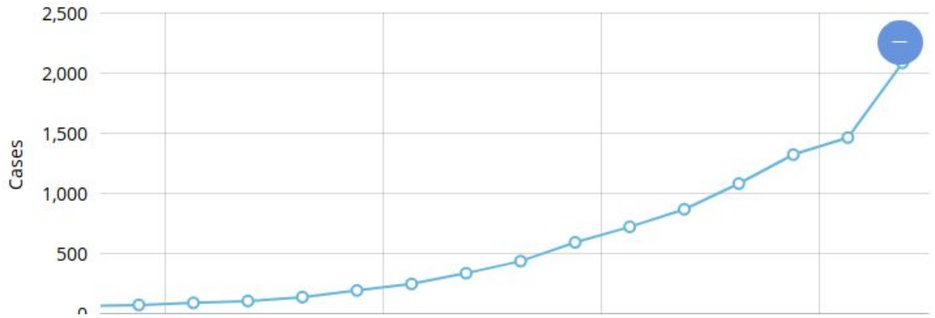
tcvetov1998@gmail.com

Country: **Canada**

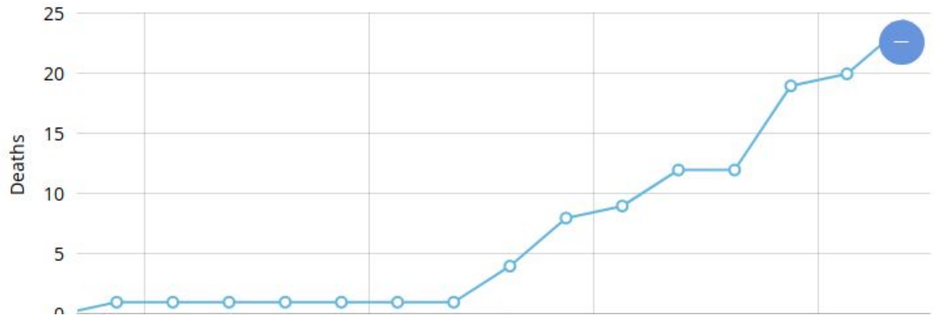
Last updated: March 24, 2020, 08:22 GMT

Population:	37,742,154
Land Area(km^2):	9,093,510
Density(P/km^2):	4
Total cases:	2091 (0.547% of world's)
Total deaths:	24 (0.145% of world's)
Total recovered:	320

Canada Coronavirus Cases Plot

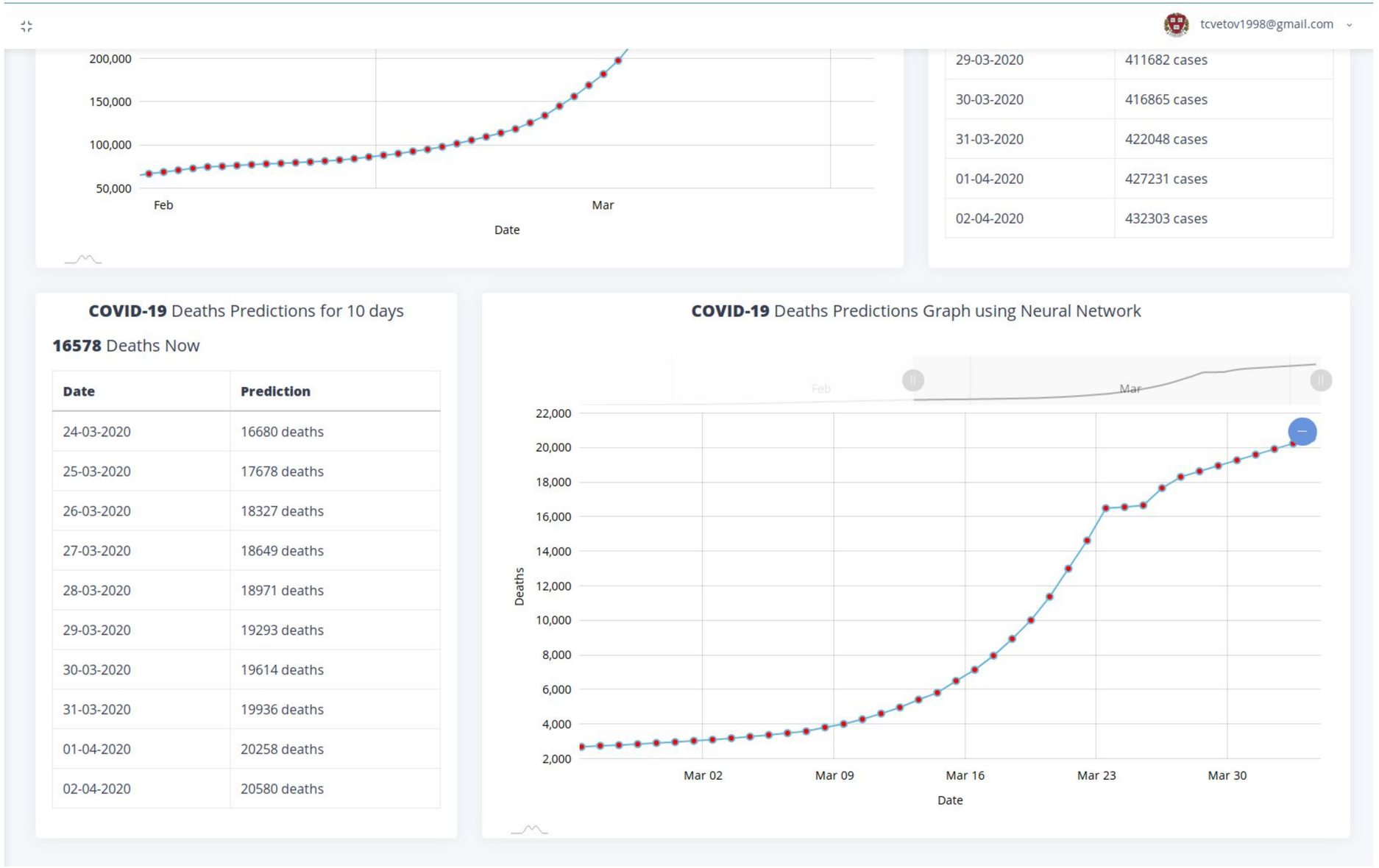


Canada Coronavirus Deaths Plot



Сценарии работы приложения (Predictions)

Сценарии работы приложения (Predictions)



Сценарии работы приложения (News)

Сценарии работы приложения (PDF-отчет)

COVID-19 Main Live Statistics

COVID-19 Total Cases: 382741

COVID-19 Total Deaths: 16578

COVID-19 Total Recovered: 102522

COVID-19 Demographics Data:

Fig.1. Age of Coronavirus Deaths:

Age	Death Rate of Confirmed Cases
80+	14.8%
70-79	8.0%
60-69	3.6%
50-59	1.3%
40-49	0.4%
30-39	0.2%
20-29	0.2%
10-19	0.2%
0-9	no fatalities

Fig.2. Sex of Coronavirus Deaths:

Sex	Death Rate of Confirmed Cases	Death Rate of All Cases

Сценарии работы приложения (PDF-отчет)

COVID-19 Confirmed Cases and Deaths by Country, Territory, or Conveyance

Country	Total Cases	New Cases	Total Deaths	New Deaths	Total Recovered	Active Cases	Serious	Cases 1M Pop
China	81171	+78	3277	+7	73159	4735	1573	56
Italy	63927		6077		7432	50418	3204	1057
USA	46145	+2411	582	+29	295	45268	1040	139
Spain	35212	+76	2316	+5	3355	29541	2355	753
Germany	29056		123		453	28480	23	347
Iran	23049		1812		8376	12861		274
France	19856		860		2200	16796	2082	304
S. Korea	9037	+76	120	+9	3507	5410	59	176
Switzerland	8795		120		131	8544	141	1016
UK	6650		335		135	6180	20	98
Netherlands	4749		213		2	4534	435	277
Austria	4517	+43	25	+4	9	4483	16	502
Belgium	3743		88		401	3254	322	323
Norway	2625		11	+1	6	2608	41	484
Australia	2136	+249	8	+1	118	2010	11	84
Canada	2091		24		320	1747	1	55
Portugal	2060		23		14	2023	47	202
Sweden	2059	+13	33	+6	16	2010	110	204
Brazil	1924		34		2	1888	18	9
Israel	1656	+214	1		49	1606	31	191
Turkey	1529		37			1492		18
Malaysia	1518		15	+1	159	1344	57	47

Участие в проекте

Тип работы	Цветов Павел	Устимов Владислав
Разработка дизайна и макета приложения	+	+
Инфографика, графика, карты, JS скрипты	+	
Разработка HTML страниц	+	
JavaScript, CSS Developer	+	
Парсер данных, API	+	+
Python Developer	+	+
База данных	+	+
Построение и тренировка нейронной сети		+
Внедрение нейронной сети в существующую веб-платформу		+
Тестирование	+	+
Документация	+	+

Возможные доработки и нововведения

1. Нейросеть - влияние коронавируса на экономику, американские рынки, акции
2. Инфографика, статистика по экономике в режиме Live
3. Несколько различных моделей предсказания для Коронавируса
4. Система уведомлений в приложении(Notifications)
5. Форма для оплаты для доступа в приложение)
6. Разделение пользователей на группы(администраторы, обычные, и т.д.)
7. Кастомный и расширенный профиль пользователя
8. Flask-Mail - отправка приложением сообщений на почту, восстановление пароля
9. Рефакторинг кода, улучшение кода, избавление от зависимостей