МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. ЛОМОНОСОВА
ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ
Курс основы WEB-технологий

СОЗДАНИЕ СТОХАСТИЧЕСКИХ МОДЕЛЕЙ, ОПИСЫВАЮЩИЙ РАСПРОСТРАНЕНИЕ ВИРУСА COVID-19 В РАЗЛИЧНЫХ УСЛОВИЯХ

Курсовая работа
студента 214 группы
Кудряшова А. Д.

Преподаватель:
Алексеев А. А.

Москва - 2020

Оглавление

Введение

Covid-19 - одна из основных проблем, с которой столкнулось современное человечество. Ежедневно мы видим все новые и новые цифры зараженных и умерших людей. До этого подобные случаи были описаны только в фантастических фильмах, которые мы все так охотно смотрели и обсуждали. Конечно, было много разговоров о том, что с нашим миром что-то подобное рано или поздно обязательно случится, но никто не думал, что так быстро и настолько сильно.

Обстановка меняется ежедневно, можно сказать ежечасно. Поэтому важно уметь быстро приспосабливаться к современным реалиям нашей жизни. В противном случае люди не смогут адекватно отреагировать и обезопасить себя, что может привести к печальным последствиям.

В контексте вышесказанного очень важно понимать, как вирус распространяется. В данной работе представлено web-приложение, которое содержит различные модели, наглядно показывающие, как быстро происходит распространение вируса в различных условиях, а также содержит общую статистику активности пользователей во время пандемии. Основная цель приложение носит информативный характер, оно помогает лучше понять процесс распространения вируса в мире и в частности в Москве.

Глава I

Основная часть

1. Исследование ситуации в мире с Covid-19.

Перед тем как преступить к постановке задачи, необходимо было исследовать реальную ситуацию в мире с Covid-19. Оказалось, что на сегодняшний день существует довольно много интернет-ресурсов, агрегирующих данные по Covid-19 [1], [8], [9]. Наиболее точную картину для российских реалий собирают общеизвестные ресурсы, такие как [10], [12].
Но, на самом деле, если пренебречь локальными различиями, общая картина для любой странны/города/поселка примерно одинакова. Динамика распространения вируса представляет собой экспоненциальную кривую. На рисунке 1 показана динамика распространения вируса в мире по дням.

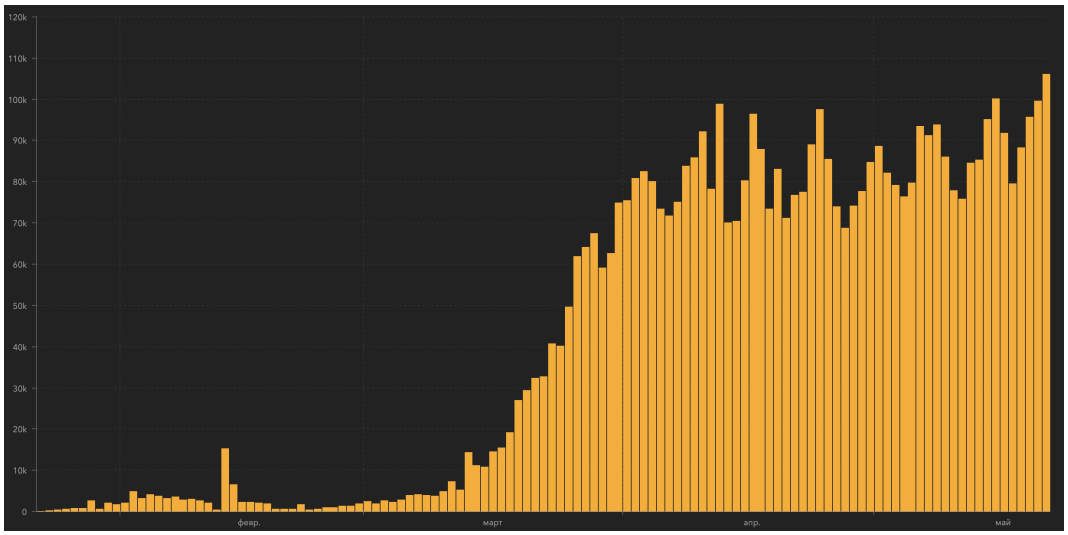

Рис. 1: Динамика распространения вируса в мире по дням. Каждому столбцу соответствует количество выявленных зараженных [8].

Так или иначе, все ресурсы показывают схожую (с точностью до цифр) динамику распространения. Моей целью было построить на основе реальных данных элементарную модель, на физическом уровне объясняющую такую форму кривой.

2. Выбор модели. Постановка задачи.

В работе я преследовал цель проиллюстрировать и наглядно показать процесс передачи вируса от человека к человеку. Для описания этого процесса была выбрана модель шариков (частиц), взаимодействующих друг с другом в замкнутом пространстве и подчиняющихся закону сохранения импульса и энергии. Эта модель может описывать многие реальные сценарии распространения вируса, при этом требуя минимальных процессорных мощностей. Что наилучшим образом подходит для web-приложения.
В качестве средства реализации задачи был выбран framework Django, который поддерживает патерн проектирования MVC. В качестве серверного языка был выбран python, как

самый простой и в то-же время функциональный язык. Для простоты создания моделей и динамичной отрисовки графиков на клиентской части был выбран язык JavaScript.

2.1. Описание моделей.

Web-приложение представляет собой сайт, с различными моделями распространения вируса между людьми. Для иллюстрации передачи вируса от человека к человеку используется, так называемая, линейная модель распространения с одной степью свободы у частиц (рис. 2a). В ней учтена только возможность взаимодействия между частицами по одной координате. На рисунке 2b показана та-же модель, только теперь в ней учитывается возможность людей выздоравливать. Ниже строиться график, характеризующий динамику процесса. Красным цветом показано количество зараженных вирусом в конкретный момент времени, фиолетовая кривая соответствует количеству выздоровевших людей в каждый момент времени. Надо сказать, что время, через которое человек должен выздороветь, определяться каждый раз случайным образом, что придает каждому эксперименту случайный характер. Тем не менее, процесс не совсем случаен, так как интервал, в котором меняется время выздоровления человека, ограничен. Таким образом, каждый запуск модели носит случайный характер. Но, тем не менее, имеет общие черты с другими запусками и в общей сложности результат работы модели также меняется в ограниченном диапазоне.

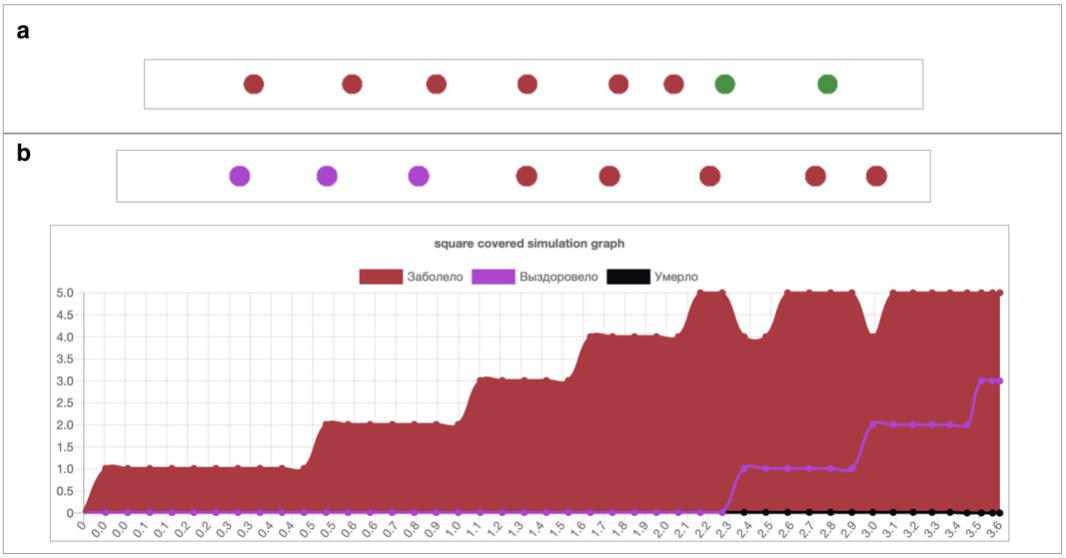

Рис. 2: Линейная модель распространения вируса. На **a** показана простейшая линейная модель, описывающая распространение вируса Красным отмечены заболевшие люди, зеленым - здоровые. На рисунке **b** показана та-же модель, но с учетом способность людей выздоравливать. Ниже строиться график, характеризующий динамику процесса. Красным цветом показано количество зараженных вирусом в конкретный момент времени, фиолетовая кривая соответствует количеству выздоровевших людей в каждый момент времени.

В приложении также присутствуют двумерные аналоги показанных выше моделей. К ним также строятся графики, иллюстрирующие процесс.
Модель, изображенная на рисунке 3, описывает процесс более приближенный к реальной

ситуации. Здесь учтена смертность от вируса. Модель построена по примеру реальной картины, процент смертности (относительно общего числа зараженных) практически соответствует реальным данным, которые наблюдаются на данный момент в г. Москве.
Также в модели учтена вероятность взаимодействия между "условно умершими"и остальными людьми. Вероятность характеризуется размерами шарика: чем меньше шарик, тем меньше вероятность его взаимодействия с остальными объектами. В момент "смерти"объекта он останавливается и его размеры резко уменьшаются. Дальнейшее его движение может быть обусловлено только взаимодействием с другими объектами. Как и в случае со временем до выздоровления, время до смерти также определяться случайным образом. Но, несмотря на свой случайны характер, оно четко соотноситься с временем до выздоровления, именно благодаря этому общий процент смертей удается приблизить к конкретным значениям. В данном случае, московским.

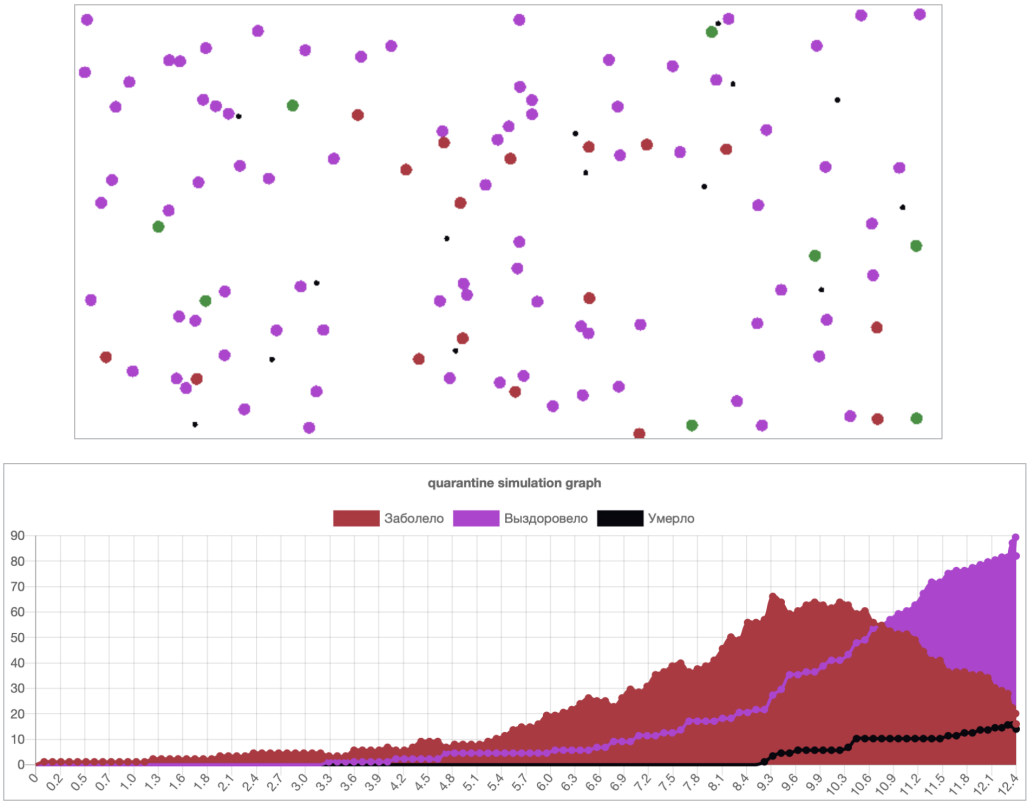

Рис. 3: Плоская модель распространения вируса. На рисунке показана модель с учетом выздоровления и смертности людей. Красным отмечены заболевшие люди, зеленым - здоровые, а черным - умершие. Ниже строиться график, характеризующий динамику процесса. Красным цветом показано количество зараженных вирусом в конкретный момент времени, фиолетовая кривая соответствует количеству выздоровевших людей в каждый момент времени, а черная - количеству умерших людей.

Видно, что скорость распространения вируса растет по экспоненциальному закону, по форме напоминающим рисунок 1. Таким образом, модель приближенно описывает реальную динамику распространения вируса в мире. Следовательно результаты работы этой модели можно считать приемлемыми.

В приложении также показана модель, учитывающая карантинные меры, которые развитые страны применяют для борьбы с эпидемией. В модели случайным образом выбирается количество областей, в которых будет введен карантин. Таким образом, частицы (люди), оказавшиеся в этой области, лишаются возможности передвигаться, но, все-же, по прежнему могут взаимодействовать с другими частицами (людьми). Тем самым я моделирую условные походы в магазин, мелкие встречи, которые неизбежно возникают даже у тех людей, которые находятся на карантине. Тем не менее, фиксация положения людей, как оказываться, существенно влияет на динамику распространения вируса (рис. 4b).
На вкладке "Собственная модель"есть возможность задать начальные параметры для модели, там самым воспроизводя ту или иную ситуацию. Отдельного внимания заслуживает возможность создавать частицы с разным радиусом. Это сделано для того, чтобы дать возможность ввести разные группы частиц с разной вероятностью взаимодействия. Частицы с большим радиусом можно интерпретировать как людей с высокой подвижностью (молодежь и другие), частицы с малым радиусом - малоподвижные люди (представители старшего поколения и другие).
Отдельно стоит отметить, что в приложении существует возможность оценить активность людей в разных городах и странах в период эпидемии на основе данных компании Apple. даные содержат информацию об активности пользователей как на машинах, так и передвигающихся пешком. Соответствующие графики можно посмотреть в приложении.
В приложении также возможно посмотреть анимированую демонстрацию распространения вируса на мировой карте.

2.2. Анализ некоторых результатов.

Проанализируем полученные результаты последних двух моделей (рис. 4). Как было сказано выше, 4a приближенно описывает реальную динамику распространения вируса в мире. Таким образом, данную модель можно приближенно считать описывающей реальную динамику распространения. Видно, что после прохождения пика заражения, следует небольшое плато (лучше это видно на рис. 1), на котором количество заражений в единицу времени практически не меняется. Затем количество выздоровевших начинает превышать количество заболевших, смертность перестает расти и ситуация в целом стабилизируется. Важно заметить, что в модели не учитывалось наличие больниц и системы здравоохранения, а учитывался лишь иммунитет человека, который способен преодолеть болезнь. Таким образом, ситуация стабилизируется независимо от наличия системы здравоохранения. Последняя может влиять лишь на высоту этого пика и на длину пологого участка.
Конечно, даные выводы не могут считаться абсолютно верными, так как все оценки были сделаны довольно грубо и ошибка, которая может присутствовать в вычислениях, никак не оценена. Но, тем не менее, такие выводы могут иметь место и быть использованы для оценки общей ситуации.
На рисунке 4b показано распространение вируса в модели с карантинных мер. На демонстрации наглядно видно, что в этом случае скорость распространения вируса заметно снижается (по сравнению с 4a). Самоизоляция и карантин, таким образом, снижают вероятность быст-

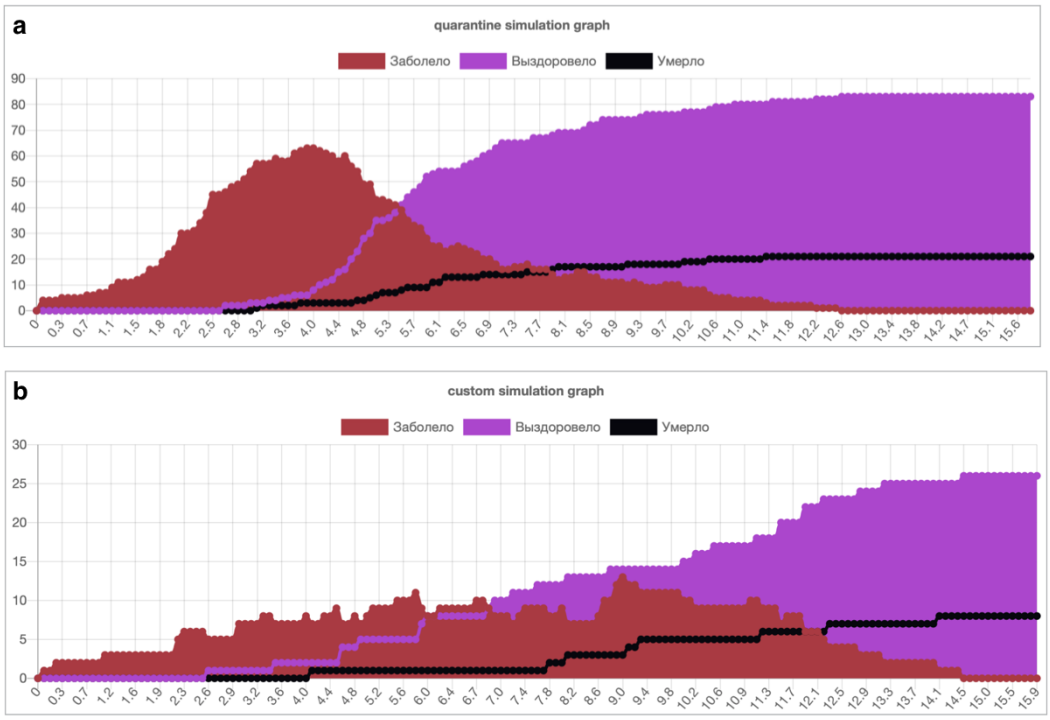

Рис. 4: Сравнение моделей с учетом карантинных мер и без них. На рисунке **a** показано распространение вируса в модели без введенных мер карантина. На рисунке **b** показано распространение вируса в модели с учетом карантинных мер .

рого роста заражения вирусом и позволяют системе здравоохранения принять на себя возросшую нагрузку и оказать качественную помощь пострадавшим, тем самым еще продлив рост кривой заражения. Сравнивая графики на a и b на рисунке 4 это становиться очевидно. Более того, видно, что во втором случае количество зараженных на порядки меньше, чем количество зараженных в первом случае.

3. Архитектура программы.

Перейдем к описанию архитектуры самой программы. Как уже отмечалось выше, приложение создано согласно патерну **MVC**. Опишем вкратце реализацию каждой части.

3.1. Реализация back-end.

Серверная часть реализована на языка python с использованием Django. Проект состоит из одного Django-приложения **simulations**, в котором реализованы основные элементы, обслуживающие сайт. Нет смысла подробно описывать устройство серверной части, так как ничего особенного и выдающегося она из себя не представляет. Подробное описание Django-приложений можно найти в документации [7].
Вкратце перечислим ключевые элементы. Класс models.Information() обеспечивает сохранение данных в базе данных. В ней содержатся все подписи к моделям, которые отображаются на клиентской части. Это сделано для того, чтобы можно было возможно редактировать данные, пользуясь панелью администратора сайта. В файле views.py реализованы функции,

выступающие в роли контролеров, связующих серверную часть и клиента. Некоторые из-них работают в режиме rest-api.

3.2. Реализация front-end.

Часть программы, которая представляет основной интерес, реализована на клиентской части. Главная страница обслуживается тремя фалами: **basic.js** отвечающает за построение моделей и отрисовку графиков, **graph.js** отвечает за подгрузку данных с серве для графика, показывающего активность пользователей во время пандемии, **control.js**, - за дополнительную анимацию на странице. Рассмотрим каждый файл детальнее:

- Файл **basic.js**, как уже отмечалось выше, выполняет две ключевые функции. Рассмотрим их по отдельности:

  1. Обслуживание моделей. Все модели являются наследниками одного класса **Particle**, который содержит общие для всех моделей свойства:

```
class Particle {
    constructor(x, y, velocity_x, velocity_y, radius, color, c , frame
       ){...}
    draw()){...}
    _checkSick(particle)){...}
    move(){}
    update(particles)){...}
}
```

  Метод **draw** рисует частицы на экране, **move** - обеспечивает движение частицы, а метод **update** обновляет положение частицы на экране (включает в себя два предыдущих метода). От класса **Particle** в свою очередь отнаследованы другие классы, каждый из которых добавляет свои свойства и/или переопределяет методы родительского класса:

```
class RecoveredParticle extends Particle{...}

class LifeParticle extends RecoveredParticle{...}

class QuarantineParticle extends LifeParticle{...}
```

  Эти классны являются экземплярами частиц с разными свойствами. Например класс **RecoveredParticle** поведение частицы (людей), которые со временем должны выздороветь. Класс **LifeParticle** описывает частицы (людей),в которых учтена вероятность смерти от вируса, а класс **QuarantineParticle** - частицы, которые можно посадить на карантин.
  Класс Simulations выполняет роль класса-контейнера для частиц. Именно он обеспечивает работу моделей. Для того чтобы создать новую модель, необходимо создать экземпляр класса **Simulations** и положить в него нужные частицы, определить вспомоглаетльные параметры, такие как: размер обасти, в котором будет происходить симуляция, количество частиц, их размеры и другие параметры.

```
class Simulations{
    constructor(canvas_id, Object, frame, amount, custom=false){...}
    draw(){...}
    linerInit(){...}
    squareDefaultInit(){...}
    objectsInit(r1, r2, v1, v2, amount, animating_time, sicked=2,
       quarantines=0){...}
    _get_nums(r1, r2, v1, v2, p=0){...}
    _check_x_y(x, y, radius){...}
    init(){...}
    update(){...}
    checkTime(){...}
}
```

Ключевыми методами тут являются **draw(), update()**, именно в них реализовано движение частиц. Отрисовка всех элементов происходит по средствам библиотеки **canvas**. Помимо этого метод **update()** отрисовывает графики, иллюстрирующие динамику распространения вируса в конкретной модели. Названия остальных методов говорит само за себя.

2. Отрисовка графиков. Как отмечалось выше, отрисовка графиков происходит в методах **update()**:

```
update(){
    ...
    config.data.labels.push(time.toFixed(1));
    config.data.datasets[0].data.push(sic);
    config.data.datasets[1].data.push(rec);
    config.data.datasets[2].data.push(die);

    window.graphs[this.elements.graph_id].update();
    ...
}
```

Глобальная переменная **config** помимо всего прочего еще содержит несколько массивов данных : количество заболевших людей , количество выздоровевших и количество умерших людей в единицу времени. После того, как необходимые данные переданы в переменную, она передаться в экземпляр класса **Chart** (из библиотеки Chart.js), который уже рисует полученные данные на графике.

- Файл **graph.js** отвечает за построение графика активности пользователей. Как только страница загружаться, по происходит ajax-запрос в внутреннему api. Результатом это запросы являются полученные данные для активности москвичей, которые успешно отрисовываються на графике по средствам библиотеки AMCharts.js. **graph.js** также отвечает за поиск данных, которые вбил пользователь, на сервере. Алгоритм устроен так, пользователь не может отравить запрос с не валидными данными.

- Файл **control.js** носит больше вспомогательные функции. Он обеспечивает анимированные переходы между элементами сайта.

Отдельного внимания заслуживает модель, параметры которой может задавать сам пользователь. Такая возможность обеспечивается средствам библиотеки DatGui [4].
Все переходы происходят по средствам библиотеки JQuery [5]. Для наглядности страницы были использован boostrap4 [6].
Помимо всего перечисленного в приложении присутствует интерактивная иллюстрация истории распространения вируса на мировой карте [2]. Данные, использованные в модели, взяты из тех-же отрытых источников, о которых шла речь в самом начале [1], [9].

4. Основные результаты.

Основным результатом моей работы является работающее приложение [11], с описанным выше функционалом. Исходный код приложения можно найти здесь [3].

Заключение

Проблема быстрого распространения вируса COVID-19 по нашей планете сейчас является одной из глобальных угроз для человечества. На борьбу с эпидемией брошены огромные силы и ресурсы. Значительная часть ученых сейчас занимается этой проблемой и, будем надеяться, в скором времени решит ее.

Тем не менее, уже сейчас каждому очевидно, что наш мир станет чуть другим после победы над вирусом. Последствия этой пандемии еще долго будут ощутимы. Именно поэтому, на мой взгляд, очень важно понимать и представлять механизм распространения вируса, основные особенности этого процесса и, что главное, последствия, который неизбежно влечет за собой каждый вирус.

Данное web-приложение помогает детальнее понять механизм распространения вируса, а также его последующую динамику развития.

Список литературы

[1] The covid tracking project. https://covidtracking.com/about/software/, 2020.

[2] Coronavirus animated dashboard. https://www.amcharts.com/demos/corona-virus-animated-dashboard/.

[3] GitHub code of project. https://github.com/andrew3203/CovidSimulation/.

[4] DatGui. https://github.com/dataarts/dat.gui/.

[5] DatGui. https://jquery.com/.

[6] DatGui. https://getbootstrap.com/docs/4.0/getting-started/introduction//.

[7] Django documentation. https://docs.djangoproject.com/en/3.0/.

[8] Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU). Covid-19 dashboard. https://covidtracking.com/about/software/, 2020.

[9] Johns Hopkins. Coronavirus resource center. https://coronavirus.jhu.edu/, 2020.

[10] Rosbotrebnanzor. https://www.rospotrebnadzor.ru/about/info/news_time/news_details.php?ELEMENT_ID=13566/, 2020.

[11] Covid-19 Simulations. https://covid-simulation.herokuapp.com/.

[12] Yandex. https://yandex.ru/covid19/, 2020.