

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

УТВЕРЖДАЮ

Зав.кафедрой,

к. ф.-м. н.

_____ С. В. Миронов

ОТЧЕТ О ПРАКТИКЕ

студента 4 курса 411 группы факультета КНиИТ

Власова Андрея Александровича

вид практики: учебная

кафедра: математической кибернетики и компьютерных наук

курс: 4

семестр: 2

продолжительность: 2 нед., с 01.07.2020 г. по 14.07.2020 г.

Руководитель практики от университета,

к. ф.-м. н.

С. В. Миронов

Руководитель практики от организации (учреждения, предприятия),

к. ф.-м. н.

С. В. Миронов

Тема практики: «Создание приложения для анализа генетического алгоритма поиска центральных вершин»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Использование приложения	5
2 Описание технологий и архитектуры приложения	5
2.1 Структура базы данных	5
2.2 Уровень доступа к данным.....	7
2.3 Уровень бизнес-логики	8
2.4 Уровень представления	9
Приложение А Нумеруемые объекты в приложении.....	10

ВВЕДЕНИЕ

adasdasd

1 Использование приложения

2 Описание технологий и архитектуры приложения

В качестве языка программирования для решения поставленной задачи был выбран объектноориентированный язык C#. Вместе с тем для создания клиентсерверного приложения был использован фреймворк ASP.NET MVC 5, который позволяет создавать веб-приложения с использованием архитектуры MVC. Кроме этого в качестве системы объектнореляционного отображения используется технология Entity Framework 6. При этом приложение разделено на три слоя абстракции — уровень доступа к данным, уровень бизнеслогики и уровень визуального представления.

2.1 Структура базы данных

Хранение графов в базе данных были созданы следующие таблицы: Graphs и Edges. При этом таблица Graphs содержит следующие поля:

- поле Id (типа данных INT) — уникальный идентификатор, внутренний ключ,
- поле N (типа данных INT) — количество вершин в графе,
- поле M (типа данных INT) — количество ребер в графе,
- поле Name (типа данных NVARCHAR) — название графа,
- поле R (типа данных INT) — радиус графа.

Кроме этого таблица Edges состоит из следующих полей:

- поле Id (типа данных INT) — уникальный идентификатор, внутренний ключ,
- поле V1 (типа данных INT) — одна из вершин, которые соединяет ребро,
- поле V2 (типа данных INT) — вторая из вершин, которые соединяет ребро,
- поле Graph_Id (типа данных INT) — Id графа, которому принадлежит ребро, внешний ключ.

Кроме этого для хранения зарегистрированных пользователей существует таблица Users:

- поле Id (типа данных INT) — уникальный идентификатор, внутренний ключ,
- поле Login (типа данных NVARCHAR) — логин пользователя,
- поле Password (типа данных NVARCHAR) — зашифрованный пароль пользователя.

Полная диаграмма таблиц представлена на рисунке 1:

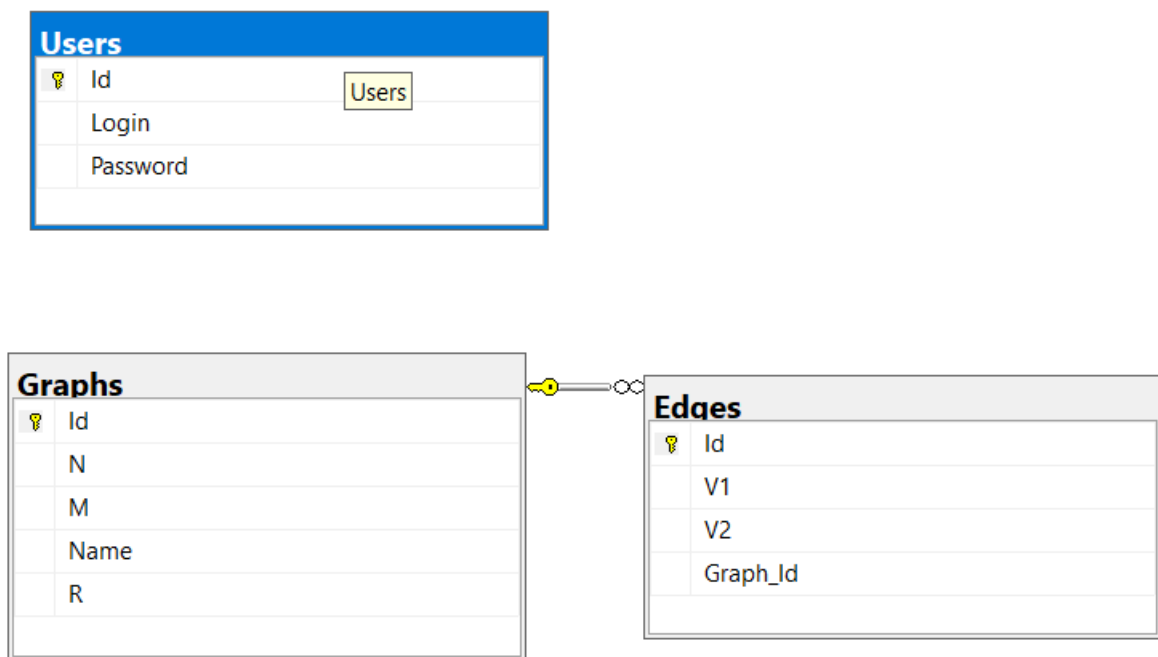


Рисунок 1 – Диаграмма базы данных

Для создания базы данных используется технология Entity Framework, вместе с чем использовался подход Code-first, согласно которому были созданы классы Graph, GraphInfo, Edge, User, описывающие модели данных:

```

1 public class GraphInfo
2 {
3     public int Id { get; set; }
4     public int N { get; set; }
5     public int M { get; set; }
6     public string Name { get; set; }
7     public int R { get; set; }
8 }

1 public class Graph : GraphInfo
2 {
3     public ICollection<Edge> Edges { get; set; }
4     public Graph()
5     {
6         Edges = new List<Edge>();
7     }
8 }
  
```

```

1 public class Edge
2 {
3     public int Id { get; set; }
4     public int V1 { get; set; }
5     public int V2 { get; set; }
6 }

```

2.2 Уровень доступа к данным

Для гибкой и стандартизированной работы с базой данных был создан ряд интерфейсов, в которые были вынесены основные методы для доступа к данным и их изменениям. Классы, реализующие эти интерфейсы представляют собой уровень доступа к данным, при этом использование интерфейсов позволяет с легкостью изменять реализацию этих классов, а также упрощает процесс тестирования.

Далее приводится код основных интерфейсов, которые используются при работе с данными:

```

1 public interface IGraphDao
2 {
3     IEnumerable<GraphInfo> GetAllGraphInfo();
4     Graph GetById(int id);
5     Graph Add(Graph graph);
6 }

1 public interface IUserDao
2 {
3     User GetById(int id);
4     User GetByName(string name);
5     User Add(User user);
6 }

```

Вместе с тем для использования технологии Entity Framework созданы классы `GraphContext` и `UserContext`, которые наследуются от класса `System.Data.Entity.DbContext`, что позволяет получить возможность для легкого доступа к базе данных без написания SQL запросов. Классы `GraphContext` и `UserContext` содержат в себе поля типа `DbSet<Graph>` и `DbSet<User>`, через которые происходит добавление, чтение или изменение данных в базе данных. Кроме этого в этих классах описан статический конструктор, внутри которого указан способ начальной инициализации базы данных, за счет

классов `UserContextInitializer` и `GraphContextInitializer`. Эти классы наследуются от класса `CreateDatabaseIfNotExists`, что позволяет фреймворку выполнить начальное заполнение данными, если база данных еще не существует, за счет кода, который описан в переопределенном методе `Seed`. Внутри этого метода происходит чтение нескольких созданных графов из текстовых файлов, все это происходит внутри метода, описанного в классе `GraphContextInitializer`, в этом же методе в классе `UserContextInitializer` добавляется пользователь с логином `admin` и паролем `admin`. Полный код представлен в приложении [ссылка].

2.3 Уровень бизнес-логики

Уровень бизнес-логики представляет собой похожую структуру, как и уровень доступа данных — так же созданы ряд интерфейсов и классы, которые их реализуют. При этом многие из этих интерфейсов похожи на те, которые описаны в уровне доступа к данным, однако именно на этом уровне происходит запуск генетического алгоритма с различными параметрами и анализ загруженных графов. С определением интерфейсов и классов реализующих бизнес-логику приложения можно ознакомиться в приложении [ссылка]. Классы `GraphBL` и `UserBL`, реализуют интерфейсы `IGraphBL` и `IUserBL`. Они содержат ссылки на объекты, реализующие интерфейсы `IGraphDao` и `IUserBL`, при этом эти объекты передаются в качестве параметров в соответствующие конструкторы. При добавлении нового пользователя происходит проверка на существование записи с таким же логином, а кроме этого происходит шифровка пароля.

Также при добавлении нового графа в базу данных в классе, который отвечает за работу с моделью графа, происходит проверка графа на связность и его размеры. При неудачном прохождении проверки выбрасывается исключение, которое отлавливается на уровне представления.

Кроме этого для работы с генетическим алгоритмом создан интерфейс `IAlgorithm`:

```
1 public interface IAlgorithm
2     {
3         FindingVertexResponse FindCentralVertex(Graph graph);
4         ResearchAlgorithmResponse ResearchAlgorithm(ResearchRequest param);
5     }
```


Интерфейс определяет набор методов, в которых будет реализована логика для работы с генетическим алгоритмом. Вместе с тем класс `Algorithm` реализует данный интерфейс и в нем содержится вся логика работы с алгоритмом — получение результатов поиска центральных вершин, замеры времени работы и процента неверно найденных решений.

Результаты измерений возвращаются из методов при помощи классов `FindingVertexResponse` и `ResearchAlgorithmResponse`:

```
1 public class FindingVertexResponse
2 {
3     public int[] Center { get; set; }
4     public int R { get; set; }
5     public double Time { get; set; }
6 }

1 public class ResearchAlgorithmResponse
2 {
3     public double AvgTime { get; set; }
4     public double Error { get; set; }
5 }
```

2.4 Уровень представления

Так как проект представляет собой веб-приложение, то уровень, отвечающий за пользовательский интерфейс реализован при помощи технологии ASP .NET MVC 5. В связи с чем весь код этого уровня разделен на:

- контроллеры, которые отвечают на HTTP запросы клиента с помощью представлений,
- представления, которые написаны с использованием технологии Razor, позволяющей внедрять серверный C# код,
- модели данных, внутри которых происходит передача данных от клиента серверу и обратно.

ПРИЛОЖЕНИЕ А

Нумеруемые объекты в приложении

HELLO