 **Универзитет „Св. Климент Охридски“ - Битола **

**ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ И КОМУНИКАЦИСКИ ТЕХНОЛОГИИ**

ПРОЕКТНА ЗАДАЧА

ПО ПРЕДМЕТОТ **БЕЗЖИЧНИ И АД-ХОК КОМПЈУТЕРСКИ МРЕЖИ**

**TRAFFIC MANAGEMENT**

|  |  |
| --- | --- |
| Предметен наставник:  Доц д-р Томе Димовски | Изработиле:  Андреа Стаменковска бр. инд. КНИ307,  Ленче Танаскоска бр. инд. КНИ310,  Стефан Стојановски бр. инд. КНИ407 |

Битола, декември, 2017

[СОДРЖИНА 2](#_Toc498367506)

[АПСТРАКТ 3](#_Toc498367507)

[ВОВЕД 4](#_Toc498367508)

[ГЛАВА 1 – БАЗА 5](#_Toc498367509)

[1.1 Табели во базата 5](#_Toc498367510)

[1.2 Релации во базата 6](#_Toc498367511)

[1.3 Процедури во базата 6](#_Toc498367512)

[ГЛАВА 2. WEB API - TRAFFICMANAGEMENTAPI 7](#_Toc498367513)

[2.1 Модели - namespace TrafficManagementApi.Models 7](#_Toc498367514)

[2.1.1 Base response 7](#_Toc498367515)

[2.1.2 Crossroad 7](#_Toc498367516)

[2.1.3 CrossroadParameters 7](#_Toc498367517)

[2.1.4 Pollution 8](#_Toc498367518)

[2.1.5 ResultCrossroad 8](#_Toc498367519)

[2.1.6 ResultRoute 8](#_Toc498367520)

[2.1.7 Route 8](#_Toc498367521)

[2.1.8 RouteCrossroad 8](#_Toc498367522)

[2.1.9 TrafficCongestion 8](#_Toc498367523)

[2.2 Контролери - namespace TrafficManagementApi.Controllers 9](#_Toc498367524)

[2.2.1 BaseController 9](#_Toc498367525)

[2.2.2 CalculateController 9](#_Toc498367526)

[2.2.3 CrossroadController 9](#_Toc498367527)

[2.2.4 CrossroadParametersController 9](#_Toc498367528)

[2.2.5 CrossroadPriorityController 9](#_Toc498367529)

[2.2.6 ResultCrossroadController 10](#_Toc498367530)

[2.2.7 ResultRouteController 10](#_Toc498367531)

[2.7.8 RouteController 10](#_Toc498367532)

[2.7.9 RouteCrossroadsController 10](#_Toc498367533)

[ГЛАВА 3 TRAFFICMANAGEMENT WEB PROJECT 10](#_Toc498367534)

[3.1 AnglularJS контролери 10](#_Toc498367535)

[3.1.1 ctrlTrafficManagement 10](#_Toc498367536)

[3.2 HTML документи 10](#_Toc498367537)

[ЗАКЛУЧОК 11](#_Toc498367538)

**АПСТРАКТ**

Traffic Management претставува софтвер кој ја пресметува најдобрата рута за стигање до одредена цел при претходно одредени почетна и крајна точка. Изработен е како веб апликација. Начинот на пресметка е врз основа на податоци кои се читаат од база. Главен фактор при пресметката се податоците за загадувањето и густината на сообраќајот на крстосниците низ кои поминува рутата. Базирано на овие податоци за секоја рута која ги поврзува почетната и крајната точка се пресметува приоритет и потоа се избира рутата со најмал прироритет како најповолна. Користена е релациона база (MS SQL) до која од веб страната се пристапува со користење на Web API. За изработка на Web API-то е користен C# програмскиот јазик. Самата веб страна е креирана со користење на HTML5, CSS3, Bootstrap, а до апито се пристапува со користење на AngularJS.

**ВОВЕД**

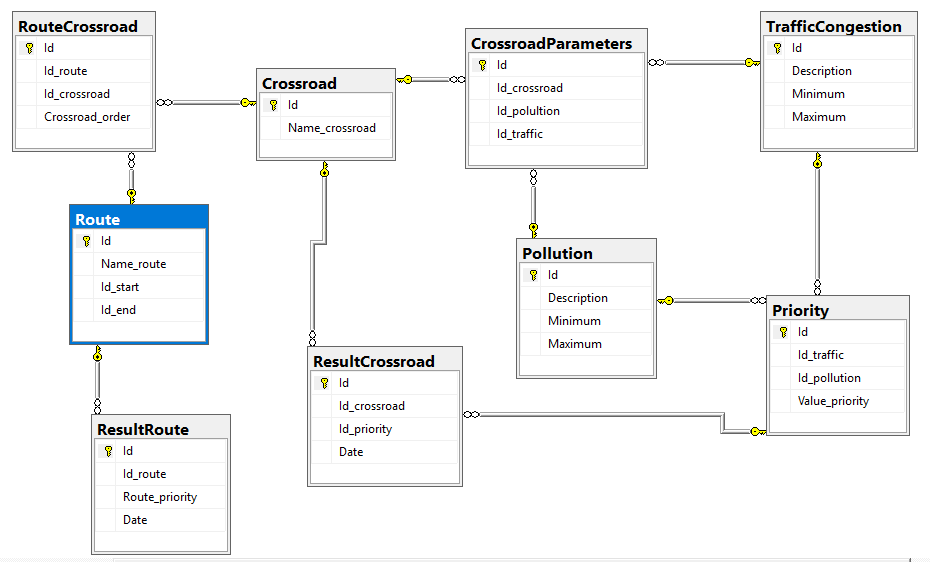
Како резултат на брзото темпо на живот, се почесто се наоѓаме во ситуација да сме заглавени во сообраќајот. Поради тоа дојдовме до идеја како тоа да го спречиме или пак да го превенираме. Идејата е да има апликација која ни овозможува да ја избереме почетната и крајната дестинација на нашето патување и врз основа на параметри како што се густина на сообраќајот и индекс на загадување да ни ја каже најдобрата рута. Истата е имплементирана како веб апликација која се состои од неколку слоеви: релациона база на податоци од која читаме и запишуваме податоци која е разработена во Глава 1, веб апи преку кое правиме повици до базата кое е разработено во Глава 2 и веб кориснички интерфејс на кој со користење на AngularJS фрејмворкот имаме можност да ја избереме почетната и крајната дестинација и ни ја прикажува најдобрата рута кој е разработен во Глава 3. Кога ќе ја избереме почетната и крајната дестинација, се прави повик до веб апито каде врз основа на изберените дестинации како и параметрите на загадување и густина на сообраќајот пресметува која е најдобрата рута , го запишува резултатот во база и го враќа резултатот назад до веб корисничкиот интерфејс кажувајќи ни која е најдобрата рута.

**ГЛАВА 1 – БАЗА**

**1.1 Табели во базата**

* Crossroad – Се чуваат податоци за раскрсниците – името на расрсниците
  + Id float PK
  + Name\_crossroad nvarchar(50)
* RouteCrossroad – ги поврзува табелите Crossroad и Route и дополнително се чува податок за која по ред е раскрсницата во дадената рута
  + Id int PK
  + Id\_route int FK
  + Id\_crossroad float FK
  + Crossroad\_order nvarchar(10)
* CrossroadParameters – ги поврзува табелите crossroad, Pollution и Traffic. Се користи за селектирање на податоци за загадување и сообраќај за дадена раскрница.
  + Id int PK
  + Id\_crossroad float FK
  + Id\_polultion nvarchar(2) FK
  + Id\_traffic nvarchar(2) FK
* Pollution – чува податоци за загадувањето, минимален и максимален праг за даденото загадување
  + Id nvarchar(2) PK
  + Description nvarchar(50)
  + Minimum float
  + Maximum float
* Priority – чува податоци за вредноста на приоритетот според загадувањето и густината на собраќај
  + Id int PK
  + Id\_traffic nvarchar(2) FK
  + Id\_pollution nvarchar(2) FK
  + Value\_priority int
* ResultCrossroad – се запишуваат податоци за приоритетот на крстосницата во дадено време.
  + Id int PK
  + Id\_crossroad float FK
  + Id\_priority int FK
  + Date datetime
* ResultRoute - се запишуваат податоци за приоритетот на рутата во дадено време. Приоритетот претсавува збир од приоритетите на крстосниците.
  + Id int PK
  + Id\_route int FK
  + Route\_priority int
  + Date datetime
* Route – чува податоци за рутата со почетна и крајна раскрница
  + Id int PK
  + Name\_route nvarchar(50)
  + Id\_start float
  + Id\_end float
* TrafficCongestion - чува податоци за густината на сообраќајот, минимален и максимален праг за дадената густина на сообраќајот
  + Id nvarchar(2) PK
  + Description nvarchar(50)
  + Minimum float
  + Maximum float

**1.2 Релации во базата**



Слика 1.1 Дијаграм на базата

**1.3.Процедури во база**

* USP\_Crossroad\_SelectAll
* USP\_CrossroadParameters\_Select (@Id\_crossroad float = 0.0)
* USP\_CrossroadPriority\_Select (@Id\_traffic NVARCHAR(2), @Id\_pollution NVARCHAR(2))
* USP\_ResultCrossroad\_Insert(@Id\_crossroad float, @Id\_priority int, @Date datetime)
* USP\_ResultRoute\_Insert (@Id\_route int, @Route\_priority int, @Date datetime)
* USP\_Route\_Select (@Id\_start float =1.1, @Id\_end float = 1.1)
* USP\_Route\_SelectAll
* USP\_Route\_SelectAllEnds
* USP\_Route\_SelectAllStarts
* USP\_Route\_SelectById(@Id\_route int =0)

**ГЛАВА 2. WEB API - TRAFFICMANAGEMENTAPI**

**2.1 Модели - namespace TrafficManagementApi.Models**

2.1.1 Base response

/// <summary>

/// Base response class

/// </summary>

public class BaseResponse

{

/// <summary>

/// Base message returned to client

/// </summary>

public string Message { get; set; }

/// <summary>

/// Response status

/// </summary>

public ResponseStatus Status { get; set; }

}

public enum ResponseStatus

{

Error = 0,

Success = 1

}

}

2.1.2 Crossroad

public class Crossroad: BaseResponse

{

public decimal Id { get; set; }

public String Name { get; set; }

}

2.1.3 CrossroadParameters

public class CrossroadParameters: BaseResponse

{

public int Id { get; set; }

public decimal Id\_crossroad { get; set; }

public String Id\_pollution { get; set; }

public String Id\_traffic { get; set; }

}

2.1.4 Pollution

public class Pollution: BaseResponse

{

public String Id { get; set; }

public String Description { get; set; }

public decimal Minimum { get; set; }

public decimal Maximum { get; set; }

}

2.1.5 ResultCrossroad

public class ResultCrossroad: BaseResponse

{

public int Id { get; set; }

public decimal Id\_Crossroad { get; set; }

public int Id\_Priority { get; set; }

public DateTime Date { get; set; }

}

2.1.6 ResultRoute

public class ResultRoute: BaseResponse

{

public int Id { get; set; }

public int Id\_Route { get; set; }

public int Route\_Priority { get; set; }

public DateTime Date { get; set; }

}

2.1.7 Route

public class Route: BaseResponse

{

public int Id { get; set; }

public String Name{ get; set; }

public decimal Id\_Start { get; set; }

public decimal Id\_End { get; set; }

}

2.1.8 RouteCrossroad

public class RouteCrossroad: BaseResponse

{

public int Id { get; set; }

public int Id\_Route { get; set; }

public decimal Id\_Crossroad { get; set; }

public int Crossroad\_Order { get; set; }

}

2.1.9 TrafficCongestion

public class TrafficCongestion

{

public String Id { get; set; }

public String Description { get; set; }

public decimal Minimum { get; set; }

public decimal Maximum { get; set; }

}

**2.2 Контролери - namespace TrafficManagementApi.Controllers**

2.2.1 BaseController

Методи:

* public string ConnectionStringName()

2.2.2 CalculateController

Методи:

* [HttpGet] public Route calculateRoute(decimal idStart, decimal idEnd) – се повикува на клик на копчето ПРЕСМЕТАЈ при што според избраните почетна и крајна точка се селектираат рутите од табелата Route кои ги имаат соодветните почетни и крајни раскрсници. Потоа според Id-то на секоја рута се селектираат сите крстосници од Crossroads табелата кои припаѓаат на дадената рута. За секоја крстосница од табелата CrossroadParameter се селектираат параметрите според Id-то на крстосницата. Со добиените параметри се врши селекција на приоритетот од табелата Priority при што се добива приоритетот за самата крстосница и истиот се запишува во табелата ResultCrossroad и се чува во листа за понатамошна употреба. Откако ќе се пресметаат приоритетите на сите крстосници се собираат нивните приоритети и се добива приоритетот на рутата кој се запишува во табелата ResultRoute и тој резултат се чува во друга листа. Истото се повторува за сите рути. На крај се врши спореба на податоците од листата во која се зачувани приоритетите за сите крстосници и се избира крстосницата со најмал приоритет. Се врши селектирање на соодветната рута од табелата Route и селектираните податоци се прикажуваат.

2.2.3 CrossroadController

Методи

* [HttpGet] public List<Crossroad> GetData()

2.2.4 CrossroadParametersController

Методи

* [HttpGet] public CrossroadParameters GetData(decimal idCrossroad)

2.2.5 CrossroadPriorityController

Методи:

* [HttpGet] public Priority GetData(String idTraffic, String idPollution)

2.2.6 ResultCrossroadController

Методи

* [HttpPost] public ResultCrossroad AddResult (ResultCrossroad response)

2.2.7 ResultRouteController

Методи

* [HttpPost] public ResultRoute AddResult (ResultRoute response)

2.7.8 RouteController

Методи

* [HttpGet] public List<Route> GetRoutes()
* [HttpGet] public List<Route> GetRoute(Route insert)
* [HttpGet] public Route GetRouteById(int id)
* [HttpGet] public List<Route> GetEnds()
* [HttpGet] public List<Route> GetStarts()

2.7.9 RouteCrossroadsController

Методи

* [HttpGet] public List<RouteCrossroad> GetData(Route route)

**ГЛАВА 3 TRAFFICMANAGEMENT WEB PROJECT**

**3.1 AnglularJS контролери**

3.1.1 ctrlTrafficManagement

Методи

* function getRouteStarts()
* function getRouteEnds()
* function calculateRoute(idStart, idEnd)

2.3.2 HTML документи

* index.html

**ЗАКЛУЧОК**

Како резултат на брзото темпо на живот, се почесто се наоѓаме во ситуација да сме заглавени во сообраќајот. Поради тоа дојдовме до идеја како тоа да го спречиме или пак да го превенираме. Идејата е да има апликација која ни овозможува да ја избереме почетната и крајната дестинација на нашето патување и врз основа на параметри како што се густина на сообраќајот и индекс на загадување да ни ја каже најдобрата рута. Истата е имплементирана како веб апликација која се состои од неколку слоеви: релациона база на податоци од која читаме и запишуваме податоци, веб апи преку кое правиме повици до базата и веб кориснички интерфејс на кој со користење на AngularJS фрејмворкот имаме можност да ја избереме почетната и крајната дестинација и ни ја прикажува најдобрата рута. Кога ќе ја избереме почетната и крајната дестинација, се прави повик до веб апито каде врз основа на изберените дестинации како и параметрите на загадување и густина на сообраќајот пресметува која е најдобрата рута , го запишува резултатот во база и го враќа резултатот назад до веб корисничкиот интерфејс кажувајќи ни која е најдобрата рута.