МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 05 01 Информационные системы и технологии

Специализация

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Применение технологии полнотекстового поиска для базы данных приложениясистемы управления проектами»

Выполнил студент Юзефчик Д.П.

(Ф.И.О.)

Руководитель проекта Жигаровская С.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: Жигаровская С.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: Жигаровская С.С.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2019

Содержание

[**Введение** 3](#_Toc8655649)

[**1.** **Постановка задачи** 4](#_Toc8655650)

[**2.** **Теоретическое введение** 5](#_Toc8655651)

[**2.1.** **MS SQL Server** 5](#_Toc8655652)

[**2.2.** **Технология «Full Text Search»** 5](#_Toc8655653)

[**3.** **Проектирование** 6](#_Toc8655654)

[**3.1.** **Проектирование модели базы данных** 6](#_Toc8655655)

[**3.2.** **Применение технологии «Full Text Search»** 8](#_Toc8655656)

[**3.3.** **Описание генерации данных, импорта и экспорта** 9](#_Toc8655657)

[**4.** **Тестирование** 11](#_Toc8655658)

[**5.** **Руководство пользователя** 12](#_Toc8655659)

[**Заключение** 13](#_Toc8655660)

[**Список используемых источников** 14](#_Toc8655661)

[**Приложение** 15](#_Toc8655662)

# **Введение**

Программное обеспечение для управления проектами — комплексное программное обеспечение, включающее в себя приложения для планирования задач, составления расписания, распределения ресурсов, совместной работы, общения, быстрого управления, документирования и администрирования системы, которое используются совместно для управления проектами.

В моем случае эта система будет реализовывать лишь часть этих функций в связи с ложностью ее реализации, но в целом, такая система должна позволять проводить, в первую очередь, именно планирование, которое будет реализовано в виде досок задачами.

1. **Постановка задачи**

В ходе выполнения курсового проекта должны быть реализованы следующие задачи:

* Написание API для взаимодействия с БД;
* Использование технологии Full-Text Search;
* Возможность создания рабочих проектов и их изменение;
* Возможность создания и редактирования отдельных задач;
* Возможность создания и редактирования команд разработчиков;
* Разделение прав доступа к общим ресурсам.

1. **Теоретическое введение**
   1. **MS SQL Server**

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

* 1. **Технология «Full Text Search»**

Методы полнотекстового поиска позволяют очень быстро находить нужную информацию в больших объемах текста.

В отличие от оператора LIKE, такой тип поиска предусматривает создание соответствующего полнотекстового индекса, который представляет собой своеобразный словарь упоминаний слов в полях. Под словом обычно понимается совокупность из не менее 3-х не пробельных символов (но это может быть изменено). В зависимости от данных словаря может быть вычислена релевантность – сравнительная мера соответствия запроса найденной информации.

Текстовый индекс FULLTEXT может быть наложен на поля CHAR, VARCHAR и TEXT. Причем, как и в случае с обычным индексом – если происходит поиск по 2-м полям, то нужен объединенный индекс 2-х полей, используйте поиск по одному полю – нужен индекс только этого поля.

1. **Проектирование**
   1. **Проектирование модели базы данных**

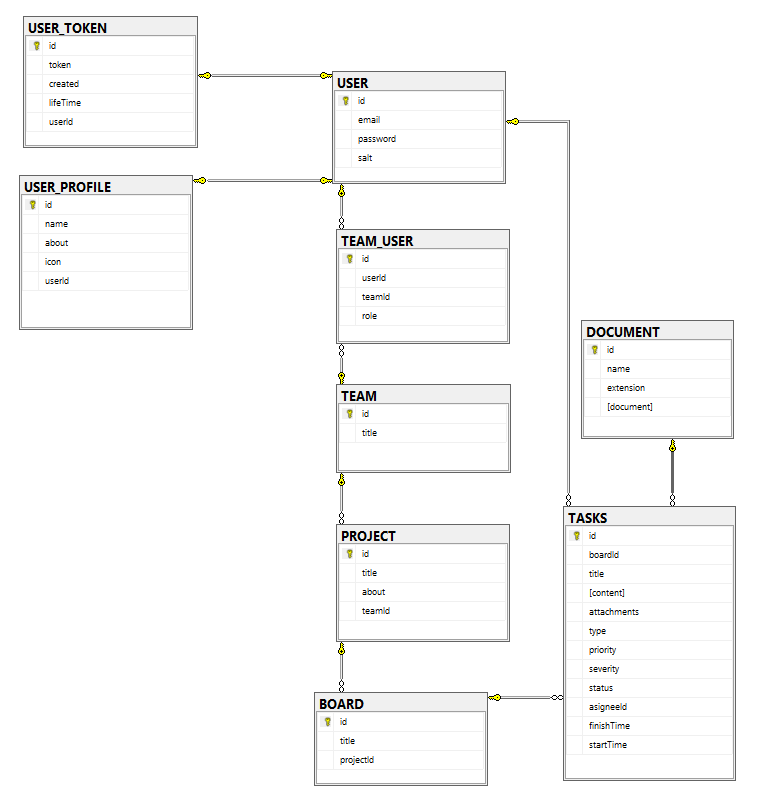


Рисунок 3.1 - схема базы данных

База данных состоит из следующих таблиц:

USER

Таблица содержит данные, необходимые для идентификации и авторизации пользователя.

* Id
* Email
* Пароль в виде хэша
* Salt – случайно число, которое добавляется к паролю для увеличения криптостойкости и избавления от появления одинаковых значений в таблице

USER\_PROFILE

Таблица содержит основные данные пользователя.

* Id
* Имя
* Краткую сводку пользователя
* Изображение для аккаунта
* UserId для связи один к одному с таблицей USER

USER\_TOKEN

Таблица содержит токены доступа к системе, которые генерируются при входе пользователя в систему, эти токены удаляются при помощи задач каждые 30 минут.

* Id
* Токен – случайное число
* Дата создания
* Время жизни
* UserId для связи один к одному с таблицей USER

TEAM

Таблица служит для разделения пользователей на команды.

* Id
* Название

TEAM\_USER

Таблица нужна для связи многие ко многим таблиц TEAM и USER и для разделения на роли пользователей.

* Id
* UserId
* TeamId
* Роль

PROJECT

Таблица нужна для создания проектов.

* Id
* Название
* Краткая сводка
* TeamId связь один ко многим

BOARD

Таблица содержит в себе список задач, это сделано с целью возможности перемещения задач из одной доски в другую, пример: задача после завершения должна быть протестирована, а позже закрыта, для этого нужно 3 доски: текущие задачи, на тестировании, завершенные.

* Id
* Название
* ProjectId связь один ко многим

TASKS

Содержит в себе перечень информации, необходимой для выполнения поставленной задачи.

* Id
* BoardId
* Название
* Основное описание
* Приложения в виде скриншотов или документов, связь с таблицей
* Тип задачи
* Приоритет
* Опасность
* Статус выполнения
* Назначенный для выполнения, связь с таблицей USER
* Дата окончания выполнения задачи

DOCUMENT

Таблица содержит в себе бинарные файлы

* Id
* Название документа
* Расширение
* Сам файл в бинарном виде
  1. **Применение технологии «Full Text Search»**

Применение заключается в поиске задач по одной из следующих колонок: название, описание и возможность поиска в файлах, которые были помещены во вложения. Это позволяет вкладывать большие объемы текста и делать выборку по ключевым задачам, к примеру, такие файлы могут содержать ключевые слова (такими функциями обладают многие актуальные системы проектами, но есть одна ситуация, при которой это может быть полезно, заказчик может внести правки и сам прикрепить такой документ с ключевым словом и открыть задачу заново).

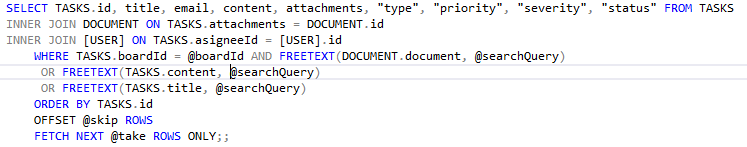


Рисунок 3.2 – основной код процедуры полнотекстового поиска

В данном на рисунке 3.2 функция FREETEXT() выполняет поиск по заранее созданному индексу в таблице DOCUMENT.

Для реализации данной задачи необходимо предварительно сохранить файл и создать полнотекстовые индексы, это изображено на рисунке 3.4.

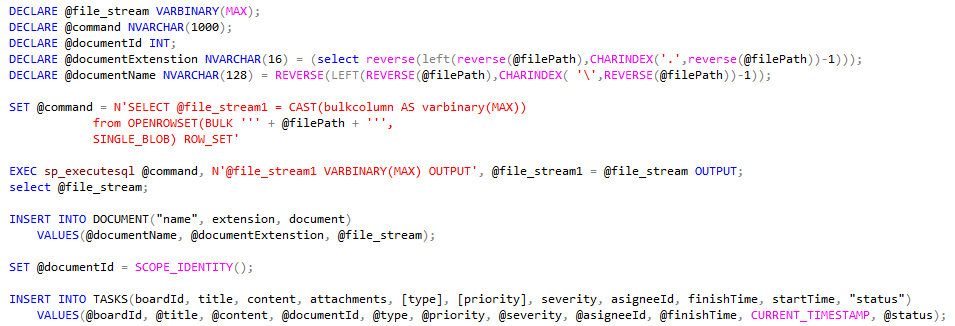


Рисунок 3.3 – основной код процедуры сохранения файла в бинарном виде

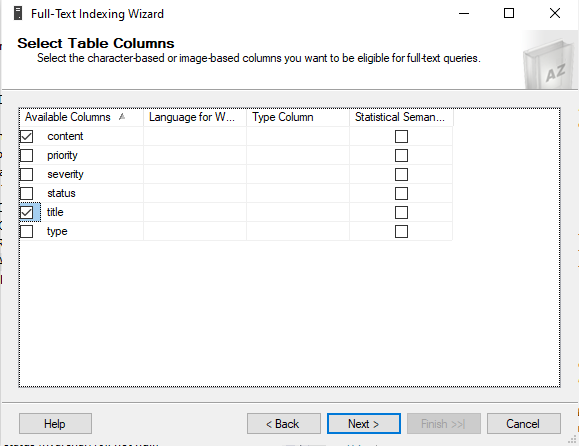


Рисунок 3.4 – добавление полнотекстового индекса в таблицу TASKS

* 1. **Описание генерации данных, импорта и экспорта**

Для наполнения базы данных мною был написан простой генератор, который выполняет хранимые процедуры, пример кода изображен на рисунке 3.5.





Рисунок 3.5 – генерация новых пользователей и команд

Для экспортирования данных в JSON файлы необходимо выполнить SELECT запрос, который изображен на рисунке 3.6.



Рисунок 3.6 – экспорт данных из таблицы

Для импорта необходимо выполнить запрос, который изображен на рисунке 3.7.



Рисунок 3.7 – импорт данных из файла

После этого данные можно вставить в таблицу.

1. **Тестирование**

Тестирование проводилось на стороне сервера при различном количестве объеме вставленных данных.

1. **Руководство пользователя**

# **Заключение**

В результате выполнения курсовой работы была спроектирована база данных системы управления проектами. В качестве СУБД использовался MS SQL Server, для реализации API использовалась технология WPF и язык программирования C#.

Также мною были изучены возможности технологии полнотекстового поиска, это положительно сказывается на быстродействие, если сравнивать с обычным сравнением срок и гораздо проще чем написание сложных регулярных выражений, также, это позволяет выполнять поиск в файлах, которые хранятся в базе данных.

**Список используемых источников**

1. Microsoft docs [электронный ресурс] - [https://docs.microsoft.com/en-us/sql](https://docs.microsoft.com/en-us/sql/t-sql/queries/freetext-transact-sql?view=sql-server-2017)
2. Sybase [электронный ресурс] - <http://infocenter.sybase.com>
3. Metanit [электронный ресурс] - <https://metanit.com/>
4. Wikipedia [электронный ресурс] - https://www.wikipedia.org/
5. MSDN [электронный ресурс] - <https://msdn.microsoft.com/en-us/>

**Приложение**

USE Taskboard;

CREATE TABLE "USER"

(

id INT IDENTITY(1, 1) PRIMARY KEY,

email NVARCHAR(256) UNIQUE NOT NULL,

"password" BINARY(64) NOT NULL,

salt UNIQUEIDENTIFIER UNIQUE NOT NULL

);

CREATE TABLE "USER\_TOKEN"

(

id INT IDENTITY(1, 1) PRIMARY KEY,

token UNIQUEIDENTIFIER UNIQUE NOT NULL,

created DATETIME NOT NULL,

"lifeTime" INT NOT NULL DEFAULT 30,

userId INT UNIQUE FOREIGN KEY REFERENCES "USER"(id)

);

CREATE TABLE USER\_PROFILE

(

id INT IDENTITY(1, 1) PRIMARY KEY,

"name" NVARCHAR(64) NOT NULL,

about NVARCHAR(256),

icon IMAGE,

userId INT UNIQUE FOREIGN KEY REFERENCES "USER"(id)

);

CREATE TABLE TEAM

(

id INT IDENTITY(1, 1) PRIMARY KEY,

title NVARCHAR(64) NOT NULL DEFAULT 'Team'

);

CREATE TABLE PROJECT

(

id INT IDENTITY(1, 1) PRIMARY KEY,

title NVARCHAR(64) NOT NULL DEFAULT 'Project',

about NVARCHAR(256),

teamId INT FOREIGN KEY REFERENCES TEAM(id)

);

CREATE TABLE BOARD

(

id INT IDENTITY(1, 1) PRIMARY KEY,

title NVARCHAR(64) NOT NULL DEFAULT 'Board',

projectId INT FOREIGN KEY REFERENCES PROJECT(id)

);

CREATE TABLE DOCUMENT

(

id INT IDENTITY(1, 1) PRIMARY KEY,

"name" NVARCHAR(128),

extension NVARCHAR(16),

document VARBINARY(MAX),

);

CREATE TABLE TASKS

(

id INT IDENTITY(1, 1) PRIMARY KEY,

boardId INT FOREIGN KEY REFERENCES BOARD(id),

title NVARCHAR(64) NOT NULL DEFAULT 'Task',

content NVARCHAR(1024),

attachments INT REFERENCES DOCUMENT(id),

"type" NVARCHAR(11) NOT NULL

CHECK ("type" IN ('BUG', 'FEATURE', 'IMPROVEMENT'))

DEFAULT 'BUG',

"priority" NVARCHAR(6) NOT NULL

CHECK ("priority" IN ('LOW', 'MEDIUM', 'HIGH', 'URGENT'))

DEFAULT 'LOW',

severity NVARCHAR(8) CHECK (severity IN ('BLOCKER', 'CRITICAL', 'NORMAL', 'LOW'))

DEFAULT 'LOW',

"status" NVARCHAR(16) NOT NULL

CHECK ("status" IN ('CREATED', 'IN PROGRESS', 'FINISHED'))

DEFAULT 'CREATED',

asigneeId INT FOREIGN KEY REFERENCES [USER](id),

finishTime DATETIME,

startTime DATETIME

);

CREATE TABLE TEAM\_USER

(

id INT IDENTITY(1, 1) PRIMARY KEY,

userId INT FOREIGN KEY REFERENCES "USER"(id),

teamId INT FOREIGN KEY REFERENCES TEAM(id),

"role" VARCHAR(8) NOT NULL

CHECK ("role" IN ('CREATOR', 'USER'))

);

Хранимые процедуры для создания задач и полнотекстового поиска:

CREATE OR ALTER PROCEDURE [Task.Search]

@userId INT,

@teamId INT,

@boardId INT,

@searchQuery NVARCHAR(256),

@skip INT,

@take INT

AS

BEGIN

IF NOT EXISTS (SELECT TOP 1 id FROM [USER\_TOKEN] WHERE userId = @userId)

BEGIN

PRINT 'TOKEN EXPIRED';

RETURN;

END;

IF NOT EXISTS (SELECT TOP 1 id FROM BOARD WHERE id = @boardId)

BEGIN

PRINT 'BOARD NOT EXISTS';

RETURN;

END;

IF NOT EXISTS (SELECT TOP 1 id FROM [TEAM\_USER] WHERE userId = @userId AND teamId = @teamId)

BEGIN

PRINT 'ACCESS DENIED OR SUCH USER NOT EXISTS';

RETURN;

END;

SELECT TASKS.id, title, email, content, attachments, "type", "priority", "severity", "status" FROM TASKS

INNER JOIN DOCUMENT ON TASKS.attachments = DOCUMENT.id

INNER JOIN [USER] ON TASKS.asigneeId = [USER].id

WHERE TASKS.boardId = @boardId AND FREETEXT(DOCUMENT.document, @searchQuery)

OR FREETEXT(TASKS.content, @searchQuery)

OR FREETEXT(TASKS.title, @searchQuery)

ORDER BY TASKS.id

OFFSET @skip ROWS

FETCH NEXT @take ROWS ONLY;;

END;

CREATE OR ALTER PROCEDURE [Task.Create]

@userId INT,

@teamId INT,

@boardId INT,

@title NVARCHAR(64),

@content NVARCHAR(1024),

@type NVARCHAR(11),

@priority NVARCHAR(6),

@severity NVARCHAR(8),

@status NVARCHAR(128),

@asigneeEmail NVARCHAR(256),

@finishTime DATETIME,

@filePath VARCHAR(256)

AS

BEGIN

DECLARE @asigneeId INT = (SELECT TOP 1 id FROM [USER] WHERE email = @asigneeEmail);

IF NOT EXISTS (SELECT TOP 1 id FROM [USER\_TOKEN] WHERE userId = @userId)

BEGIN

PRINT 'TOKEN EXPIRED';

RETURN;

END;

IF NOT EXISTS (SELECT TOP 1 id FROM BOARD WHERE id = @boardId)

BEGIN

PRINT 'BOARD NOT EXISTS';

RETURN;

END;

IF NOT EXISTS (SELECT TOP 1 id FROM [TEAM\_USER] WHERE userId = @userId AND teamId = @teamId)

BEGIN

PRINT 'ACCESS DENIED OR SUCH USER NOT EXISTS';

RETURN;

END;

IF NOT EXISTS (SELECT TOP 1 id FROM [TEAM\_USER] WHERE userId = @asigneeId AND teamId = @teamId)

BEGIN

PRINT 'USER NOT IN TEAM';

RETURN;

END;

BEGIN TRY

IF (@filePath = '')

BEGIN

INSERT INTO TASKS(boardId, title, content, [type], [priority], severity, asigneeId, finishTime, startTime, "status")

VALUES(@boardId, @title, @content, @type, @priority, @severity, @asigneeId, @finishTime, CURRENT\_TIMESTAMP, @status);

END;

ELSE

BEGIN

DECLARE @file\_stream VARBINARY(MAX);

DECLARE @command NVARCHAR(1000);

DECLARE @documentId INT;

DECLARE @documentExtenstion NVARCHAR(16) = (select reverse(left(reverse(@filePath),CHARINDEX('.',reverse(@filePath))-1)));

DECLARE @documentName NVARCHAR(128) = REVERSE(LEFT(REVERSE(@filePath),CHARINDEX( '\',REVERSE(@filePath))-1));

SET @command = N'SELECT @file\_stream1 = CAST(bulkcolumn AS varbinary(MAX))

from OPENROWSET(BULK ''' + @filePath + ''',

SINGLE\_BLOB) ROW\_SET'

EXEC sp\_executesql @command, N'@file\_stream1 VARBINARY(MAX) OUTPUT', @file\_stream1 = @file\_stream OUTPUT;

select @file\_stream;

INSERT INTO DOCUMENT("name", extension, document)

VALUES(@documentName, @documentExtenstion, @file\_stream);

SET @documentId = SCOPE\_IDENTITY();

INSERT INTO TASKS(boardId, title, content, attachments, [type], [priority], severity, asigneeId, finishTime, startTime, "status")

VALUES(@boardId, @title, @content, @documentId, @type, @priority, @severity, @asigneeId, @finishTime, CURRENT\_TIMESTAMP, @status);

END;

END TRY

BEGIN CATCH

SELECT

ERROR\_LINE() AS ErrorLine,

ERROR\_MESSAGE() AS ErrorMessage;

END CATCH

END;

Пример применения хранимых процедур в приложении:

using (TaskboardEntities db = new TaskboardEntities())

{

SqlParameter userId = new SqlParameter("@userId", MainWindow.User.Id);

SqlParameter teamId = new SqlParameter("@teamId", this.teamId);

SqlParameter boardId = new SqlParameter("@boardId", BoardId);

SqlParameter searchQuery = new SqlParameter("@searchQuery", SearchQuery);

SqlParameter skip = new SqlParameter("@skip", currentPage \* pageSize);

SqlParameter take = new SqlParameter("@take", pageSize);

Tasks.Clear();

Tasks = await db.Database.SqlQuery<BoardTask>("[Task.Search] @userId, @teamId, @boardId, @searchQuery, @skip, @take", new[] { userId, teamId, boardId, searchQuery, skip, take }).ToListAsync();

}