Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский биотехнологический университет»

Кафедра *\_\_\_ПЦК «Информационные системы\_\_\_\_\_\_\_*

**ОТЧЕТ**

**по \_\_\_\_\_\_\_\_\_производственной \_\_\_\_\_\_\_\_\_\_\_\_ практике**

*(указать вид или тип практики)*

Место прохождения практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Срок прохождения практики с «24»\_\_\_02\_\_\_\_\_2025г. по «23»\_\_\_03\_\_\_2025г.

Руководитель практики от выпускающей кафедры Воронкова Я.А., преподаватель СПО

 *(ФИО, должность)*

Руководитель практики от организации\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 *(ФИО, должность)*

Выполнил:

студент \_\_\_21-ИСо-1\_\_\_группы \_\_4\_\_ курса

института (колледжа) \_\_МТК\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_Лавринчук К. Д.\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_

 *(ФИО, подпись)*

«\_24\_\_» \_\_\_\_03\_\_\_\_\_\_\_\_ 20\_25\_г.

Москва, 2025 г.

**Тема диплома: «**Разработка системы оповещения администратора о критических событиях в базе данных**»**

**Создание базы данных**Листинг кода:

CREATE DATABASE DataShieldDB;

GO

USE DataShieldDB;

GO

CREATE TABLE dbo.Users (

 UserID INT IDENTITY(1,1) PRIMARY KEY,

 Username NVARCHAR(50) NOT NULL UNIQUE,

 Email NVARCHAR(100) NOT NULL UNIQUE,

 PasswordHash NVARCHAR(256) NOT NULL,

 Role NVARCHAR(50) NOT NULL,

 CreatedAt DATETIME NOT NULL DEFAULT GETDATE()

);

GO

CREATE TABLE dbo.CriticalEvents (

 EventID INT IDENTITY(1,1) PRIMARY KEY,

 EventType NVARCHAR(50) NOT NULL,

 Description NVARCHAR(500) NOT NULL,

 Severity NVARCHAR(20) NOT NULL,

 EventTimestamp DATETIME NOT NULL DEFAULT GETDATE(),

 IsNotified BIT NOT NULL DEFAULT 0

);

GO

CREATE TABLE dbo.EventLogs (

 LogID INT IDENTITY(1,1) PRIMARY KEY,

 EventID INT NOT NULL,

 LogDetail NVARCHAR(MAX) NOT NULL,

 LogTimestamp DATETIME NOT NULL DEFAULT GETDATE(),

 CONSTRAINT FK\_EventLogs\_CriticalEvents FOREIGN KEY (EventID)

 REFERENCES dbo.CriticalEvents(EventID)

);

GO

CREATE TABLE dbo.Thresholds (

 ThresholdID INT IDENTITY(1,1) PRIMARY KEY,

 EventType NVARCHAR(50) NOT NULL,

 ThresholdValue INT NOT NULL,

 CreatedAt DATETIME NOT NULL DEFAULT GETDATE()

);

GO

CREATE TABLE dbo.Notifications (

 NotificationID INT IDENTITY(1,1) PRIMARY KEY,

 EventID INT NOT NULL,

 UserID INT NOT NULL,

 NotificationMethod NVARCHAR(50) NOT NULL,

 NotificationTimestamp DATETIME NOT NULL DEFAULT GETDATE(),

 SentStatus NVARCHAR(20) NOT NULL,

 CONSTRAINT FK\_Notifications\_CriticalEvents FOREIGN KEY (EventID)

 REFERENCES dbo.CriticalEvents(EventID),

 CONSTRAINT FK\_Notifications\_Users FOREIGN KEY (UserID)

 REFERENCES dbo.Users(UserID)

);

GO

CREATE TRIGGER dbo.Trigger\_NotifyCriticalEvent

ON dbo.CriticalEvents

AFTER INSERT

AS

BEGIN

 SET NOCOUNT ON;

 INSERT INTO dbo.Notifications (EventID, UserID, NotificationMethod, SentStatus)

 SELECT i.EventID, u.UserID, 'Email', 'Pending'

 FROM inserted i

 CROSS JOIN dbo.Users u

 WHERE i.Severity = 'High' AND u.Role = 'Administrator';

END;

GO

Скриншот:



**Наполнение БД**

Листинг кода:

IF DB\_ID('DataShieldDB') IS NULL

BEGIN

 CREATE DATABASE DataShieldDB;

END

GO

USE DataShieldDB;

GO

IF OBJECT\_ID('dbo.Users', 'U') IS NULL

BEGIN

 CREATE TABLE dbo.Users (

 UserID INT IDENTITY(1,1) PRIMARY KEY,

 Username NVARCHAR(50) NOT NULL UNIQUE,

 Email NVARCHAR(100) NOT NULL UNIQUE,

 PasswordHash NVARCHAR(256) NOT NULL,

 Role NVARCHAR(50) NOT NULL,

 CreatedAt DATETIME NOT NULL DEFAULT GETDATE()

 );

END

GO

IF OBJECT\_ID('dbo.CriticalEvents', 'U') IS NULL

BEGIN

 CREATE TABLE dbo.CriticalEvents (

 EventID INT IDENTITY(1,1) PRIMARY KEY,

 EventType NVARCHAR(50) NOT NULL,

 Description NVARCHAR(500) NOT NULL,

 Severity NVARCHAR(20) NOT NULL,

 EventTimestamp DATETIME NOT NULL DEFAULT GETDATE(),

 IsNotified BIT NOT NULL DEFAULT 0

 );

END

GO

IF OBJECT\_ID('dbo.EventLogs', 'U') IS NULL

BEGIN

 CREATE TABLE dbo.EventLogs (

 LogID INT IDENTITY(1,1) PRIMARY KEY,

 EventID INT NOT NULL,

 LogDetail NVARCHAR(MAX) NOT NULL,

 LogTimestamp DATETIME NOT NULL DEFAULT GETDATE(),

 CONSTRAINT FK\_EventLogs\_CriticalEvents FOREIGN KEY (EventID)

 REFERENCES dbo.CriticalEvents(EventID)

 );

END

GO

IF OBJECT\_ID('dbo.Thresholds', 'U') IS NULL

BEGIN

 CREATE TABLE dbo.Thresholds (

 ThresholdID INT IDENTITY(1,1) PRIMARY KEY,

 EventType NVARCHAR(50) NOT NULL,

 ThresholdValue INT NOT NULL,

 CreatedAt DATETIME NOT NULL DEFAULT GETDATE()

 );

END

GO

IF OBJECT\_ID('dbo.Notifications', 'U') IS NULL

BEGIN

 CREATE TABLE dbo.Notifications (

 NotificationID INT IDENTITY(1,1) PRIMARY KEY,

 EventID INT NOT NULL,

 UserID INT NOT NULL,

 NotificationMethod NVARCHAR(50) NOT NULL,

 NotificationTimestamp DATETIME NOT NULL DEFAULT GETDATE(),

 SentStatus NVARCHAR(20) NOT NULL,

 CONSTRAINT FK\_Notifications\_CriticalEvents FOREIGN KEY (EventID)

 REFERENCES dbo.CriticalEvents(EventID),

 CONSTRAINT FK\_Notifications\_Users FOREIGN KEY (UserID)

 REFERENCES dbo.Users(UserID)

 );

END

GO

IF OBJECT\_ID('dbo.Trigger\_NotifyCriticalEvent', 'TR') IS NOT NULL

 DROP TRIGGER dbo.Trigger\_NotifyCriticalEvent;

GO

CREATE TRIGGER dbo.Trigger\_NotifyCriticalEvent

ON dbo.CriticalEvents

AFTER INSERT

AS

BEGIN

 SET NOCOUNT ON;

 INSERT INTO dbo.Notifications (EventID, UserID, NotificationMethod, SentStatus)

 SELECT i.EventID, u.UserID, 'Email', 'Pending'

 FROM inserted i

 CROSS JOIN dbo.Users u

 WHERE i.Severity = 'High' AND u.Role = 'Administrator';

END;

GO

INSERT INTO dbo.Users (Username, Email, PasswordHash, Role)

VALUES

('admin1', 'admin1@datashield.com', 'hash\_admin1', 'Administrator'),

('admin2', 'admin2@datashield.com', 'hash\_admin2', 'Administrator'),

('operator1', 'operator1@datashield.com', 'hash\_operator1', 'Operator'),

('operator2', 'operator2@datashield.com', 'hash\_operator2', 'Operator'),

('operator3', 'operator3@datashield.com', 'hash\_operator3', 'Operator');

GO

INSERT INTO dbo.Thresholds (EventType, ThresholdValue)

VALUES

('Unauthorized Access', 5),

('Data Integrity Violation', 3),

('Service Outage', 2),

('High Load', 10);

GO

DECLARE @i INT = 1;

DECLARE @max INT = 1000;

DECLARE @eventType NVARCHAR(50);

DECLARE @severity NVARCHAR(20);

DECLARE @randInt INT;

WHILE @i <= @max

BEGIN

 SET @eventType = CASE (@i % 4)

 WHEN 0 THEN 'Unauthorized Access'

 WHEN 1 THEN 'Data Integrity Violation'

 WHEN 2 THEN 'Service Outage'

 ELSE 'High Load'

 END;

 SET @randInt = ABS(CHECKSUM(NEWID())) % 100;

 SET @severity = CASE

 WHEN @randInt < 20 THEN 'High'

 WHEN @randInt < 50 THEN 'Medium'

 ELSE 'Low'

 END;

 INSERT INTO dbo.CriticalEvents (EventType, Description, Severity, EventTimestamp)

 VALUES (@eventType, CONCAT('Описание события №', @i, ' типа ', @eventType), @severity, DATEADD(SECOND, @i, GETDATE()));

 SET @i = @i + 1;

END;

GO

INSERT INTO dbo.EventLogs (EventID, LogDetail)

SELECT EventID, CONCAT('Подробная информация для события №', EventID)

FROM dbo.CriticalEvents;

GO

Скриншот:





Клиент-серверная архитектура подразумевает разделение функционала на две основные компоненты:

Серверная часть:

Обрабатывает запросы, управляет бизнес-логикой и взаимодействует с базой данных. Здесь располагается центральный сервер приложений и база данных, обеспечивающие хранение, обработку и безопасность данных.

Клиентская часть:
Отвечает за взаимодействие с пользователем. Клиент (веб-приложение, мобильное приложение или десктопное приложение) отправляет запросы к серверу, получает данные и отображает их пользователю, обеспечивая интуитивный и быстрый интерфейс.