



universidade
de aveiro

deti departamento de eletrónica,
telecomunicações e informática

Routine View

Base de dados de uma aplicação de gestão de rotina gamificada

Universidade de Aveiro

Licenciatura em Engenharia de Computadores e Informática

Base de Dados P10G6

Regente: Prof. Carlos Costa (carlos.costa@ua.pt)

Professor Orientador: Prof. Joaquim Sousa Pinto (jsp@ua.pt)

Miguel Soares Francisco, 108304- 50%

Rafael Kauati, 105925 - 50%

Índice

Introdução.....	3
Aplicação.....	4
Análise de Requisitos.....	5
Entidades centrais.....	6
Requisitos Funcionais.....	6
Diagrama Entidade Relação (DER).....	7
Esquema Relacional (ER).....	7
DDL.....	8
DML.....	9
Stored Procedure (SP).....	10
Views.....	15
User Defined Function (UDF).....	16
Triggers.....	18
Indexes.....	22
Cursorres.....	23
Segurança.....	26
Conclusão.....	28

Introdução

No âmbito da unidade curricular de Base de Dados, da Licenciatura em Engenharia de Computadores e Informática foi-nos proposto a criação de um projeto que fizesse a gestão de um sistema funcional e com complexidade razoável aplicável ao mundo real. O seguinte relatório irá descrever sucintamente o nosso projeto dando ênfase nas partes que achamos mais fulcrais para o funcionamento do mesmo.

Neste projeto, escolhemos apresentar uma aplicação de gestão de rotina gamificada, pois as pessoas podem ter uma rotina muito ocupada e querer organizá-la melhor e o uso de um sistema de gestão de rotina gamificado pode ajudá-la a se organizar e se manter motivada.

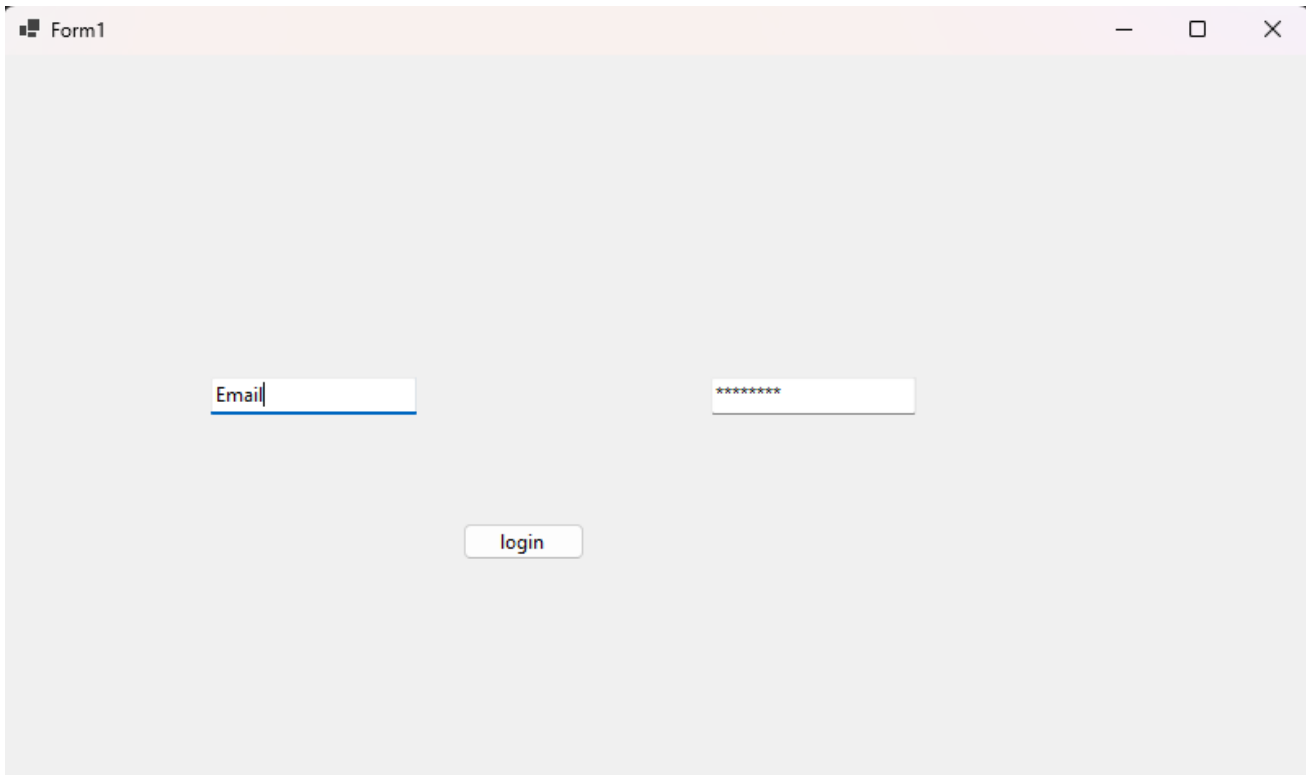
Importante : Os **scripts SQL** se encontram na pasta **SQL/** do projeto e aplicação feita em **Windows Forms** na pasta **App/**. Para abrir a aplicação WF, recomenda-se utilizar a opção **“Abrir um projeto ou uma solução”**, na página inicial do **Visual Studio** e **Navegar até a pasta App e selecionar** a solução **Routine-View-Forms.sln**

No ficheiro **ConnectionStringHelper.cs** mudar na variável estática **“connectionString”** os campos ID e Password para os dos respectivos utilizadores.

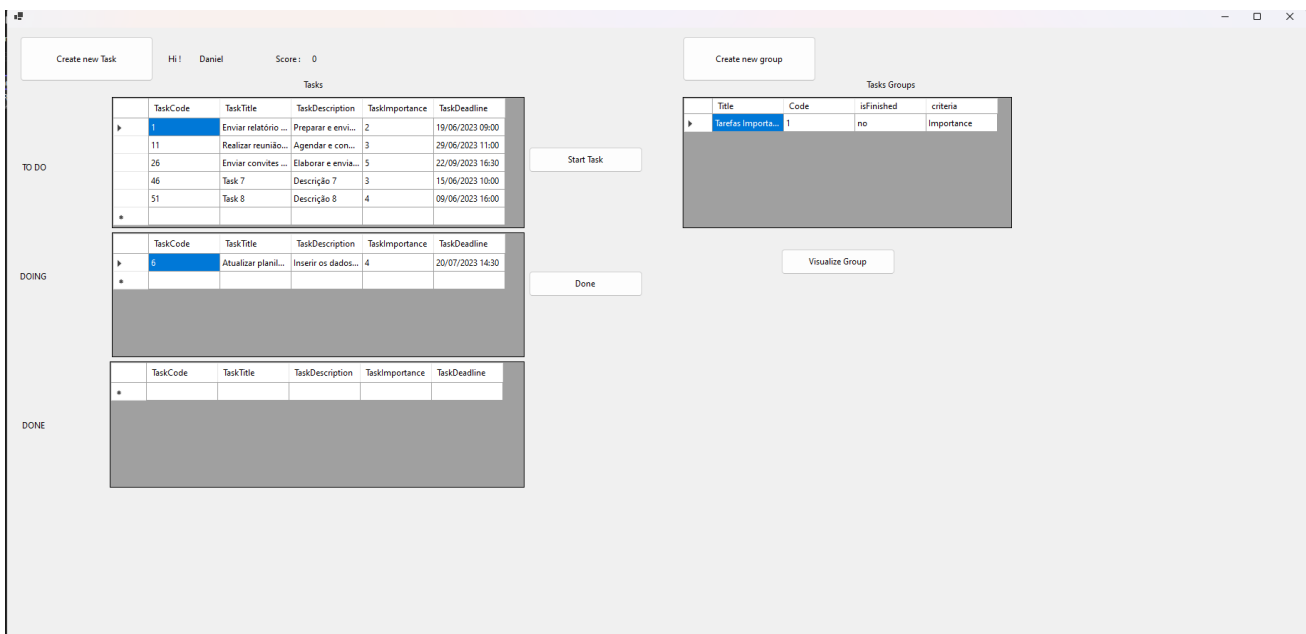
Por fim, a aplicação estará pronta para conectar com a **base de dados** das aulas e **operar** com a **camada de dados** do projeto.

Aplicação

A aplicação desenvolvida como um **Windows Forms App** foi feito para testar as funcionalidades mais básicas e centrais, até então feitas e prontamente testadas :



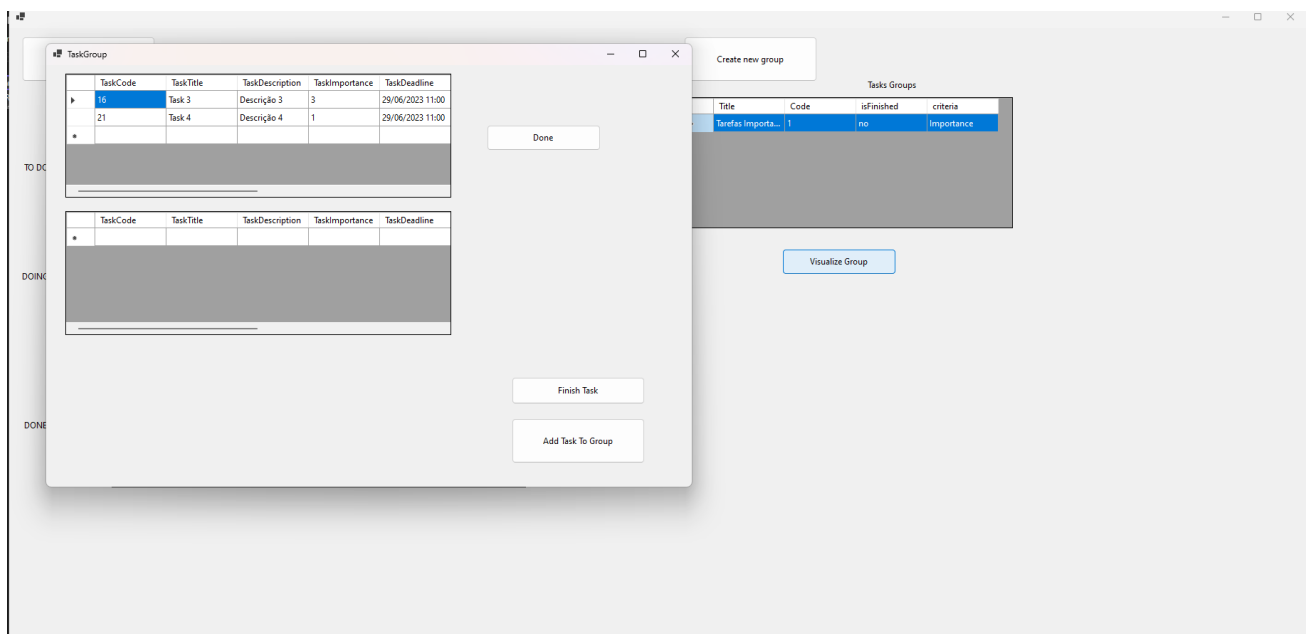
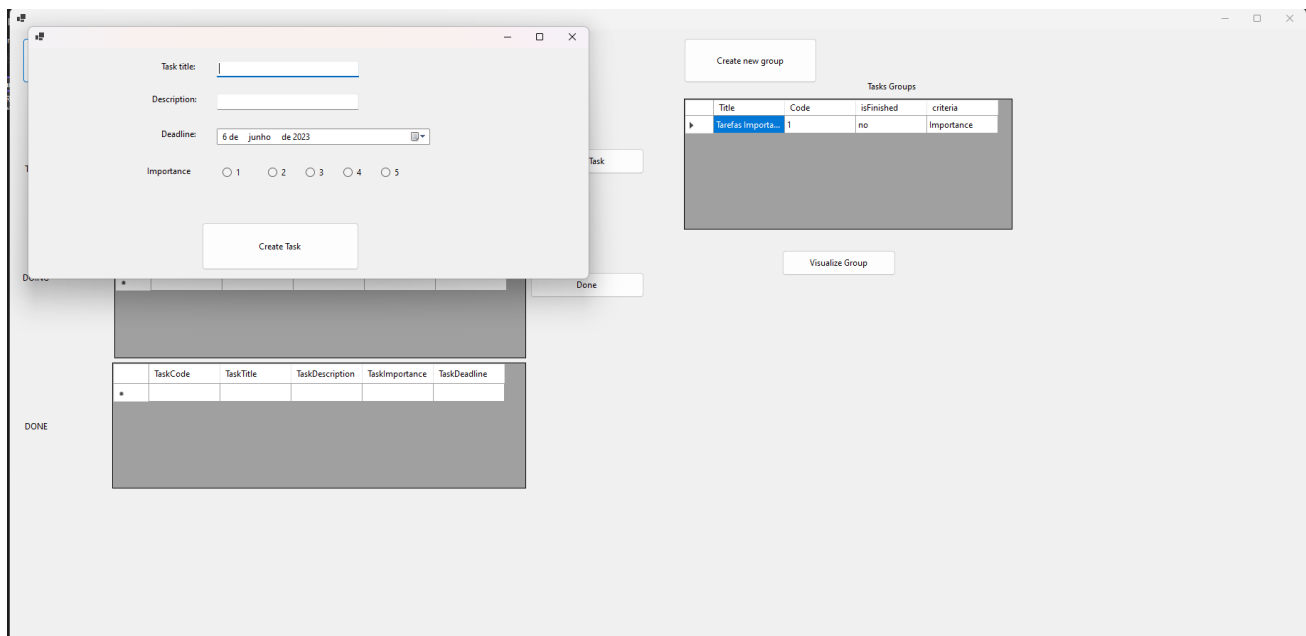
The screenshot shows a simple login form within a window titled 'Form1'. It contains two input fields: one for 'Email' and another for a password, which is masked with asterisks. Below these fields is a button labeled 'login'.



The screenshot displays a task management application. On the left, there are three sections: 'TO DO', 'DOING', and 'DONE'. Each section contains a table of tasks with columns for TaskCode, TaskTitle, TaskDescription, TaskImportance, and TaskDeadline. The 'TO DO' section has a 'Start Task' button, and the 'DOING' section has a 'Done' button. On the right, there is a 'Tasks Groups' section with a table showing group details (Title, Code, isfinished, criteria) and a 'Visualize Group' button. At the top, there are buttons for 'Create new Task' and 'Create new group', along with user information (Hi! Daniel) and a score (Score: 0).

TaskCode	TaskTitle	TaskDescription	TaskImportance	TaskDeadline
1	Enviar relatório ...	Preparar e envi...	2	19/06/2023 09:00
11	Realizar reunião...	Agendar e con...	3	29/06/2023 11:00
26	Enviar convites ...	Elaborar e envia...	5	22/09/2023 16:30
46	Task 7	Descrição 7	3	15/06/2023 10:00
51	Task 8	Descrição 8	4	09/06/2023 16:00

Title	Code	isfinished	criteria
Tarefas Importa...	1	no	Importance



Análise de Requisitos

Nesta aplicação as tarefas são organizadas em uma estrutura de stacks.

Cada task possui informações como título e descrição. Possuem ainda um estado e uma posição na stack de tasks na qual ela pertence.

É possível ainda criar grupos de tasks que são associadas baseadas em um dos critérios de : importância, deadline ou categoria.

Em um nível mais técnico, a posição de cada task na stack é determinada pelo seu grau de prioridade, determinada usando a lógica de matriz de eisenhower pelo grau de importância e deadline de cada task (dados pelo utilizador). No caso de um grupo de task, a prioridade do grupo é a média de todas as prioridades de cada task do grupo.

Também é possível atribuir uma “recompensa” para a conclusão de uma task ou um grupo de tasks, esta recompensa vem na forma de “pontos” que são somados ao total de pontos do utilizador.

Ao realizar algum número de tasks ou atingir alguma meta, o utilizador recebe uma “conquista” (Semelhante a um achievement de um jogo), o qual possui uma descrição e está associado a uma categoria.

Em suma, a aplicação consiste em criar, editar, visualizar e deletar tarefas (ou conjuntos de tarefas), listar estas, além de listas de recompensas e conquistas.

Entidades centrais

- Task(s)
- Stack(s)
- Grupo(s) de Tasks
- Recompensas
- Conquistas

Requisitos Funcionais:

- Utilizador ser capaz criar, modificar e deletar uma task, ou um grupo de tasks
- Utilizador ser capaz de modificar dados(descrição, importância, urgência, etc) das tasks ou grupo de tasks
- Utilizador ser capaz de pesquisar uma task ou grupo de tasks por nome, grau de importância, urgência, deadline, data de criação, etc
- Utilizador se capaz de visualizar informações sobre a task ou grupo de taks
- Para cada grupo, deve ser associadas task baseadas em um dos critérios : Importância, Urgência (Deadline) ou Categoria
- Associação de uma task ou um grupo de tasks a uma recompensa
- Capacidade de visualizar as task ou de grupos de tasks já concluídas
- Capacidade de listar as recompensas(e seus dados) associadas a uma task ou a um grupo de tasks
- Visualizar todas as conquistas, ou uma única detalhada

Diagrama Entidade Relação (DER)

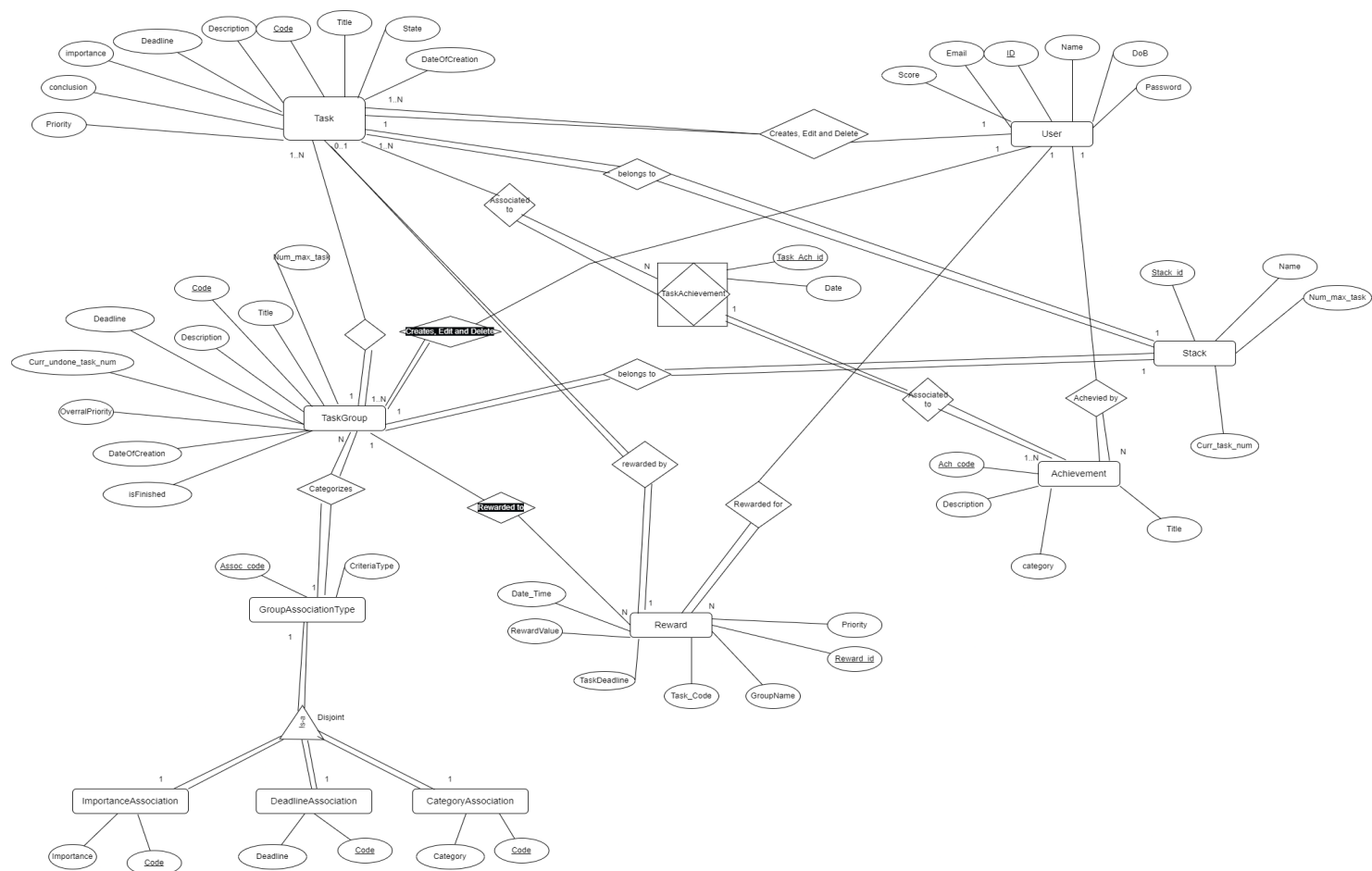
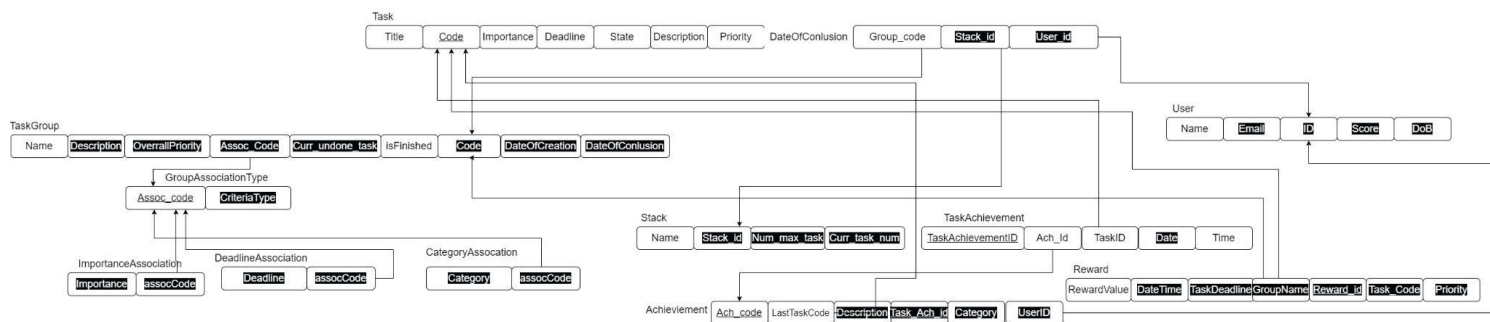


Diagrama Entidade Relação

Esquema Relacional (ER)



Esquema Relacional

DDL :

```
create table [User] (
    ID int identity(10,5),
    [Name] varchar(256) NOT NULL,
    [Email] varchar(40) NOT NULL,
    [Password] varbinary(8000) NOT NULL,
    [Score] int default 0,
    [DoB] date NOT NULL
)
go

create unique clustered index user_ID_index on [User](ID);

alter index user_ID_index on [User] rebuild with (fillfactor = 100) ;

create table [Stack] (
    StackID int identity(1,1),
    [Name] varchar(40) not null,
    NumMaxTasks int not null default 20,
    CurrMaxTasks int not null default 0
);
go
```

```
CREATE TABLE Task_Group_Assoc (
    Assoc_Code INT IDENTITY (1,1) primary key,
    CriteriaType VARCHAR(20) CHECK (CriteriaType IN ('Importance', 'Deadline', 'Category'))
);

create table Task_Group(
    Code int identity (1,1) primary key,
    Assoc_code int foreign key references [Task_Group_Assoc] (Assoc_Code),
    [StackID] int foreign key references [Stack] (StackID),
    [userID] int foreign key references [User] (ID),
    [Title] varchar(50) not null,
    [Description] varchar(100) ,
    [isFinished] varchar(5) default 'no',
    [Num_max_task] int default 10,
    [Curr_undone_task_num] int not null default 0,
    [OverallPriority] int not null default 5,
    [DateOfCreation] datetime NOT NULL
);

create table [Task] (
    Code int identity(1,5),
    Title varchar(100) NOT NULL UNIQUE,
    [Description] varchar(100) NOT NULL,
    Importance int NOT NULL,
    Deadline datetime NOT NULL,
    [State] varchar(10) ,
    [Priority] int,
    [StackID] int foreign key references [Stack] (StackID),
    [Conclusion] datetime default null,
    [UserID] int foreign key references [User] (ID),
    [TaskGroupCode] int foreign key references [Task_Group] (Code)
)
```



```

create table Reward(
    Reward_id int identity(1,5),
    [Priority] int default null,
    [Group_Name] varchar(50) default null,
    [Task_code] int foreign key references [Task] (Code),
    [Task_Deadline] datetime NOT NULL,
    [Reward_Value] int not null default 10,
    [Date_Time] datetime NOT NULL,
);

create table Achievement (
    Ach_code int identity(1,2),
    [Description] varchar(1000) default '',
    [WasAchieved] varchar(4) default 'no',
    Category varchar(20) default null,
    [UserID] int foreign key references [User] (ID)
);
go

create unique clustered index Ach_code_index on Achievement(Ach_code);
go
alter index Ach_code_index on [Achievement] rebuild with (fillfactor = 70) ;
go

create table TaskAchievement (
    TaskAchievement int identity(1,5),
    Ach_code int foreign key references [Achievement] (Ach_code),
    TaskID int foreign key references [Task] (Code),
);
go

```

DML + uso de algumas funcionalidades da camada de dados :

```

INSERT INTO [User] ([Name], [Email], [Password], [DoB])
VALUES
    ('Daniel', 'daniel@ua.pt', CONVERT(VARBINARY(8000), 'password'), '2011-01-01'),
    ('Pedro Ramos', 'PR@ua.pt', CONVERT(VARBINARY(8000), 'anotherpassword'), '2019-01-01');
go

INSERT INTO [Stack] ([Name]) VALUES ('To Do');
INSERT INTO [Stack] ([Name]) VALUES ('Doing');
INSERT INTO [Stack] ([Name]) VALUES ('Done');

INSERT INTO [Task] ([Title], [Description], [Importance], [Deadline], [UserID])
VALUES ('Enviar relatório semanal', 'Preparar e enviar relatório de desempenho semanal da equipe', 2, '2023-06-19 09:00:00', 10),
('Atualizar planilha de vendas', 'Inserir os dados de vendas atualizados na planilha de acompanhamento', 4, '2023-07-20 14:30:00', 10),
('Realizar reunião de equipe', 'Agendar e conduzir uma reunião com a equipe para alinhar as metas do mês', 3, '2023-06-29 11:00:00', 10),
('Task 3', 'Descrição 3', 3, '2023-06-29 11:00:00', 10),
('Task 4', 'Descrição 4', 1, '2023-06-29 11:00:00', 10),
('Enviar convites para evento', 'Elaborar e enviar convites para os participantes do evento corporativo', 5, '2023-09-22 16:30:00', 10),
('Task 2', 'Elaborar e enviar convites para os participantes do evento corporativo', 5, '2023-09-22 16:30:00', 15),
('Task 5', 'Elaborar e enviar convites para os participantes do evento corporativo', 5, '2023-09-22 16:30:00', 15),
('Task 6', 'Descrição 6', 2, '2023-06-12 13:45:00', 15),
('Task 7', 'Descrição 7', 3, '2023-06-15 10:00:00', 10),
('Task 8', 'Descrição 8', 4, '2023-06-09 16:00:00', 10);

```

```

insert into [Task_Group_Assoc] (CriteriaType) values
('Importance'), ('Deadline'), ('Category');

execute createTaskGroup @title = 'Tarefas Importantes', @description = 'Tarefas Importantes a serem feitas',
@assoc_type = 'Importance', @userid = 10;

EXECUTE addTaskToGroup
@TaskGroupTitle = 'Tarefas Importantes', @TaskTitle = 'Task 3';

EXECUTE addTaskToGroup
@TaskGroupTitle = 'Tarefas Importantes', @TaskTitle = 'Task 4';

SELECT * FROM getTaskGroup('Tarefas Importantes', 'ToDo');

EXECUTE startTask @taskname = 'Atualizar planilha de vendas';

select * from [Task];
select * from [User];
select * from [Stack];
select * from [Reward];
select * from getAllTaskGroups(10);

```

Stored Procedure (SP)

Criamos também stored procedures que são utilizados como **handlers** para boa parte dos requisitos funcionais do sistema. São procedimentos que, em sua maioria, fazem múltiplas modificações em várias tabelas da base de dados, por isso quase todos possuem verificações das **transações**, com **commits** em casos das operações forem bem sucedidas e **rollbacks** em caso contrários, incluindo mensagens de erros personalizadas.

```

1  --use [Routine View]
2  --go
3
4  DROP PROCEDURE IF EXISTS startTask
5  GO
6
7  CREATE PROCEDURE startTask(@taskname varchar(100))
8  AS
9  BEGIN
10     BEGIN TRY
11         BEGIN TRANSACTION;
12         UPDATE [Stack]
13         SET CurrMaxTasks = CurrMaxTasks + 1
14         WHERE StackID = 2;
15
16
17         UPDATE [Stack]
18         SET CurrMaxTasks = CurrMaxTasks - 1
19         WHERE StackID = 1;
20
21         UPDATE T
22         SET T.StackID = T.StackID + 1
23         FROM [Task] T
24         WHERE T.Title = @taskname;
25
26         COMMIT TRANSACTION;
27     END TRY
28     BEGIN CATCH
29         IF @@TRANCOUNT > 0
30             PRINT 'Error on starting task : '+@taskname;
31             ROLLBACK;
32     END CATCH
33 END
34 GO
35
36
37 DROP PROCEDURE IF EXISTS concludeTask
38 GO
39
40 CREATE PROCEDURE concludeTask (@taskname VARCHAR(30))
41 AS
42 BEGIN
43     SET NOCOUNT ON;
44
45     BEGIN TRY
46         BEGIN TRANSACTION;
47
48         DECLARE @taskcode INT;
49         DECLARE @taskdeadline DATETIME;
50         DECLARE @taskpriority INT;
51
52         SELECT @taskcode = Code, @taskdeadline = Deadline, @taskpriority = [Priority]
53         FROM [Task]
54         WHERE Title = @taskname;
55
56         UPDATE [Task]
57         SET StackID = 3, Task.Conclusion = GETDATE()
58         WHERE Title = @taskname;
59
60         UPDATE [Stack]
61         SET CurrMaxTasks = CurrMaxTasks + 1
62         WHERE [Stack].StackID = 3;
63
64         UPDATE [Stack]
65         SET CurrMaxTasks = CurrMaxTasks - 1
66         WHERE [Stack].StackID = 2;
67
68         INSERT INTO [Reward] ([Task_code], [Task_Deadline], [Date_Time], [Reward_Value])
69         VALUES (@taskcode, ISNULL(@taskdeadline, GETDATE()), GETDATE(), @taskpriority * 2);
70
71         COMMIT TRANSACTION;
72     END TRY

```

```

71      COMMIT TRANSACTION;
72  END TRY
73  BEGIN CATCH
74      IF @@TRANCOUNT > 0
75      BEGIN
76          PRINT 'Error on concluding task : '+@taskname;
77          ROLLBACK TRANSACTION;
78      END
79  END CATCH;
80  END
81  GO
82
83  DROP PROCEDURE IF EXISTS finishGroupTask
84  GO
85
86  CREATE PROCEDURE finishGroupTask (@taskGroup varchar(50))
87  AS
88  BEGIN
89      SET NOCOUNT ON;
90
91      BEGIN TRY
92          BEGIN TRANSACTION;
93
94          DECLARE @unfinishedTasks int;
95
96          SELECT @unfinishedTasks = [Curr_undone_task_num]
97          FROM Task_Group
98          WHERE Task_Group.Title = @taskGroup;
99
100         IF @unfinishedTasks = 0
101         BEGIN
102             UPDATE Task_Group
103             SET Task_Group.isFinished = 'yes'
104             WHERE Task_Group.Title = @taskGroup;
105
106             DECLARE @sumPriority int;
107
108             DECLARE @lastTaskCode int;
109             DECLARE @lastTaskDeadline datetime;
110
111             SELECT @sumPriority = sum(TaskPriority)
112             FROM getTaskGroup(@taskGroup, 'Done');
113
114             SELECT @lastTaskCode = t.TaskCode, @lastTaskDeadline = t.TaskDeadline
115             FROM dbo.getTaskGroup(@taskGroup, 'Done') t
116             WHERE t.TaskConclusion = (SELECT max(TaskConclusion) FROM dbo.getTaskGroup(@taskGroup, 'Done'));
117
118             INSERT INTO [Reward] ([Task_code], [Task_Deadline], [Date_Time], [Reward_Value])
119             VALUES (@lastTaskCode, ISNULL(@lastTaskDeadline, GETDATE()), GETDATE(), @sumPriority * 2);
120         END
121     ELSE
122     BEGIN
123         UPDATE Task_Group
124         SET Task_Group.isFinished = 'no'
125         WHERE Task_Group.Title = @taskGroup;
126     END
127
128     COMMIT TRANSACTION;
129  END TRY
130  BEGIN CATCH
131      IF @@TRANCOUNT > 0
132      BEGIN
133          PRINT 'Error on finishing task group : '+@taskGroup;
134          ROLLBACK TRANSACTION;
135      END
136  END CATCH;
137  END
138  GO
139
140  DROP PROCEDURE IF EXISTS checkLogIn;

```

```

140 DROP PROCEDURE IF EXISTS checkLogIn;
141 GO
142
143 CREATE PROCEDURE checkLogIn
144 (
145     @email varchar(40),
146     @password varchar(40),
147     @confirmation int OUTPUT
148 )
149 AS
150 BEGIN
151     DECLARE @DecryptedPassword varbinary(8000);
152     DECLARE @userid int;
153
154     SELECT @DecryptedPassword = DecryptByPassPhrase('ThePassphrase', [Password])
155     FROM [User]
156     WHERE [Email] = @email;
157
158     IF CONVERT(varchar(40), @DecryptedPassword) = @password
159     BEGIN
160         --DECLARE @username varchar(50);
161         SELECT @userid = [User].ID--, @username = [User].[Name]
162         FROM [User]
163         WHERE [Email] = @email;
164         --EXEC sp_addlogin @username, @DecryptedPassword, [Routine View];
165
166         SET @confirmation = @userid;
167     END
168     ELSE
169     BEGIN
170         SET @confirmation = 0;
171     END
172 END;
173 GO
174
175
176
177 DROP PROCEDURE IF EXISTS addTaskToGroup
178 GO
179
180 CREATE PROCEDURE addTaskToGroup
181     @TaskGroupTitle varchar(50),
182     @TaskTitle varchar(100)
183 AS
184 BEGIN
185     SET NOCOUNT ON;
186
187     BEGIN TRY
188         DECLARE @ErrorUpdates TABLE (
189             TaskTitle varchar(100),
190             TaskGroupTitle varchar(50)
191         );
192
193         BEGIN TRANSACTION;
194
195         -- Update the task with the group code
196         DECLARE @TaskGroupCode int;
197
198         SELECT @TaskGroupCode = Code
199         FROM Task_Group
200         WHERE Title = @TaskGroupTitle;
201
202         UPDATE Task_Group
203         SET Curr_undone_task_num = Curr_undone_task_num + 1
204         WHERE Title = @TaskGroupTitle;
205
206         IF @@ROWCOUNT = 0
207         BEGIN
208             INSERT INTO @ErrorUpdates (TaskTitle, TaskGroupTitle)
209             VALUES (@TaskTitle, @TaskGroupTitle);
210         END
211     END TRY

```

```

211
212 UPDATE Task
213 SET TaskGroupCode = @TaskGroupCode, Task.[State] = 'ToDo'
214 WHERE Title = @TaskTitle;
215
216 IF @@ROWCOUNT = 0
217 BEGIN
218     INSERT INTO @ErrorUpdates (TaskTitle, TaskGroupTitle)
219     VALUES (@TaskTitle, @TaskGroupTitle);
220 END
221
222 -- Update the overall priority of the group task
223 DECLARE @overallPrior int;
224
225 SELECT @overallPrior = AVG([Priority])
226 FROM [Task_Group] g
227 JOIN [Task] t ON g.Code = t.TaskGroupCode
228 WHERE g.Title = @TaskGroupTitle;
229
230 UPDATE Task_Group
231 SET Task_Group.OverallPriority = @overallPrior
232 WHERE Task_Group.Title = @TaskGroupTitle;
233
234 COMMIT TRANSACTION;
235 END TRY
236 BEGIN CATCH
237     IF @@TRANCOUNT > 0
238     BEGIN
239         PRINT 'Error on adding task ' + @TaskTitle + ' to group ' + @TaskGroupTitle;
240         ROLLBACK TRANSACTION;
241     END
242 END CATCH;
243 END
244 GO
245
246
247 DROP PROCEDURE IF EXISTS concludeTaskOfTheGroup
248 GO
249
250
251 CREATE PROCEDURE concludeTaskOfTheGroup
252     @TaskTitle varchar(100)
253 AS
254 BEGIN
255     SET NOCOUNT ON;
256
257     BEGIN TRY
258         BEGIN TRANSACTION;
259
260         DECLARE @TaskGroupCode int;
261
262         SELECT @TaskGroupCode = [TaskGroupCode]
263         FROM Task
264         WHERE Task.Title = @TaskTitle;
265
266         UPDATE Task_Group
267         SET Curr_undone_task_num = Curr_undone_task_num - 1
268         WHERE Code = @TaskGroupCode;
269
270         UPDATE Task
271         SET Task.[State] = 'Done', Task.Conclusion = GETDATE()
272         WHERE Title = @TaskTitle;
273
274         COMMIT TRANSACTION;
275     END TRY
276     BEGIN CATCH
277         IF @@TRANCOUNT > 0
278         PRINT 'Error on concluding task '+@TaskTitle+' from task group : '+str(@TaskGroupCode);
279         ROLLBACK TRANSACTION;

```

```

277         IF @@TRANCOUNT > 0
278             PRINT 'Error on concluding task '+@TaskTitle+ ' from task group : '+str(@TaskGroupCode);
279             ROLLBACK TRANSACTION;
280
281     END CATCH;
282 END
283 GO
284
285
286 DROP PROCEDURE IF EXISTS createTaskGroup
287 GO
288
289 CREATE PROCEDURE createTaskGroup
290     @title varchar(50),
291     @description varchar(100),
292     @assoc_type varchar(20),
293     @userid int
294 AS
295 BEGIN
296     declare @assoc_code int;
297
298     select @assoc_code = tg.Assoc_Code
299     from [Task_Group_Assoc] tg
300     where tg.CriteriaType = @assoc_type;
301
302     insert into Task_Group ([Title],[Description],[Assoc_code], [DateOfCreation] , [userID])
303     values (@title,@description,@assoc_code, GETDATE(),@userid );
304
305 END

```

Views

Criamos algumas views básicas que facilitam a visualização de tarefas baseadas no grau de suas prioridades, assim, caso necessário, para visualizar tarefas sem precisar criar uma consulta nova para isso.

```

1  --use [Routine View]
2  --go
3
4  drop view if exists high_Priority_Tasks
5  go
6
7  create view high_Priority_Tasks
8  as
9
10     select t.Title, t.Description, t.Deadline,t.Priority, t.Conclusion, t.TaskGroupCode ,s.[Name] as Stack
11     from [Task] t
12     join [Stack] s on t.StackID = s.StackID
13     where t.Priority >= 3;
14 go
15
16
17 drop view if exists low_Priority_Tasks
18 go
19
20 create view low_Priority_Tasks
21 as
22
23     select t.Title, t.Description, t.Deadline,t.Priority, t.Conclusion, t.TaskGroupCode ,s.[Name] as Stack
24     from [Task] t
25     join [Stack] s on t.StackID = s.StackID
26     where t.Priority < 3;
27 go
28
29

```

User Defined Function (UDF)

Temos 3 udfs utilizadas para consultas de tasks e de grupos de tasks(retornam tabelas com os campos mais importantes de cada), todas do tipo tabela (Table-Valued Function)

```
Run on active connection | Select block
1  --use [Routine View]
2  --go
3
4  drop function if exists GetAllTasksInStack
5  go
6
7  drop function if exists getTaskGroup
8  go
9
10
11 drop function if exists getAllTaskGroups
12 go
13
14 create function getAllTaskGroups
15     (@userid int)
16 RETURNS TABLE
17 AS
18 RETURN(
19     select distinct g.Code, g.[Title], g.[isFinished],a.CriteriaType
20     from   [Task_Group] g
21     join [Task_Group_Assoc] a on g.Assoc_code = a.Assoc_Code
22     where g.[userID] = @userid
23 );
24 go
25
26
27 create function getTaskGroup
28     (@TaskGroupTitle varchar(50), @state varchar(10))
29 RETURNS TABLE
30 AS
31 RETURN(
32     select
33         t.Code AS TaskCode,
34         t.Title AS TaskTitle,
35         t.Description AS TaskDescription,
36         t.Importance AS TaskImportance,
37         t.Deadline AS TaskDeadline,
38         t.State AS TaskState,
39         t.Priority AS TaskPriority,
40         t.StackID,
41         t.Conclusion AS TaskConclusion,
42         t.UserID
43     from
44         [Task_Group] g
45     join [Task] t on g.Code = t.TaskGroupCode
46     where
47         g.Title = @TaskGroupTitle AND t.[State] = @state
48 );
49 go
50
```



```

48 );
49 go
50
51 drop function if exists GetAllTasksInStack
52 go
53
54 CREATE FUNCTION GetAllTasksInStack
55     (@stackName VARCHAR(40), @userid int)
56 RETURNS TABLE
57 AS
58 RETURN
59 (
60     SELECT
61         t.Code AS TaskCode,
62         t.Title AS TaskTitle,
63         t.TaskGroupCode AS TaskGroup,
64         t.Description AS TaskDescription,
65         t.Importance AS TaskImportance,
66         t.Deadline AS TaskDeadline,
67         t.State AS TaskState,
68         t.Priority AS TaskPriority,
69         t.StackID,
70         t.Conclusion AS TaskConclusion,
71         t.UserID
72     FROM
73         [Stack] s
74     JOIN [Task] t ON s.StackID = t.StackID
75     WHERE
76         s.[Name] = @stackName
77         AND t.TaskGroupCode IS NULL AND t.UserID = @userid
78 );
79 go

```

Triggers

Temos também triggers(gatilhos) **after** e **instead of** que funcionam como **handlers** para inserção e atualização das entidades, em especial vale destacar o **update_priority** que implementa a lógica de atribuição da prioridade para uma **task** baseado nos valores das colunas **deadline** e **importance**.

```
1  --use [Routine View]
2  --go
3
4  DROP TRIGGER IF EXISTS updateTask;
5  GO
6
7  CREATE TRIGGER updateTask ON [Task]
8  INSTEAD OF UPDATE
9  AS
10 BEGIN
11     DECLARE @Title VARCHAR(100),
12             @Description VARCHAR(100),
13             @Importance INT,
14             @Priority INT,
15             @StackID INT,
16             @State varchar(10),
17             @TaskGroupCode int,
18             @Deadline DATETIME;
19
20     DECLARE cursorTasks CURSOR FOR
21     SELECT Title, [Description], Importance,[Priority], StackID ,State, TaskGroupCode,Deadline
22     FROM inserted;
23
24     OPEN cursorTasks;
25
26     FETCH NEXT FROM cursorTasks INTO @Title, @Description, @Importance, @Priority,@StackID, @State,@TaskGroupCode ,@Deadline;
27
28     WHILE @@FETCH_STATUS = 0
29     BEGIN
30         UPDATE t
31         SET t.Title = ISNULL(i.Title, t.Title),
32             t.[Description] = ISNULL(i.[Description], t.[Description]),
33             t.Importance = ISNULL(i.Importance, t.Importance),
34             t.Priority = ISNULL(i.Priority, t.Priority),
35             t.StackID = ISNULL(i.StackID, t.StackID),
36             t.State = ISNULL(i.State, t.State),
37             t.[TaskGroupCode] = ISNULL(i.[TaskGroupCode], t.[TaskGroupCode]),
38             t.Deadline = ISNULL(i.Deadline, t.Deadline)
39         FROM [Task] t
40         INNER JOIN inserted i ON t.[Code] = i.[Code];
41
42         FETCH NEXT FROM cursorTasks INTO @Title, @Description, @Importance, @Priority,@StackID, @State,@TaskGroupCode ,@Deadline;
43     END;
44
45     CLOSE cursorTasks;
46     DEALLOCATE cursorTasks;
47 END;
48 GO
49
50
51
52 DROP TRIGGER IF EXISTS update_priority
53 GO
54
55 CREATE TRIGGER update_priority ON Task
56 AFTER INSERT
57 AS
58 BEGIN
59     SET NOCOUNT ON;
60
61     IF @@TRANCOUNT > 0
62     BEGIN
63         IF XACT_STATE() = 1
64         BEGIN
65             UPDATE t
66             SET t.[Priority] = CASE
67                 WHEN DATEDIFF(MINUTE, GETDATE(), i.[Deadline]) / 60 BETWEEN 0 AND 24 THEN
68                     CASE
69                         WHEN i.[Importance] >= 1 AND i.[Importance] < 3 THEN 3
70                         WHEN i.[Importance] >= 3 AND i.[Importance] <= 5 THEN 5
71                     END
72                 ELSE 1
73             END
74             FROM Task t
75             INNER JOIN inserted i ON t.[Code] = i.[Code];
76         END
77     END
```

```

58 BEGIN
59 SET NOCOUNT ON;
60
61 IF @@TRANCOUNT > 0
62 BEGIN
63     IF XACT_STATE() = 1
64     BEGIN
65         UPDATE t
66         SET t.[Priority] = CASE
67             WHEN DATEDIFF(MINUTE, GETDATE(), i.[Deadline]) / 60 BETWEEN 0 AND 24 THEN
68                 CASE
69                     WHEN i.[Importance] >= 1 AND i.[Importance] < 3 THEN 3
70                     WHEN i.[Importance] >= 3 AND i.[Importance] <= 5 THEN 5
71                 END
72             WHEN DATEDIFF(MINUTE, GETDATE(), i.[Deadline]) / 60 BETWEEN 24 AND 48 THEN
73                 CASE
74                     WHEN i.[Importance] >= 1 AND i.[Importance] < 3 THEN 2
75                     WHEN i.[Importance] >= 3 AND i.[Importance] <= 5 THEN 4
76                 END
77             WHEN DATEDIFF(MINUTE, GETDATE(), i.[Deadline]) / 60 >= 48 THEN
78                 CASE
79                     WHEN i.[Importance] >= 1 AND i.[Importance] < 3 THEN 1
80                     WHEN i.[Importance] >= 3 AND i.[Importance] <= 5 THEN 3
81                 END
82             END
83         FROM Task t
84         INNER JOIN inserted i ON t.Code = i.Code;
85
86         IF @@ERROR <> 0
87         BEGIN
88             ROLLBACK;
89             RETURN;
90         END
91     END
92 ELSE
93     BEGIN
94         RAISERROR('A transação em andamento implícita. O trigger "update_priority" requer uma transação explícita.', 16, 1);
95         RETURN;
96     END
97 END
98 ELSE
99 BEGIN
100     BEGIN TRANSACTION;
101
102     BEGIN TRY
103         UPDATE t
104         SET t.[Priority] = CASE
105             WHEN DATEDIFF(MINUTE, GETDATE(), i.[Deadline]) / 60 BETWEEN 0 AND 24 THEN
106                 CASE
107                     WHEN i.[Importance] >= 1 AND i.[Importance] < 3 THEN 3
108                     WHEN i.[Importance] >= 3 AND i.[Importance] <= 5 THEN 5
109                 END
110             WHEN DATEDIFF(MINUTE, GETDATE(), i.[Deadline]) / 60 BETWEEN 24 AND 48 THEN
111                 CASE
112                     WHEN i.[Importance] >= 1 AND i.[Importance] < 3 THEN 2
113                     WHEN i.[Importance] >= 3 AND i.[Importance] <= 5 THEN 4
114                 END
115             WHEN DATEDIFF(MINUTE, GETDATE(), i.[Deadline]) / 60 >= 48 THEN
116                 CASE
117                     WHEN i.[Importance] >= 1 AND i.[Importance] < 3 THEN 1
118                     WHEN i.[Importance] >= 3 AND i.[Importance] <= 5 THEN 3
119                 END
120             END
121         FROM Task t
122         INNER JOIN inserted i ON t.Code = i.Code;
123
124         COMMIT;
125     END TRY
126     BEGIN CATCH
127         ROLLBACK;

```

```

124         COMMIT;
125     END TRY
126     BEGIN CATCH
127         ROLLBACK;
128         THROW;
129     END CATCH
130 END
131 END
132 GO
133
134 drop trigger if exists encryptPassword
135 go
136
137 create trigger encryptPassword on [User]
138 after insert
139 as
140 begin
141     DECLARE @UserID INT
142     DECLARE @Password CHAR(40)
143
144     DECLARE UserCursor CURSOR FOR
145     SELECT ID, [Password] FROM inserted
146
147     OPEN UserCursor
148     FETCH NEXT FROM UserCursor INTO @UserID, @Password
149
150     WHILE @@FETCH_STATUS = 0
151     BEGIN
152         -- Atualiza a coluna Password com o valor criptografado
153         UPDATE [User]
154         SET [Password] = CONVERT(VARBINARY(8000), EncryptByPassPhrase('ThePassphrase', @Password))
155         WHERE ID = @UserID
156
157         FETCH NEXT FROM UserCursor INTO @UserID, @Password
158     END
159
160     CLOSE UserCursor
161     DEALLOCATE UserCursor
162 end
163 go
164
165 drop trigger if exists createTask
166 go
167
168 create trigger createTask on [Task]
169 instead of insert
170 as
171 begin
172     DECLARE @Title varchar(100),
173             @Description varchar(100),
174             @Importance int,
175             @Deadline datetime,
176             @UserID int;
177
178     -- Definir o cursor
179     DECLARE cursorTasks CURSOR FOR
180     SELECT Title, [Description], Importance, Deadline, [UserID]
181     FROM inserted;
182
183     OPEN cursorTasks;
184
185     FETCH NEXT FROM cursorTasks INTO @Title, @Description, @Importance, @Deadline, @UserID;
186
187     WHILE @@FETCH_STATUS = 0
188     BEGIN
189         INSERT INTO [Task] (Title, [Description], Importance, Deadline, StackID, [UserID])
190         VALUES (@Title, @Description, @Importance, @Deadline, 1, @UserID);
191
192         UPDATE [Stack]
193         SET CurrMaxTasks = CurrMaxTasks + 1
194         WHERE StackID = 1;
195     END

```

```

195
196     FETCH NEXT FROM cursorTasks INTO @Title, @Description, @Importance, @Deadline, @UserID;
197 END;
198
199 CLOSE cursorTasks;
200
201 DEALLOCATE cursorTasks;
202
203 end;
204 GO
205
206
207 DROP TRIGGER IF EXISTS updateUserScore;
208 GO
209
210 CREATE TRIGGER updateUserScore
211 ON [Reward]
212 AFTER INSERT
213 AS
214 BEGIN
215     DECLARE @newReward INT;
216     DECLARE @userid INT;
217
218     SELECT @newReward = [Reward_Value],
219           @userid = Task.UserID
220     FROM inserted
221     INNER JOIN Task ON inserted.Task_code = Task.Code;
222
223     UPDATE [User]
224     SET [Score] = [Score] + @newReward
225     WHERE [User].ID = @userid;
226
227 END;
228 GO
229
230

```

Indexes

Para a otimização de consultas/modificações de entidades da base dados, foram criados **indexes** para as tabelas com parâmetros de **fillfactor** (**configuração que determina a porcentagem de espaço disponível preenchido por dados em uma página de índice**), distintos, cujo os valores variam dependendo da frequência

(esperada) de leitura/modificação de sua determinada tabelas

```
create unique clustered index user_ID_index on [User](ID);

alter index user_ID_index on [User] rebuild with (fillfactor = 100) ;


create unique clustered index stackID_index on [Stack](StackID);
go

alter index stackID_index on [Stack] rebuild with (fillfactor = 70) ;
go


create unique clustered index task_Code_index on [Task](Code);
go

alter index task_Code_index on [Task] rebuild with (fillfactor = 70) ;
go


create unique clustered index Ach_code_index on Achievement(Ach_code);
go
alter index Ach_code_index on [Achievement] rebuild with (fillfactor = 70) ;
go
```

Cursors

Em alguns **triggers** foram implementados alguns **cursors** para iterar sobre linhas inseridas no **insert/updates** de entidades(na sua maioria, **tasks**) e utilizador o valor de suas colunas

```
DROP TRIGGER IF EXISTS updateTask;
GO

CREATE TRIGGER updateTask ON [Task]
INSTEAD OF UPDATE
AS
BEGIN
    DECLARE @Title VARCHAR(100),
            @Description VARCHAR(100),
            @Importance INT,
            @Priority INT,
            @StackID INT,
            @State varchar(10),
            @TaskGroupCode int,
            @Deadline DATETIME;

    DECLARE cursorTasks CURSOR FOR
        SELECT Title, [Description], Importance,[Priority], StackID ,State, TaskGroupCode,Deadline
        FROM inserted;

    OPEN cursorTasks;

    FETCH NEXT FROM cursorTasks INTO @Title, @Description, @Importance, @Priority,@StackID, @State,@TaskGroupCode ,@Deadline;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        UPDATE t
        SET t.Title = ISNULL(i.Title, t.Title),
            t.[Description] = ISNULL(i.[Description], t.[Description]),
            t.Importance = ISNULL(i.Importance, t.Importance),
            t.Priority = ISNULL(i.Priority, t.Priority),
            t.StackID = ISNULL(i.StackID, t.StackID),
            t.State = ISNULL(i.State, t.State),
            t.[TaskGroupCode] = ISNULL(i.[TaskGroupCode], t.[TaskGroupCode]),
            t.Deadline = ISNULL(i.Deadline, t.Deadline)
        FROM [Task] t
        INNER JOIN inserted i ON t.[Code] = i.[Code];

        FETCH NEXT FROM cursorTasks INTO @Title, @Description, @Importance, @Priority,@StackID, @State,@TaskGroupCode ,@Deadline;
    END;

    CLOSE cursorTasks;
    DEALLOCATE cursorTasks;
END;
GO
```

```

create trigger encryptPassword on [User]
after insert
as
begin
    DECLARE @UserID INT
    DECLARE @Password CHAR(40)

    DECLARE UserCursor CURSOR FOR
    SELECT ID, [Password] FROM inserted

    OPEN UserCursor
    FETCH NEXT FROM UserCursor INTO @UserID, @Password

    WHILE @@FETCH_STATUS = 0
    BEGIN
        -- Atualiza a coluna Password com o valor criptografado
        UPDATE [User]
        SET [Password] = CONVERT(VARBINARY(8000), EncryptByPassPhrase('ThePassphrase', @Password))
        WHERE ID = @UserID

        FETCH NEXT FROM UserCursor INTO @UserID, @Password
    END

    CLOSE UserCursor
    DEALLOCATE UserCursor
end
go

```



```

create trigger createTask on [Task]
instead of insert
as
begin
    DECLARE @Title varchar(100),
            @Description varchar(100),
            @Importance int,
            @Deadline datetime,
            @UserID int;

    -- Definir o cursor
    DECLARE cursorTasks CURSOR FOR
    SELECT Title, [Description], Importance, Deadline, [UserID]
    FROM inserted;

    OPEN cursorTasks;

    FETCH NEXT FROM cursorTasks INTO @Title, @Description, @Importance, @Deadline, @UserID;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT INTO [Task] (Title, [Description], Importance, Deadline, StackID ,[UserID])
        VALUES (@Title, @Description, @Importance, @Deadline, 1 ,@UserID);

        UPDATE [Stack]
        SET CurrMaxTasks = CurrMaxTasks + 1
        WHERE StackID = 1;

        FETCH NEXT FROM cursorTasks INTO @Title, @Description, @Importance, @Deadline, @UserID;
    END;

    CLOSE cursorTasks;

    DEALLOCATE cursorTasks;

end;
go

```

Segurança

Login e autenticação : Foi desenvolvido um procedimento responsável por validar/autenticar a entrada de usuário (especificamente os utilizadores da tabela User criada para o projeto) na aplicação e, por consequência, a visualização e manipulação de dados da DB através da **GUI**.

Esta funcionalidade ainda não está perfeitamente alinhada com os **Users** da DB (os quais possuem suas próprias credenciais de acesso), mas em trabalho futuro pretende-se conectar estas duas funcionalidades.

Sql-injection : no que diz respeito a questões de vulnerabilidade, tentamos fazer o melhor para obstruir (possíveis) pontos de vulnerabilidade da aplicação feita no Windows Forms, utilizando dados que são previamente validados para assegurar **insert** e **updates** de valores (os quais também passam por outras validações baseadas no estado da entidade a ser atualizada na BD), assim evitando que possíveis acessos inadequados a BD.

Além disso, vale destacar que boa parte das funcionalidades de leitura ou modificação de entidades da base de dados são realizados(em sua maioria) a partir de **UDFs/SPs/Triggers** **instead of** que possuem suas próprias lógicas de validação dos dados inseridos, de forma que não são realizadas operações sobre a camada de dados com “valores inadequados”.

```

DROP PROCEDURE IF EXISTS checkLogIn;
GO

CREATE PROCEDURE checkLogIn
(
    @email varchar(40),
    @password varchar(40),
    @confirmation int OUTPUT
)
AS
BEGIN
    DECLARE @DecryptedPassword varbinary(8000);
    DECLARE @userid int;

    SELECT @DecryptedPassword = DecryptByPassPhrase('ThePassphrase', [Password])
    FROM [User]
    WHERE [Email] = @email;

    IF CONVERT(varchar(40), @DecryptedPassword) = @password
    BEGIN
        SELECT @userid = [User].ID
        FROM [User]
        WHERE [Email] = @email;

        SET @confirmation = @userid;
    END
    ELSE
    BEGIN
        SET @confirmation = 0;
    END
END;
GO

```

```

private int VerifyLogin(string email, string password)
{
    int confirmation = 0;
    string query = "checkLogIn"; // Nome do procedimento armazenado
    using (SqlCommand cmd = new SqlCommand(query, cn))
    {
        cmd.CommandType = CommandType.StoredProcedure;

        // Parâmetros de entrada
        cmd.Parameters.AddWithValue("@email", email);
        cmd.Parameters.AddWithValue("@password", password);

        // Parâmetro de saída
        SqlParameter confirmationParam = new SqlParameter("@confirmation", SqlDbType.Int);
        confirmationParam.Direction = ParameterDirection.Output;
        cmd.Parameters.Add(confirmationParam);

        cmd.ExecuteNonQuery();

        // Obtém o valor de confirmação do parâmetro de saída
        confirmation = (int)confirmationParam.Value;

        // Retorna true se o login for verificado com sucesso (confirmation = 1)
    }
    return confirmation;
}

```

O **confirmation** é um parâmetro de saída de um **SP** que valida as credenciais de um utilizador inseridas no login da aplicação, caso seja válida a entrada do utilizador, o **confirmation** recebe o **ID** do utilizador, que é utilizado em operações nas outras partes da aplicação.

Conclusão

Ao longo do desenvolvimento deste projeto fizemos o possível para aprender e resolver as várias etapas de desenvolvimento de um sistema de gestão de base de dados que atendem as necessidades de uma empresa/serviço, desde as etapas de planeamento do domínio da base de dados até questões técnicas de segurança e utilização destes serviços conectados à base de dados.

A equipa reconhece que ainda falta aperfeiçoar várias partes do projeto, bem como o desenvolvimento de outras funcionalidades do serviço, principalmente sobre as entidades **task_groups** e **achievements**, cuja as funcionalidades na camada de dados ainda não foram devidamente implementadas, algo que certamente se tornará trabalho futuro.

