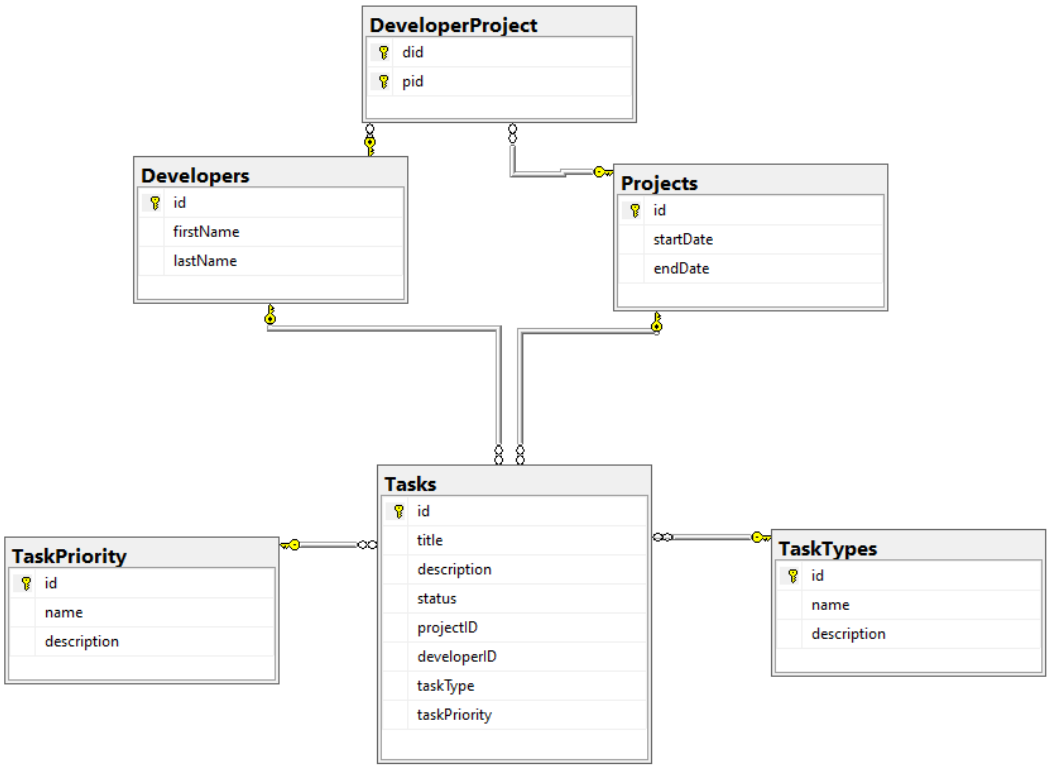

II.a)

```
create table Developers(
	id int primary key identity(1,1),
	firstName varchar(20),
	lastName varchar(20)
)

create table Projects(
	id int primary key identity(1,1),
	startDate datetime,
	endDate datetime
)

create table DeveloperProject(
	did int foreign key references Developers(id),
	pid int foreign key references Projects(id),
	primary key(did, pid)
)

create table TaskTypes(
	id int primary key identity(1,1),
```

```
	name varchar(30),
	description varchar(30)
)

create table TaskPriority(
	id int primary key identity(1,1),
	name varchar(30),
	description varchar(30)
)

create table Tasks(
	id int primary key identity(1,1),
	title varchar(30),
	description varchar(50),
	status varchar(20),
	projectID int foreign key references Projects(id),
	developerID int foreign key references Developers(id),
	taskType int foreign key references TaskTypes(id),
	taskPriority int foreign key references TaskPriority(id)
)

--Out relational data model is in 3NF because we don't have any
--transitive dependencies



II.b)

private SqlConnection sqlConnection;
private DataSet dataSet;
private SqlCommandBuilder commandBuilder;
private SqlDataAdapter daTasks, daTaskTypes;
private BindingSource bsTasks, bsTaskTypes;

sqlConnection = new SqlConnection("Data Source = CRISTI-DESKTOP\\SQLEXPRESS01; Initial
Catalog = SoftwareCompany; Integrated Security = True;");

            bsTasks = new BindingSource();
            bsTaskTypes = new BindingSource();

            //we are linking our dgvs to the data we need them to display by using
binding sources
            dgvTasks.DataSource = bsTasks;
            dgvTaskTypes.DataSource = bsTaskTypes;

            dataSet = new DataSet();

            //data adapters to retrieve the data from the database tables
            daTasks = new SqlDataAdapter("select * from Tasks", sqlConnection);
            daTaskTypes = new SqlDataAdapter("select * from TaskTypes", sqlConnection);

            //commandBuilder only for child table because we need to do
insert/update/delete operations
            commandBuilder = new SqlCommandBuilder(daTasks);

            //populating the dataSet with the tuples from the dataAdapters
            daTasks.Fill(dataSet, "Tasks");
            daTaskTypes.Fill(dataSet, "TaskTypes");
```

```
            //we need to specify the relation between tables(like a 1-to-many relation
from sql)
            DataRelation dataRelation = new DataRelation("Relation",
dataSet.Tables["TaskTypes"].Columns["id"], dataSet.Tables["Tasks"].Columns["taskType"]);
            dataSet.Relations.Add(dataRelation);

            //we give the parent the specific data from the data set
            bsTaskTypes.DataSource = dataSet;
            bsTaskTypes.DataMember = "TaskTypes";

            //we need to give the parent's data source to the child in order to display
the child records of a specific parent
            bsTasks.DataSource = bsTaskTypes;
            bsTasks.DataMember = "Relation";



//sending our changes to the database
            daTasks.Update(dataSet, "Tasks");
```

II.c)

```
BEGIN TRAN
       UPDATE Developers
       SET firstName = 'Ceva'
       WHERE id = 1

COMMIT TRAN



--default isolation level READ COMMITED
BEGIN TRAN
       SELECT * FROM Developers
       waitfor delay '00:00:10'
       SELECT * FROM Developers

COMMIT TRAN

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRAN
       SELECT * FROM Developers
       waitfor delay '00:00:10'
       SELECT * FROM Developers

COMMIT TRAN

--The non-repeatable read occurs when you read two different values from the same row(s)
--during a transaction due to another transaction changing the values from the same row
--while the first transaction still runs
--In order to prevent this issue we need to set the isolation level from the reader
transaction
--to repeatable read, because in this case in order to read a value it will require an
SLock that
```

```
--will only be released at the end of the transaction, opposed to after the operation
completes in
--read commmited. And because it only releases the lock at the end of the transaction,
the second one
--will be forced to wait until the first one completes.
```