



UNIVERSITATEA DIN CRAIOVA  
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI  
ELECTRONICĂ  
DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA  
INFORMAȚIEI



## PROIECT DE DIPLOMĂ

*Marcu Andrei Cristian*

COORDONATOR ȘTIINȚIFIC

*Prof. dr. ing. Liana Stănescu*

*Asist. drd. ing. Petcușin Felix Alin*

*Iulie 2021*

CRAIOVA



UNIVERSITATEA DIN CRAIOVA  
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI  
ELECTRONICĂ  
DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA  
INFORMAȚIEI



*Application for Human Resources Management in a Company*

*Marcu Andrei Cristian*

COORDONATOR ȘTIINȚIFIC

*Prof. dr. ing. Liana Stănescu*

*Asist. drd. ing. Petcușin Felix Alin*

*Iulie 2021*

CRAIOVA

*„Învățătura este o comoară care își urmează stăpânul pretutindeni.”*

Proverb popular

## DECLARAȚIE DE ORIGINALITATE

Subsemnatul Marcu Andrei Cristian, student la specializarea Calculatoare cu predare în limba engleză din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoștință de cele prezentate mai jos și că îmi asum, în acest context, originalitatea proiectului meu de licență:

- cu titlul Application for Human Resources Management in a Company,
- coordonată de Prof. dr. ing. Liana Stănescu, Asist. drd. ing. Inginer Petcușin Felix Alin
- prezentată în sesiunea Iulie 2021.

La elaborarea proiectului de licență, se consideră plagiat una dintre următoarele acțiuni:

- reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele și referința precisă,
- redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
- prezentarea unor date experimentale obținute sau a unor aplicații realizate de alți autori fără menționarea corectă a acestor surse,
- însușirea totală sau parțială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situații neplăcute se recomandă:

- plasarea între ghilimele a citatelor directe și indicarea referinței într-o listă corespunzătoare la sfârșitul lucrării,
- indicarea în text a reformulării unei idei, opinii sau teorii și corespunzător în lista de referințe a sursei originale de la care s-a făcut preluarea,
- precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
- precizarea referințelor poate fi omisă dacă se folosesc informații sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data,

Semnătura candidatului,

20.06.2021








UNIVERSITATEA DIN CRAIOVA  
Facultatea de Automatică, Calculatoare și Electronică  
Departamentul de Calculatoare și Tehnologia Informației

Aprobat la data de .....  
Șef de departament,  
Prof. dr. ing.  
Marius BREZOVAN

## PROIECTUL DE DIPLOMĂ

Numele și prenumele studentului/-ei:	Marcu Andrei Cristian
Enunțul temei:	Application for Human Resources Management in a Company Description: Aplicatia ofera o solutie pentru gestionarea angajatilor dintr-o companie. Oferta statistici si o vedere mai detaliata asupra companiei.
Datele de pornire:	ASP.NET Core, React, React Material UI, EF Core, Devexpress React, SQL Server
Conținutul proiectului:	<p>Introduction: Contains a small overview about the application and the documentation(this subchapter). Each module available in the application is shortly described in here.</p> <p>Technologies: Teoretic part, contains a brief description of each technology used in the application.</p> <p>Project structure: The code structure, with some teoretic details about the design patterns used. Some pieces of code from the application</p> <p>Database structure: Contains the database diagram and a description of each table</p> <p>HRDesk module: Contains screenshots from the application and description for each module. A module is a menu item present in the drawer. This is the most important chapter from the documentation since it contains the most details about the application itself and a lot of screenshots from the website.</p> <p>UML Diagrams: Class Diagram, Activity diagram, Use cases diagram</p> <p>How to use: Details about how to open the project for the first time, like creating and linking the database, installing the packages needed on the frontend.</p> <p>Terms of service: The author of the thesis and terms of use</p> <p>Conclusion: Short recap of the information presented in the other chapters</p> <p>Bibliography: Books that were used in the development of the documentation or the application.</p>

	Web references: Websites that were used in the development of the documentation or the application
Material grafic obligatoriu:	Aplication schema, database structure, UML diagrams, website screenshots
Consultații:	Periodice
Conducătorul științific (titlul, nume și prenume, semnătura):	Prof. dr. ing. Liana Stănescu Asist. drd. ing. Petcușin Felix Alin
Data eliberării temei:	27.11.2020
Termenul estimat de predare a proiectului:	27.06.2021
Data predării proiectului de către student și semnătura acestuia:	30.06.2021 



## REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC

Numele și prenumele candidatului/-ei:

Marcu Andrei Cristian

Specializarea:

Calculatoare cu predare în limba engleză

Titlul proiectului:

Application for Human Resources Management in a Company

Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta):

În facultate ☐

În producție ☐

În cercetare ☐

Altă locație: *[se detaliază]*

În urma analizei lucrării candidatului au fost constatate următoarele:

Nivelul documentării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Tipul proiectului		Cercetare <input type="checkbox"/>	Proiectare <input type="checkbox"/>	Realizare practică <input type="checkbox"/>	Altul <i>[se detaliază]</i>
Aparatul matematic utilizat		Simplu <input type="checkbox"/>	Mediu <input type="checkbox"/>	Complex <input type="checkbox"/>	Absent <input type="checkbox"/>
Utilitate		Contract de cercetare <input type="checkbox"/>	Cercetare internă <input type="checkbox"/>	Utilare <input type="checkbox"/>	Altul <i>[se detaliază]</i>
Redactarea lucrării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Partea grafică, desene		Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
Realizarea practică	Contribuția autorului	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Mare <input type="checkbox"/>	Foarte mare <input type="checkbox"/>
	Complexitatea temei	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>
	Analiza cerințelor	Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
	Arhitectura	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>



	Întocmirea specificațiilor funcționale	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Implementarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Testarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Funcționarea	Da <input type="checkbox"/>	Parțială <input type="checkbox"/>	Nu <input type="checkbox"/>	
Rezultate experimentale		Experiment propriu <input type="checkbox"/>		Preluare din bibliografie <input type="checkbox"/>	
Bibliografie		Cărți	Reviste	Articole	Referințe web
Comentarii și observații					

În concluzie, se propune:

ADMITEREA PROIECTULUI <input type="checkbox"/>	RESPINGEREA PROIECTULUI <input type="checkbox"/>
---------------------------------------------------	-----------------------------------------------------

Data,

Semnătura conducătorului științific,

# PROJECT SUMMARY

The Application for Human Resources Management in a Company is an application designed to simplify the work of the organization on an IT company. Shortly called “*HRDesk*”, the application handles the holiday requests of the employees, offers an overview of all the holidays, has the possibility to book meeting rooms and future plan the organization of the internal calls. The application also generates reports for different types of dates like employees, holiday requests, leave requests, upcoming meetings.

The reason I’ve chose this topic it’s because the application is useful in real world situations and offers multiple ways of expansion. From the point where the application is right now it can be extended to also support polls, salary reports, virtual office management, food menus, and multiple internal company necessities.

The following technologies were used in the development of the website:

Frontend:

- [React](#)
- [React Redux](#)
- [Material UI](#)
- [DevExtreme Reactive](#)

Backend:

- [ASP.NET Core](#)
- [Entity Framework Core](#)

Database:

- [Sql Server](#)

Version control:

- [Github](#)
- [Sourcetree](#)

The application is divided on multiple levels, and each user needs permission in order to access them.

1. Dashboard – Contains general data about the company

2. Leave requests – Possibility to add a leave request(a leave request is a unexpected leave from the work, like going to a doctor, which can be maximum a few hours)
3. Dayoff requests – Possibility to add a holiday request
4. Reports – Charts about the company
5. Meetings – Upcoming meetings of the logged-in user
6. Team – Company employees overview
7. Holiday Calendar – Holiday tracker for the user and it's company(can see his holidays, teams holidays and free national days)
8. Book room – Possibility to create a Meeting which will be assigned to a Team\
9. Manage employees – Add, Edit and Delete employees
10. Manage holidays – Approve or Decline Leave requests, Holidays and create National Days
11. Manage organization – Management data like offices, meeting rooms, teams and functions.

***Termenii cheie:*** website, React, Redux, SQL, ASP.NET Core, Entity Framework, Human resource management, HRDesk, Devextreme Reactive, Material UI

## **ACKNOWLEDGEMENTS**

I would like to thank my supervisor Asist. drd. ing. Petcuşin Felix Alin for the many ideas and suggestion, for great encouragement aswell as giving me the opportunity to conduct this thesis.

I would also like to thank Prof. dr. ing. Liana Stănescu for the prosoposal of the Human Resources Management topic and all the professors who taught me during the four years of university.

Finally, I would like to express my gratitude towards NetRom Software for all the support when it came to university-related needs aswell as the transfer of knowledge.

# TABLE OF CONTENTS

<b>1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1.1	THE OBJECTIVE.....	1
1.2	THE MOTIVATION .....	2
<b>2</b>	<b>TECHNOLOGIES.....</b>	<b>3</b>
2.1	ASP.NET CORE.....	3
2.1.1	<i>What is ASP.NET Core? .....</i>	<i>3</i>
2.1.2	<i>NuGet packages.....</i>	<i>3</i>
2.1.3	<i>Why should you use ASP.NET Core?.....</i>	<i>3</i>
2.1.4	<i>ASP.NET Core vs ASP.NET.....</i>	<i>4</i>
2.2	ENTITY FRAMEWORK CORE .....	4
2.2.1	<i>What is Entity Framework Core? .....</i>	<i>4</i>
2.2.2	<i>LINQ .....</i>	<i>5</i>
2.2.3	<i>Code first approach.....</i>	<i>5</i>
2.3	REACT.....	7
2.3.1	<i>What is React? .....</i>	<i>7</i>
2.3.2	<i>JSX.....</i>	<i>7</i>
2.3.3	<i>React Redux .....</i>	<i>8</i>
2.3.4	<i>Axios.....</i>	<i>10</i>
2.3.5	<i>React Material UI.....</i>	<i>10</i>
2.3.6	<i>Devexpress Scheduler.....</i>	<i>11</i>
2.4	SQL SERVER .....	11
2.5	GITHUB.....	11
2.6	SOURCETREE .....	12
<b>3</b>	<b>PROJECT STRUCTURE.....</b>	<b>13</b>
3.1	HRDESK BACKEND SOLUTION .....	13
3.2	HRDESK FRONTEND SOLUTION .....	13
3.3	REPOSITORY PATTERN .....	14
3.4	UNIT OF WORK.....	15
3.5	SOLID PRINCIPLES .....	15
3.5.1	<i>Single responsibility principle.....</i>	<i>15</i>
3.5.2	<i>Open-closed principle.....</i>	<i>17</i>
3.5.3	<i>Liskov substitution principle.....</i>	<i>18</i>
3.5.4	<i>Interface segregation principle .....</i>	<i>19</i>

3.5.5	<i>Dependency inversion principle</i>	20
3.6	EXCEPTION MIDDLEWARE	21
3.7	DEPENDENCY INJECTION	21
<b>4</b>	<b>DATABASE STRUCTURE</b>	<b>22</b>
4.1	DATABASE SCHEMA	22
4.2	TABLES	22
4.2.1	<i>Users</i>	22
4.2.2	<i>Teams</i>	23
4.2.3	<i>Functions</i>	23
4.2.4	<i>Offices</i>	23
4.2.5	<i>Leave requests</i>	23
4.2.6	<i>Daysoff</i>	23
4.2.7	<i>Company Details</i>	24
4.2.8	<i>Personal Details</i>	24
4.2.9	<i>Meeting Rooms</i>	24
4.2.10	<i>Meetings</i>	24
4.2.11	<i>National days</i>	24
4.2.12	<i>Permissions</i>	24
4.2.13	<i>Polls, PollAnswers, UserPollAnswers</i>	24
4.2.14	<i>UserPermissions</i>	25
4.2.15	<i>HardwareRequests</i>	25
<b>5</b>	<b>HRDESK MODULES</b>	<b>26</b>
5.1	DASHBOARD MODULE	26
5.2	LEAVE REQUEST MODULE	26
5.3	DAYOFF REQUEST MODULE	27
5.4	HARDWARE REQUESTS MODULE	28
5.5	REPORTS MODULE	29
5.6	MEETINGS MODULE	30
5.7	TEAM MODULE	30
5.8	HOLIDAY CALENDAR MODULE	31
5.9	BOOKING MODULE	32
5.10	MANAGE EMPLOYEES MODULE	34
5.11	MANAGE HOLIDAYS MODULE	36
5.12	MANAGE ORGANIZATION	38
5.13	UNAUTHORIZED ACCESS	40

5.14	LOGIN PAGE .....	40
5.15	EXCEL REPORTS .....	41
<b>6</b>	<b>UML DIAGRAMS .....</b>	<b>43</b>
6.1	CLASS DIAGRAM.....	43
6.2	USE CASE DIAGRAM – BASIC USER .....	44
6.3	USE CASE DIAGRAM – ADMINISTRATOR .....	45
6.4	ACTIVITY DIAGRAM – DAY OFF .....	46
<b>7</b>	<b>HOW TO USE .....</b>	<b>47</b>
7.1	FETCH OR DOWNLOAD THE CODE.....	47
7.2	LINK AND CREATE THE DATABASE .....	47
7.3	FRONTEND SOLUTION SETUP.....	48
<b>8</b>	<b>TERMS OF SERVICE .....</b>	<b>50</b>
8.1	AUTHOR.....	50
8.2	TERMS OF USE .....	50
<b>9</b>	<b>CONCLUSION .....</b>	<b>51</b>
<b>10</b>	<b>BIBLIOGRAPHY .....</b>	<b>53</b>
<b>11</b>	<b>WEB REFERENCES .....</b>	<b>54</b>
<b>A.</b>	<b>SOURCE CODE .....</b>	<b>56</b>
<b>B.</b>	<b>WEBSITE .....</b>	<b>57</b>
<b>C.</b>	<b>CD / DVD .....</b>	<b>58</b>
<b>INDEX</b>	<b>.....</b>	<b>59</b>

# LIST OF FIGURES

FIGURE 1. ASP.NET CORE LOGO [1] .....	3
FIGURE 2. LINQ QUERY SYNTAX .....	5
FIGURE 3. LINQ EXTENSION METHODS .....	5
FIGURE 4. CODE FIRST APPROACH[6].....	5
FIGURE 5. HRDESK USER ENTITY EXAMPLE .....	6
FIGURE 6 - REACT LOGO[8] .....	7
FIGURE 7. JSX CODE[9].....	8
FIGURE 8. HOW JSX REALLY LOOKS BEHIND AFTER BEING PARSED BY BABEL[9] .....	8
FIGURE 9. REACT REDUX LOGO[10] .....	8
FIGURE 10. REDUX FLOW FROM "REDUX IN ACTION"[GAR18] .....	9
FIGURE 11. AXIOS WRAPPER FROM HRDESK .....	10
FIGURE 12. GITHUB LOGO[13].....	11
FIGURE 13. HRDESK SOURCETREE HISTORY .....	12
FIGURE 14. HRDESK SOLUTION.....	13
FIGURE 15. HRDESK FRONTEND SOLUTION.....	14
FIGURE 16. UNIT OF WORK PATTERN FLOW .....	15
FIGURE 17. SINGLE RESPONSABILITY PRINCIPLE[16] .....	16
FIGURE 18. OPEN CLOSED PRINCIPLE[16] .....	17
FIGURE 19. LISKOV SUBSTITUTION PRINCIPLE[16] .....	18
FIGURE 20. INTERFACE SEGREGATION PRINCIPLE[16] .....	19
FIGURE 21. DEPENDENCY INVERSION PRINCIPLE[16] .....	20
FIGURE 22. HRDESK DATABASE DIAGRAM.....	22
FIGURE 23. DASHBOARD MODULE .....	26
FIGURE 24. LEAVE REQUEST MODULE .....	27
FIGURE 25. DAYOFF MODULE .....	28
FIGURE 26. HARDWARE REQUESTS MODULE .....	28
FIGURE 27. REPORTS MODULE .....	29
FIGURE 28. MEETINGS MODULE.....	30
FIGURE 29. TEAM MODULE .....	31
FIGURE 30. TEAM MODULE - EMPLOYEE DETAILS.....	31
FIGURE 31. HOLIDAY CALENDAR MODULE .....	32
FIGURE 32. BOOKING MODULE .....	33
FIGURE 33. BOOKING MODULE MENU .....	33
FIGURE 34. MANAGE EMPLOYEES MODULE .....	34
FIGURE 35. MANAGE EMPLOYEES MODULE STEP 1.....	34



FIGURE 36. MANAGE EMPLOYEES MODULE STEP 2.....	35
FIGURE 37. MANAGE EMPLOYEES MODULE STEP 3.....	35
FIGURE 38. MANAGE EMPLOYEES MODULE STEP 4.....	36
FIGURE 39. MANAGE HOLIDAYS MODULE LEAVE REQUESTS.....	37
FIGURE 40. MANAGE HOLIDAYS MODULE NATIONAL DAYS .....	38
FIGURE 41. MANAGE ORGANIZATION MEETING ROOMS AND OFFICES .....	39
FIGURE 42. MANAGE ORGANIZATION HARDWARE REQUESTS .....	39
FIGURE 43. ACCESS DENIED.....	40
FIGURE 44. LOGIN PAGE .....	40
FIGURE 45. EMPLOYEES EXCEL REPORT FROM HRDESK .....	41
FIGURE 46. LEAVE REQUEST EXCEL REPORT FROM HRDESK.....	41
FIGURE 47. NATIONAL DAYS EXCEL REPORT FROM HRDESK .....	42
FIGURE 48. UML CLASS DIAGRAM .....	43
FIGURE 49. UML USE CASE DIAGRAM – USER .....	44
FIGURE 50. UML USE CASE DIAGRAM – ADMINISTRATOR.....	45
FIGURE 51. UML ACTIVITY DIAGRAM - DAY OFF.....	46
FIGURE 52. CONNECTION STRING .....	47
FIGURE 53. APPLY MIGRATION.....	48
FIGURE 54. DEFAULT USER.....	48

# LIST OF TABLES

TABLE 1. ASP.NET CORE VS ASP.NET [4].....	4
-------------------------------------------	---

# 1 INTRODUCTION

## 1.1 The objective

The objective of the Human resource management application in a company called „HRDesk” from now on is to come in help of the companies, granting an alternative of employees management. The application offers different statistics, possibility to export data in Excels and it is designed to be used however the company wants. This was achieved by avoiding creating predefined roles(which would've got access to a certain module) and instead when an user is created it will be assigned to multiple modules.(For each menu item in the left drawer there is a module). This way the users can be splitted on responsibilities(for example there can be a user which accepts/declines the holiday requests of the employees). This user can also be a developer.

Available modules at the time of the presentation:

1. Dashboard – Contains general data about the company and the upcoming meetings of the user
2. Leave requests – Employee/Administrative level – the user can add or delete a leave request. A leave request is an urgent leave, for example going to a doctor
3. Dayoff requests – Employee level – the user can add or delete a day off. Also in here he can keep the track of his number of free days. When we calculate the number of free days we check the overlap with weekends, free national days and another holidays.
4. Hardware requests – Employee level – the user can ask for a permanent/temporary hardware component. The hardware request can be of three types, for the project, for personal use(Ex work from home) and an upgrade.
5. Reports – Employee/Administrative level – Different charts about the company(Age average chart, Country of provenience chart, Functions chart)
6. Meetings – Employee level – Upcoming meetings between two dates
7. Team – Employee level – Company employees ordered by employment date
8. Holiday Calendar – Employee level – Can see his own holidays, team members holidays and also national days
9. Book room – Administrative level – Reserve a meeting room for a team. This meeting will be displayed on the Meetings module.
10. Manage employees – Administrative level – A more detailed overview of the employees. Can add, delete, update them. In here we can assign the user to a module and also we can generate a report(Excel) of all the employees

11. Manage holidays – Administrative level – This module handles the leave and day off requests created before by the employee. The administration can either accept and decline them. Also on this level we have the national days, which are considered as day offs for all employees.
12. Manage organization – Administrative level – Handles the location management. For example offices, meeting rooms(used for booking), teams, functions and hardware requests.

## **1.2 The motivation**

The main reason for choosing this specific topic is because this is a real world usable application that can be extended with many functionalities. Also I am working in a company of this kind and I've seen the benefits an internal application can provide. Keeping track of employees data, like number of free days, salary, and multiple details can simplify the work of the management team of a company.

Also another reason is that the requirement of the topic was to be a Web application, the domain I'm interested in evolving the most. This way I was able to use all the knowledge acquired during the university and create an usable and opened for extension application.

## 2 TECHNOLOGIES

### 2.1 ASP.NET Core

#### 2.1.1 What is ASP.NET Core?

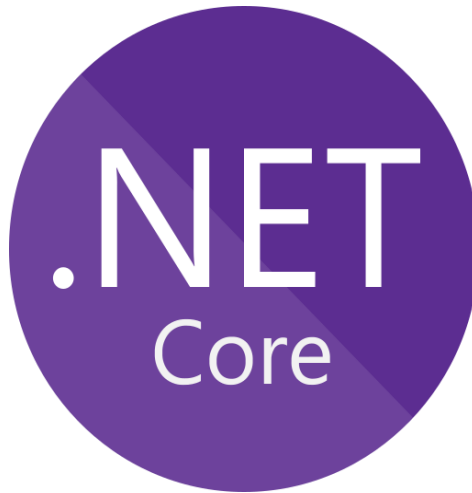


Figure 1. ASP.NET Core logo [1]

ASP.NET Core is the successor of the .NET Framework developed by Microsoft. It is a cross-platform framework targeted to run on the .NET Core platform. It's free, open-source and supports the development of multiple types of applications(Web, Cloud, Mobile etc.).[2]

„ASP.NET Core is a new web framework built with modern software architecture practices and modularization as its focus.” [LOC19]

ASP.NET Core features can be extended by the use of NuGet packages.

#### 2.1.2 NuGet packages

A NuGet package is an essential tool for any modern development platform. It's a mechanism through which developers can create, share and consume useful code. Put simply, a NuGet package is a single ZIP file that contains compiled code. This packages can be private(used by a restrained number of users) or can be public. One example of NuGet package is Entity Framework which will be detailed later on.[3]

#### 2.1.3 Why should you use ASP.NET Core?

- Cross-platform support
- Good performance

- Expandable using NuGet packages
- Cloud support
- Open source
- Can be used for many types of applications(Web, Mobile, Cloud etc)
- Created by Microsoft
- Updated constantly

#### 2.1.4 ASP.NET Core vs ASP.NET

ASP.NET Core	ASP.NET
Supports Windows, Linux and macOS	Supports Windows
Bigger CPU usage and performance	Lower CPU usage and performance
Can be hosted on multiple platforms	Can be hosted on IIS(Internet Information Server)
Supports Docker	Doesn't support Docker
It is focused on Web applications development	It is focused on both Desktop and Web applications

**Table 1. ASP.NET Core vs ASP.NET [4]**

## 2.2 Entity Framework Core

### 2.2.1 What is Entity Framework Core?

Entity Framework Core is an open source and cross platform object-relational mapper. It was initially part of .NET Framework and starting with Entity Framework 6 was delivered separately. This can be acquired from the NuGet packages. [5]

In my application Entity Framework Core was used in order to simplify the relation with the database. Writing LINQ code directly in the main project, without the need of SQL stored procedures makes the code more readable. In my case the database was created using a code first approach with migrations

## 2.2.2 LINQ

Language-Integrated Query(LINQ) is the technology that allow the user to do queries directly from C# code.

Linq syntax is of two types:

- Query syntax. Similar two classic SQL syntax.

```
0 references
public User QuerySyntaxExample(int id)
{
    var users = GetAll();
    return (from user in users
            where user.Id == id
            orderby user.CreatedDate
            select user).SingleOrDefault();
}
```

Figure 2. Linq query syntax

- Extension method syntax

```
0 references
public User ExtensionMethodExample(int id)
{
    var users = GetAll();
    return users.OrderBy(user => user.CreatedDate).FirstOrDefault(user => user.Id == id);
}
```

Figure 3. Linq extension methods

## 2.2.3 Code first approach

In entity framework code first approach the focus is on creating entities from C# code. Based on the Context of the application Entity Framework will create the corresponding tables via migrations. Using Data Annotations on Entity fields you can add limitations like in SQL(for example maximum length of string).

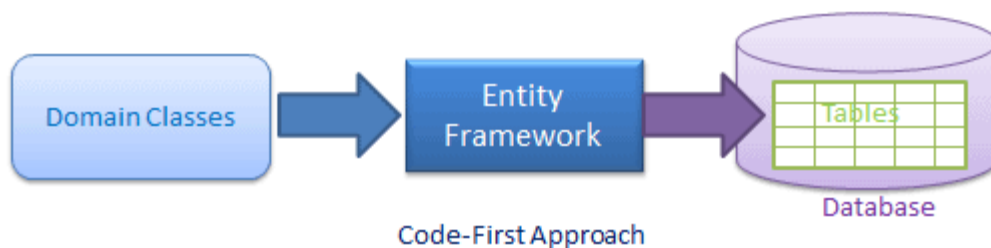


Figure 4. Code first approach[6]

In HRDesk I've used the code first approach. The following picture displays an Entity example from the application. In this case the Entity also inherits a "Base Entity" which is DeleteEntity. This entity automatically adds some extra fields, which are: IsDeleted(used for soft delete), Id(unique identifier), Creation date, Updated date(if it is updated by anyone).

```

29 references
public class User : DeleteEntity<int>
{
    6 references
    public string WorkEmail { get; set; }
    4 references
    public string Password { get; set; }
    2 references
    public bool Predefined { get; set; }
    8 references
    public int? TeamId { get; set; }

    [ForeignKey("TeamId")]
    6 references
    public Team Team { get; set; }

    4 references
    public int? FunctionId { get; set; }

    [ForeignKey("FunctionId")]
    19 references
    public Function Function { get; set; }

    4 references
    public int? OfficeId { get; set; }

    [ForeignKey("OfficeId")]
    6 references
    public Office Office { get; set; }
    2 references
    public ICollection<UserPermission> Permissions { get; set; }
    1 reference
    public int? CompanyDetailsId { get; set; }

    [ForeignKey("CompanyDetailsId")]
    27 references
    public CompanyDetails CompanyDetails { get; set; }
    1 reference
    public int? PersonalDetailsId { get; set; }

    [ForeignKey("PersonalDetailsId")]
    46 references
    public PersonalDetails PersonalDetails { get; set; }
    7 references
    public byte[] ImageSrc { get; set; }
    4 references
    public string ImageType { get; set; }
}

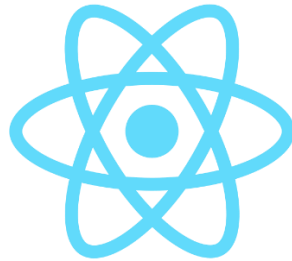
```

Figure 5. HRDesk User entity example



## 2.3 React

### 2.3.1 What is React?



**Figure 6 - React logo[8]**

React is an open-source front-end JavaScript library created by Facebook. The code is structured on UI Components. The main reason of the component is that the code is modular and one component can be used by multiple other components. This way the user is forced to create reusable code. React is mainly focused on state management and displaying the data in the DOM. There are additional libraries that extend it's functionalities. For example react-router-dom for routing, axios for api calls, material-ui for CSS components.[7]

“In contrast, React offers a new approach that streamlines front-end development. React is a powerful UI library that offers an alternative that many big firms such as Facebook, Netflix, and Airbnb have adopted and see as the way forward. Instead of defining a one-off template for your UIs, React allows you to create reusable UI components in JavaScript that you can use again and again in your sites.” [MAR17]

### 2.3.2 JSX

“JSX is an XML/HTML-like syntax used by React so that XML/HTML-like text can co-exist with JavaScript/React code. The syntax is intended to be used by preprocessors (i.e., Babel) to transform HTML-like text found in JavaScript files into standard JavaScript objects that a JavaScript engine will parse.” [9]

Basically, you write HTML-like code which is then parsed by Babel or another processor into Javascript code. The differences between normal HTML and JSX are not noticeable.

```
var nav = (
  <ul id="nav">
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
  </ul>
);
```

Figure 7. JSX Code[9]

```
var nav = React.createElement(
  "ul",
  { id: "nav" },
  React.createElement(
    "li",
    null,
    React.createElement(
      "a",
      { href: "#" },
      "Home"
    )
  ),
  React.createElement(
    "li",
    null,
    React.createElement(
      "a",
      { href: "#" },
      "About"
    )
  )
);
```

Figure 8. How JSX really looks behind after being parsed by Babel[9]

### 2.3.3 React Redux



Figure 9. React Redux logo[10]

In React the state it's placed inside a component and it's available only there. This state can be passed to the Parent Component or the Child component and this causes a limitation. React Redux provides an alternative way to create the state. React Redux creates a global state named store.

Redux is composed of the following components:

- Actions: It's a JavaScript object that contains a type field. The type field describes the action. For example: "getUser".
- Reducer: It's a function that updates the state(if needed) based on the action
- Dispatch: Receives an action as parameter and calls the reducer function of that specific function.
- Store: The actual global state. This can be accessed all over the application using Redux connect.
- Selector: Extracts a field from the state

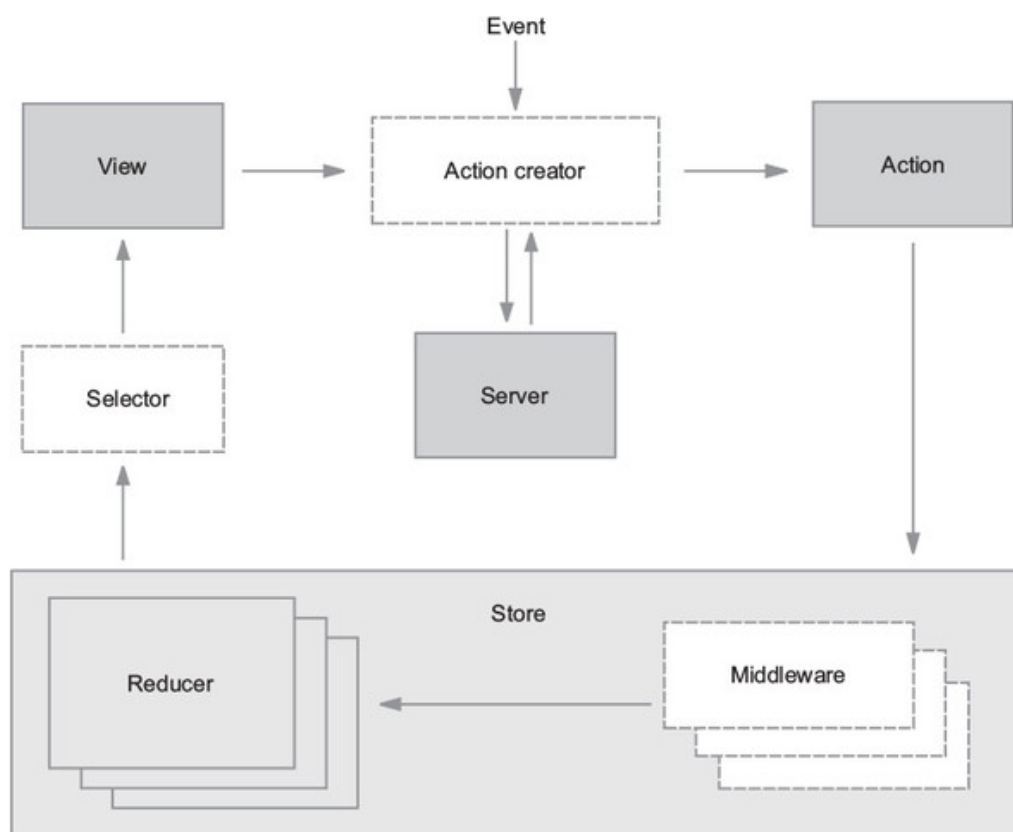


Figure 10. Redux flow from "Redux in action"[GAR18]

### 2.3.4 Axios

Axios is a HTTP client from Node.js. I've created a wrapper using this package for handling api requests. The wrapper automatically creates the request body, attaches the API base url and the authorization token if it's available

```
import axios from "axios";
import { getToken } from "../services/storage";

const axiosWrapper = () => {
  const defaultOptions = {
    baseURL: process.env.REACT_APP_BASE_URL,
    headers: {
      "Content-Type": "application/json",
    },
  };

  let instance = axios.create(defaultOptions);

  instance.interceptors.request.use(function (config) {
    const token = getToken();
    config.headers.Authorization = token ? `Bearer ${token}` : "";
    return config;
  });

  return instance;
};

export default axiosWrapper;
```

Figure 11. Axios wrapper from HRDesk

From this point the wrapper can handle any type of requests, by simply receiving the controller url and the payload.

### 2.3.5 React Material UI

Material UI is an open-source library that provides a multitude of UI components for React application. Material UI follows the guidelines provided by Google on Material Design. It is similar to Bootstrap, it's also based on a 12 column grid structure. Multiple components from Material UI were used in the development of the HRDesk application. The most encountered ones in my application are: Container, Grid, Data Grid, Card, Button, Transfer List, Drawer, Paper, App bar, Avatar and Material Icons. Using the grid system from Material UI caused the application to be as mobile responsive as possible. There are indeed some limitations on data grid with many columns, which can not be resized on a mobile device and another component should be used for this case.

### 2.3.6 Devexpress Scheduler

The Devexpress scheduler was used for the holiday and booking calendar. It offers a similar type of calendar to the one in Outlook. This calendar can be customized in any means by the user since it accepts any component to be overwritten. The scheduler supports recurrent events and for a booking system this was ideal. It's user friendly and it's interface is similar to the one from the Material Design.

## 2.4 SQL Server

SQL Server is a relational database management developed by Microsoft. It is based on the SQL query language. SQL Server can handle database with a huge amount of data. SQL Server has its own SQL derivation, called T-SQL. T-SQL extends the classic SQL queries by adding additional syntax for stored procedures and transactions. It support multiple types of data, from numbers and strings to binary data.[11][12]

SQL Server is also available as a cloud storage on Microsoft Azure. The Azure SQL Database is based on the last version of SQL Server

## 2.5 Github



**Figure 12. GitHub logo[13]**

GitHub is a version control website for software development. It offers multiple services like version control, code management, pull requests, task management, bug tracking. It's free to use, the user can create both public and private repositories. There are some limitations for the free plan, like the number of contributors but that is not an issue for a final exam project. In 2018 GitHub was bought by Microsoft. It the largest host of source code in the world. [14]

While developing HRDesk I've committed after each major change on GitHub, this way whenever an issue occurred I could check the old code, find the cause of the issue easier and or course keep the code safe on the GitHub storage. For my relation with GitHub I've used a Git client available for windows and iOS, called Sourcetree. HRDesk git repository will be found in the how to use chapter.

## 2.6 Sourcetree

Sourcetree is a free to use Git client developed by Atlassian. It is a middleware between the user and a Git. It's intuitive and there are only a few buttons, which makes it very easy to use. Writing the same commands from git bash would be way harder. Also, in case if something is not supported by it's buttons available in the UI, for example hard resetting a merge, or rollbacking a commit it also offers a git bash directly from the UI.

In my opinion the best thing about the Sourcetree is the history view, where you can see all the commits and branches, each branch with it's own color and line

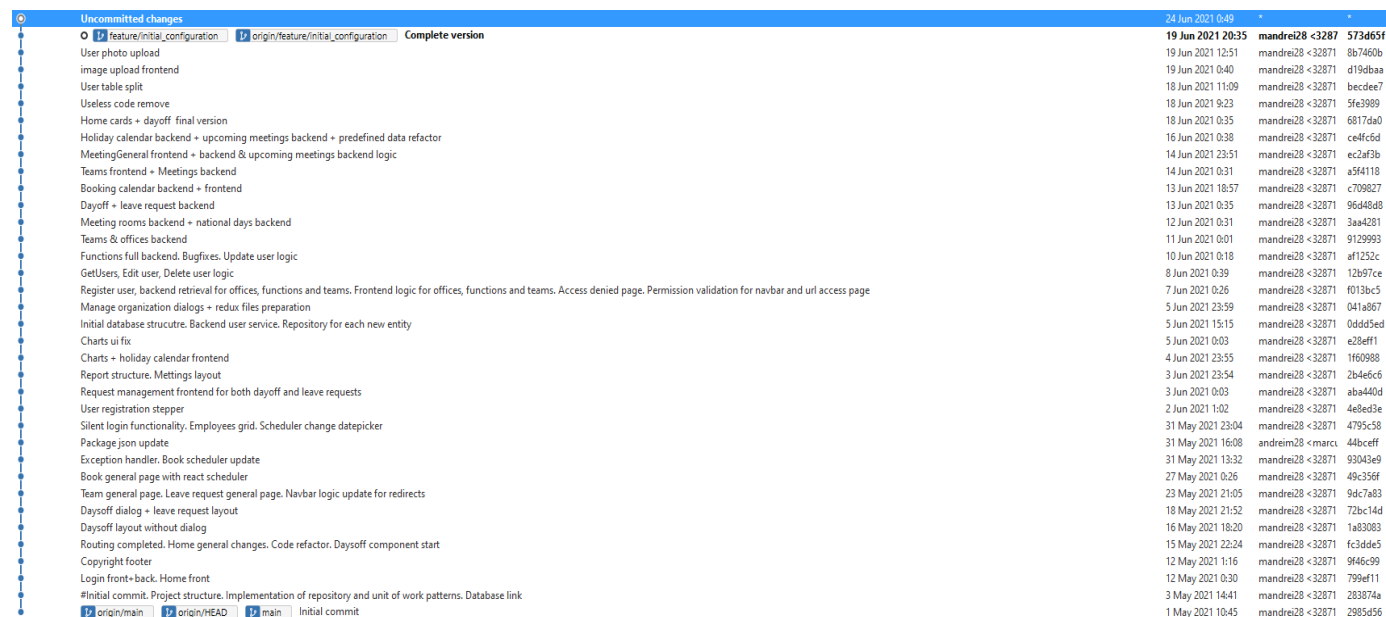


Figure 13. HRDesk sourcetree history

In my case I've used only one branch, so there is only one line. In this view I can see all the changes I've made in that commit and rollback to an exact commit if needed.

## 3 PROJECT STRUCTURE

### 3.1 HRDesk backend solution

HRDesk backend solution is found in the WebApi folder

HRDesk solution is splitted on 3 projects:

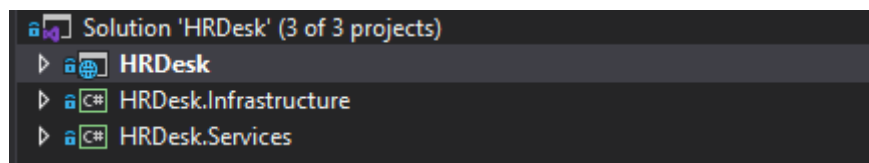


Figure 14. HRDesk solution

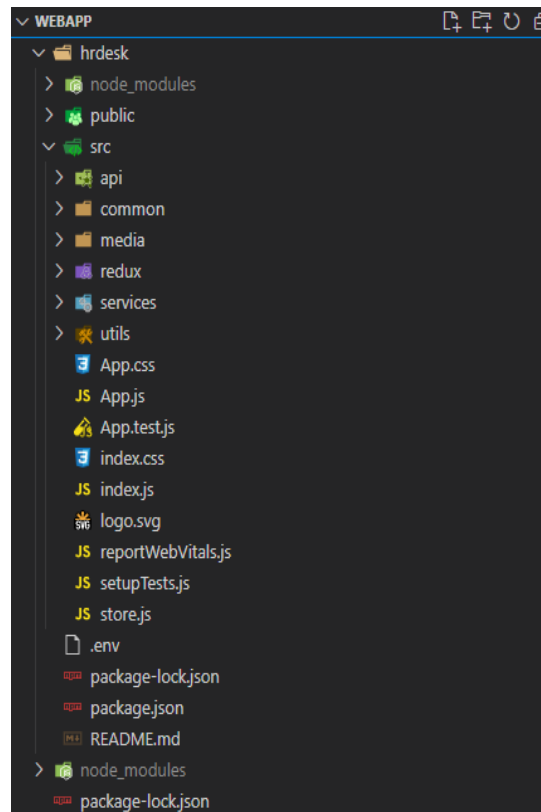
- HRDesk contains the controllers and the Startup. Also there is a wrapper for dependency injection, which is simply called on Startup.
- HRDesk.Infrastructure contains everything needed by the database. In this project we can find the entities, repositories, migrations, DbContext, UnitOfWork.
- HRDesk.Services contains the services of the application. All the logic needed all over the application, mappers, error handlers, exception middlewares.

### 3.2 HRDesk frontend solution

HRDesk frontend solution is found in the WebApp folder

The project is composed of the following structure:

- Api : Contains the Axios wrapper for api calls
- Common: Contains the actual components. A component is composed of a container(which contains the dispatch methods from Redux), general file which contains the actual React code and styles which contains css and material ui styles
- Media: Images, or any file needed in application
- Redux: The actions, reducers and selectors needed by Redux
- Services: Different services that were needed between multiple components, and in order to not duplicate the code this was exported from an external file
- Utils: Constant values used in the application



**Figure 15. HRDesk frontend solution**

### 3.3 Repository pattern

„The repository pattern abstracts the database plumbing code from the implementation,,[GOR20]

Repositories are classes that contain the logic to access the database. Most of the code is made out of Linq code using extensions methods. I've also created a Base Repository which contains CRUD operations. For each entity(table in database) there is one repository. The normal repository extends the CRUD operations provided by the BaseRepository(for example GetUserByEmail, GetUserByName etc.).

The repositories contain one extra level of abstraction, created by the Unit of Work pattern. Using repository pattern will minimize duplicate logic and will decouple the services from the actual database framework. The services are independent of the framework, as long as the logic is kept in the repositories. If the database relation framework (Entity Framework in my case) needs to be changed at any time, the code will be modified only in the repositories.



## 3.4 Unit of Work

An Unit of Work is a class that controls all the repositories, providing an extra layer of abstractisation between the services and the repositories. This way, the user doesn't depend on a framework. For example if entity framework will be replaced in the future the code needs to be changed only in the repositories, leaving the services untouched. This way the application is flexible, maintainable and opened for extension.

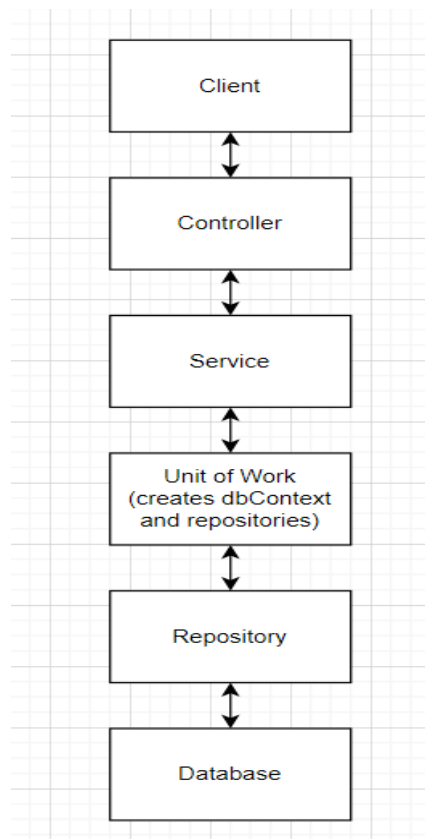


Figure 16. Unit of work pattern flow

## 3.5 SOLID Principles

### 3.5.1 Single responsibility principle

“There should never be more than one reason for a class to change. In other words, every class should have only one responsibility.” [MAR00][15]

The main point of this principle is that a class should have only a responsibility. Let's consider we have two types of persons in our application, Drivers and Programmers. If we respect the Single responsibility principle we will have a class for each type of person, so a class which will handle Drivers and a class which handles Programmers. If we create a class for both, that class will change everytime something is changed in either Driver or Programmer. So if we need to change something about the Programmers we may break the code for Drivers and vice-versa. Also doing unit tests for a single responsibility class is easier than doing them for a multi responsibility class.

"Cohesion is a general concept close to Single Responsibility principle, but the two are closely related. Classes that adhere to the principle tend to have high cohesion and are more maintainable than classes that take on multiple responsibilities and have low cohesion." [FRE20]

How can you determine if a class respects the single responsibility principle? It's simple, the class is changed way more than other classes, and doing unit test for that class has a lot of dependencies that are not exactly related to the functionality you want to test.



Figure 17. Single responsibility principle[16]

### 3.5.2 Open-closed principle

“Software entities should be open for extension, but closed for modification.” [MAR00][15]

Robert C. Martin considered this principle as “the most important principle of object-oriented design”. [MAR00]

The idea behind this principle is that adding a new functionality to an existing class should not change the existent code. So the goal is to extend instead of adapt.

The most explanatory example for this principle that I've heard across the years is the one with the Area. For example we have a function that computes the area of different shapes. The implementation was initially for a Square, then the client required also Rectangles. The code was changed to adapt also Rectangles. Then, after some time the clients wants to also support Circles or other shapes, this time the code will change again.

The solution behind this is to have an interface that creates an abstract method for Area, and then create a class for each shape. In each class we implement the interface.

How can you determine if a class doesn't respect the open closed principle? If you need to change the existent functionality to add something new then the code, most likely, doesn't respect the open closed principle.

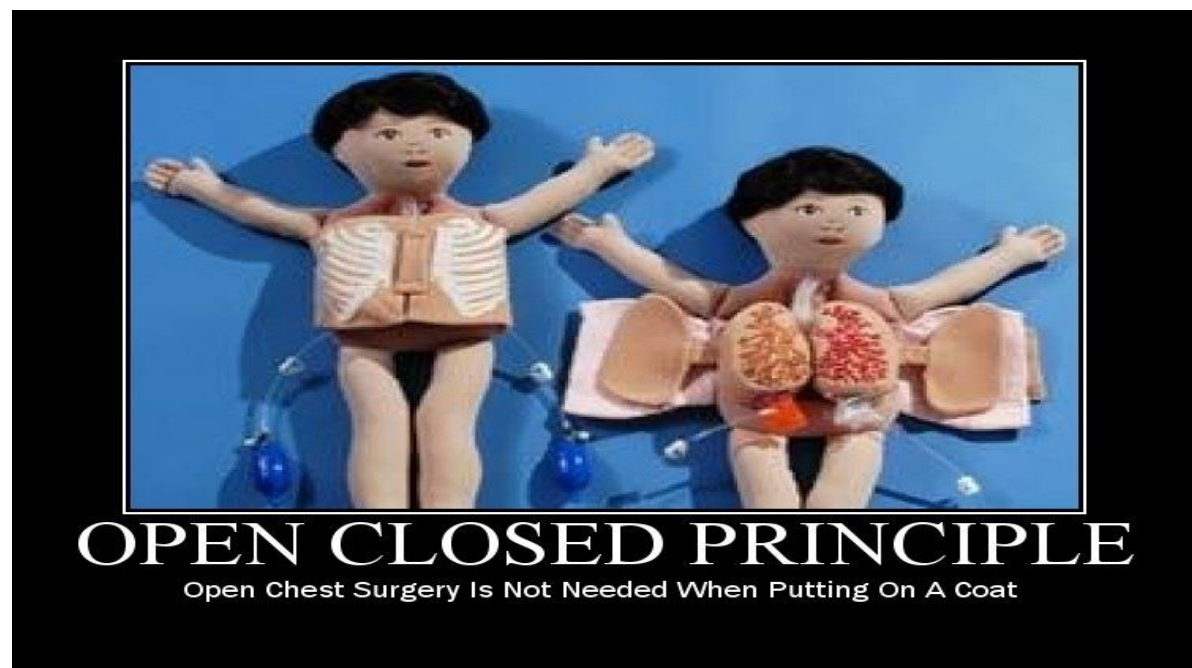


Figure 18. Open Closed principle[16]

### 3.5.3 Liskov substitution principle

“Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it” [MAR00] [15]

This principle states that a superclass should be replaceable with a subclass without breaking the code.

We are again going back to shapes. Let's consider we have two classes, Square and Rectangle. As we know, in mathematics a square is a rectangle, so in our application the Square class will extend the Rectangle. Now the Square class has width and height, which doesn't make sense for a square. [17]

How can we determine if a class doesn't respect Liskov substitution principle? We have situations like “ If object has type Square then do it, else if object has type Rectangle throw not implemented “. In c# the code smells are typeof and switch cases.



Figure 19. Liskov substitution principle[16]

### 3.5.4 Interface segregation principle

“Many client-specific interfaces are better than one general-purpose interface.” [MAR00][15]

The principle states that a client should not be exposed to methods that it doesn't need. In terms of programming a class should and need to implement all the methods from a interface. The principle is broken when let's say two classes implement the same interface but one of them doesn't need one of the methods. That method will be present in the class but will be implemented without any reason.

I've encountered one bad example of this principle a few weeks ago. We had an interface which implemented some specific Errors. This interface was implemented by two classes from two different web projects, because those errors were used when something went wrong in the services from those projects. That interface had errors from both projects, and this caused the classes to have many implementation that weren't used in the project. The solution in our case was to split the common errors in an interface, and create two different interfaces for the ones specific to the project. The new ones extend the common one.

How can we determine if a class doesn't respect Interface segregation principle? In our classes we find methods that are not implemented ( in C# most likely the code smell is throw new NotImplementedException()).

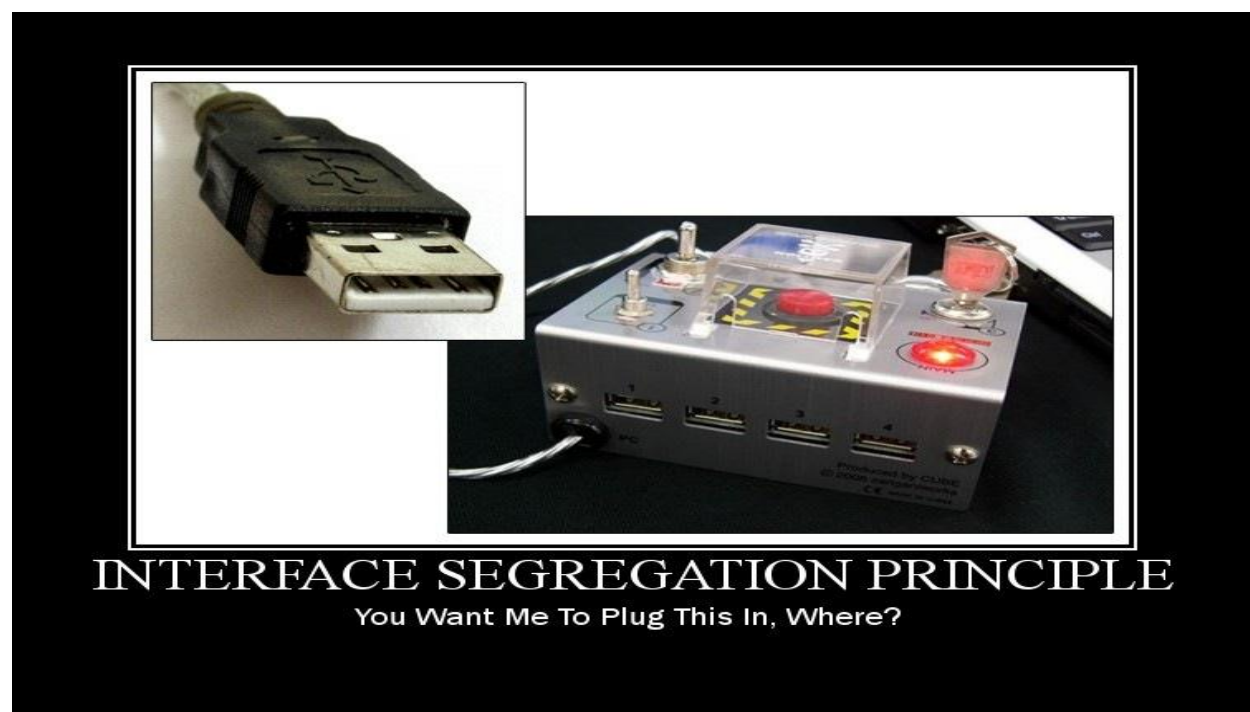


Figure 20. Interface segregation principle[16]

### 3.5.5 Dependency inversion principle

“Depend upon abstractions, [not] concretions.” [MAR00][15]

This principle states that a class should not depend on concrete implementations, only on abstractions(interfaces).[18]

Also an interface(abstraction) should not depend on concrete, concrete should depend on abstraction.[18]

This is accomplished by using Inversion of Control(IoC) or Dependency Injection(DI). ASP.NET Core has dependency injection implemented by default, and this principle is respected by simply adding the interface and the class in the startup file, then the interface can be injected anywhere in the application.

How can we determine if a class doesn't respect Dependency inversion principle? We inject the concrete implementation of a service instead of injecting it's interface.

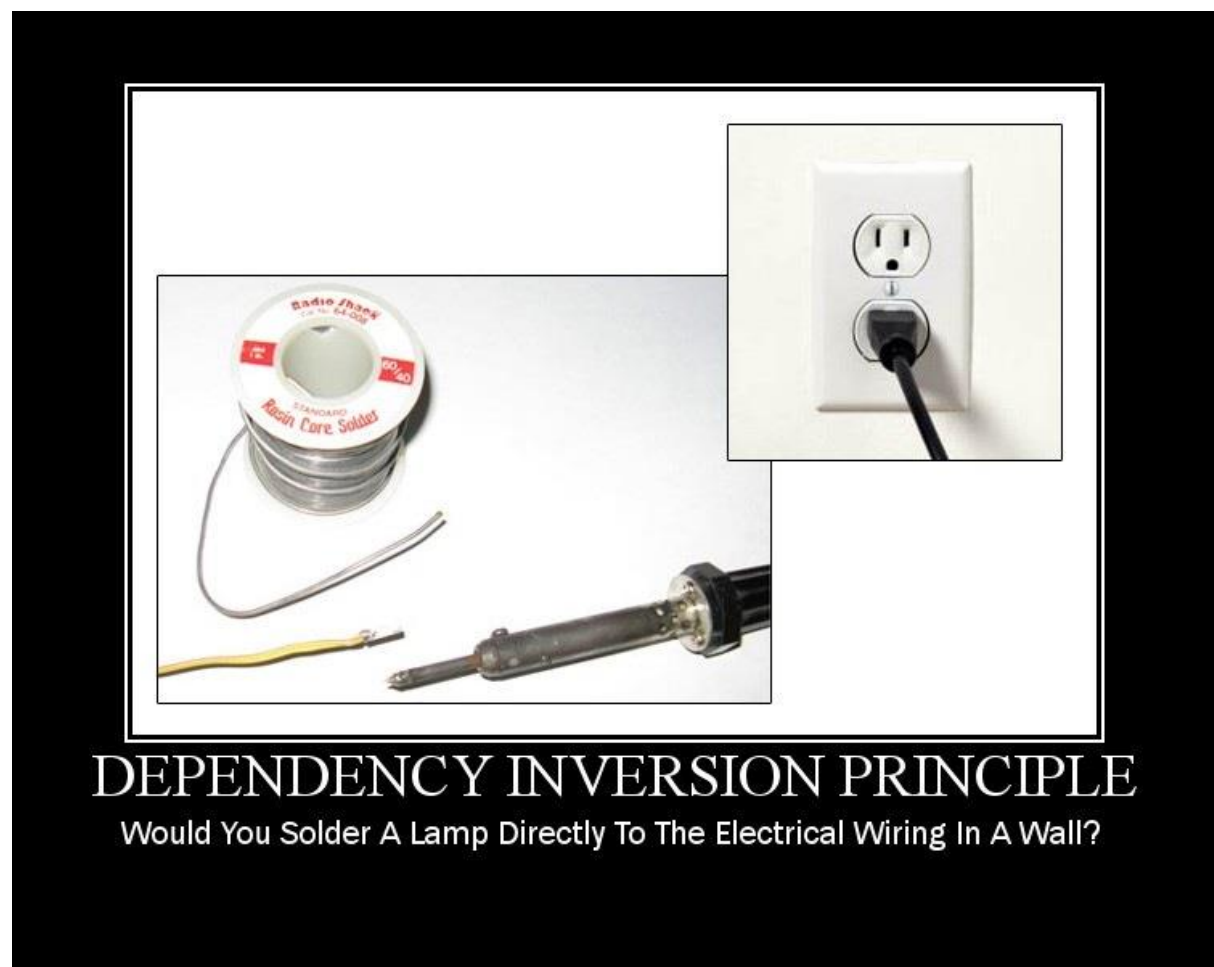


Figure 21. Dependency inversion principle[16]

### 3.6 Exception middleware

The application has an automatic exception handler, whenever an exception is thrown the message is captured by the middleware and it's sent to the frontend. In frontend the code is caught by axios and the error message is automatically displayed in a red toaster. This way the user will know whenever a request fails, and a message will indicate the cause of the issue. One example of exception is thrown when the administrator tries to create a user whose email address is already found in the database. The actual user will receive the error message in toaster "Email already in use".

### 3.7 Dependency Injection

ASP.NET Core supports dependency injection pattern. The classes and their interfaces are registered in the startup of the application. There are three types of lifetimes:

Singleton: Wherever the dependency injection is made the same instance of class will be used for all the calls

Scoped: The instance is the same within the scope of a request, but it's changed when another request is made.

Transient: The instance is always different

Dependency Injection decouples dependencies between services by using abstractions.



## 4 DATABASE STRUCTURE

### 4.1 Database schema

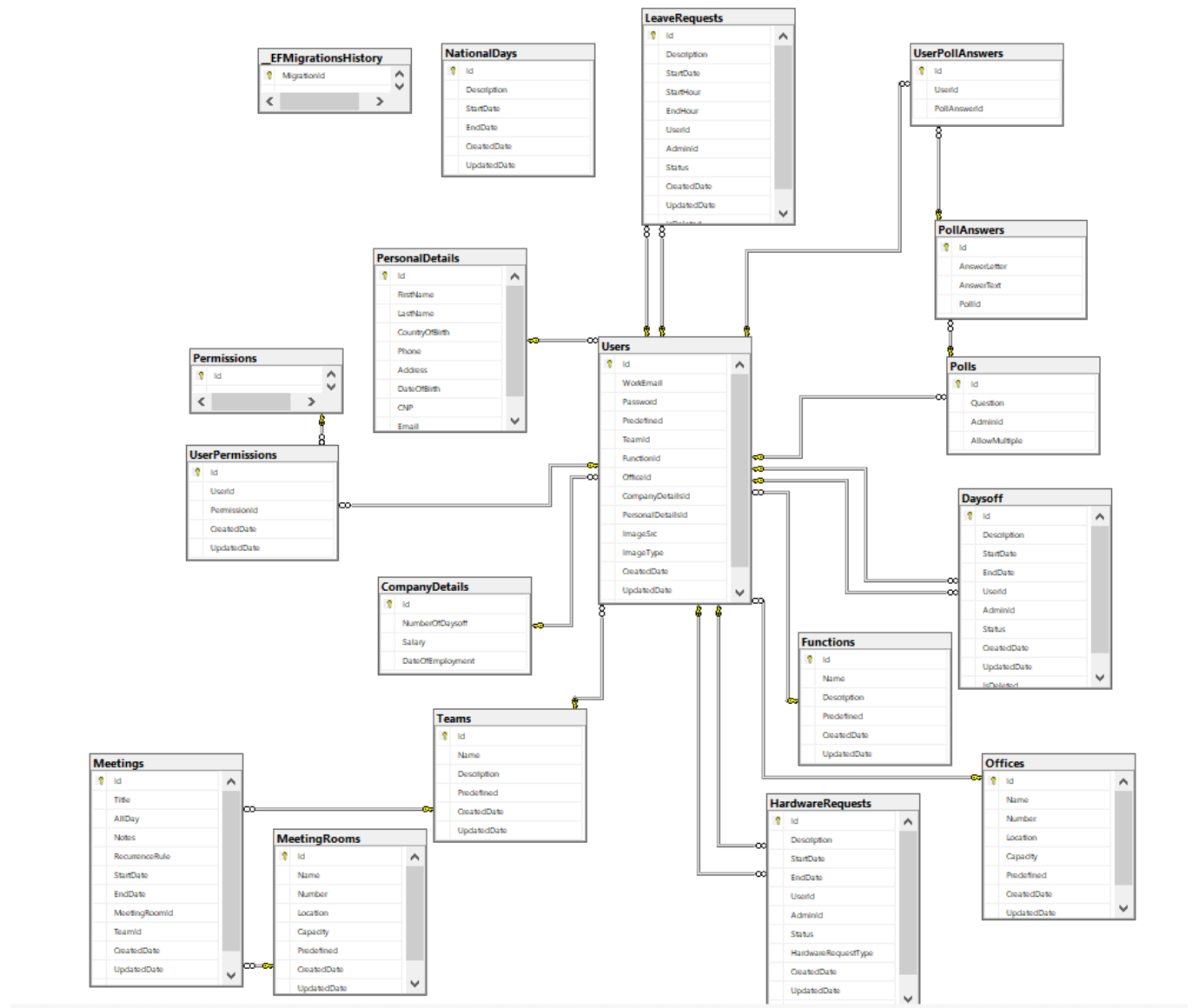


Figure 22. HRDesk database diagram

### 4.2 Tables

#### 4.2.1 Users

Users is the main table of the application, we can call it the core table. This table is related with most of the other tables. It contains all the data about the employee. The details are splitted in two, personal details and company details. An user needs to be linked to a function, table and office in order



to be usable in the application. The account and the password of the user also encoded and stored in this database. The type of the user entity is DeleteEntity, meaning that when an user is marked as deleted but not actually removed from the table.

#### **4.2.2 Teams**

Teams table is used to store the teams available in the companies. An user needs to be assigned to a team, and a team is needed when a booking is made.

#### **4.2.3 Functions**

Functions are stored in this table. An user needs to be linked to a function. The functions are used all over the application. Also there is a chart related to the functions

#### **4.2.4 Offices**

An user needs to be linked to an office. The offices are used for a better organization in the company.

#### **4.2.5 Leave requests**

The leave requests contain two FKs to the user table. One points to the user that made the leave request and the other one points the administrator that has to accept or decline it. The status of a leave request is defined by the following enum:

- Status = 0 -> Waiting
- Status = 1 -> Refused
- Status = 2 -> Approved

The type of this entity is DeleteEntity, meaning that a soft delete is performed instead of an actual remove.

#### **4.2.6 Daysoff**

The daysoff are similar to the leave request, the main difference between these entities is that a dayoff can be made on multiple days, while a leave request is limited to hours in a day. The status enum is the same one from the leave request.

- Status = 0 -> Waiting
- Status = 1 -> Refused
- Status = 2 -> Approved

### **4.2.7 Company Details**

It's an extension of the user table. Contains user's company related data, like salary, employment date, number of free days.

### **4.2.8 Personal Details**

It's an extension of the user table. Contains user's personal data like birth date, name, country of provenience, email, telephone number.

### **4.2.9 Meeting Rooms**

Meeting rooms are similar to offices. The main use of them is by the booking calendar. For each meeting room there will be a column displayed in the calendar.

### **4.2.10 Meetings**

A meeting is related to a Team and a Meeting room. The recurrence rule is a string defined by the devexpress scheduler. This string will tell the scheduler if the meeting is recurrent, or if it's an all day meeting. The meetings are created from the booking calendar and can be seen in both the Booking calendar, home page and meetings module.

### **4.2.11 National days**

National days are added by an administrator. A national day is similar to a dayoff but it's not assigned to an user and doesn't need to be approved. It's a standalone entity that it's reflected to all the employees in the company.

### **4.2.12 Permissions**

In the permissions table we can find the modules. There is no way to add or remove them. The data is inserted by the seed method at the start of the application. The permissions are linked to an user using a many to many relationship, making the user to see the actual module.

### **4.2.13 Polls, PollAnswers, UserPollAnswers**

Those tables are related.

- Polls contains the actual question of the poll and if the poll accepts multiple answers from an user.
- PollAnswers contains the possible answers for that poll, we can have one or many answers for a poll.

- UserPollAnswers is the many to many relationship between the answers and the user. If the poll accepts multiple answers each user will have multiple entries in this table. Also this table support the case when the poll accepts only one entry

#### **4.2.14 UserPermissions**

This table is the many to many table between users and permissions. Based on this data the user gets access to modules

#### **4.2.15 HardwareRequests**

This table stores the data about the hardware requests. There is a one to many relation between the user and the hardware request. A hardware request has the same statuses as a leave request, but also has a HardwareRequestType.

The HardwareRequestType is of three types:

- PersonalUse: The user needs the hardware for personal reasons, or for work from home(example a laptop)
- ProjectUse: Needed for project, can be used by multiple team members(example a tablet)
- Upgrade: Current hardware upgrade(example a new monitor)

## 5 HRDESK MODULES

### 5.1 Dashboard module

The dashboard is a module available to all the user in the application. It's main purpose it's of a home page. Contain details about the number of employees in the company and also an overview about the upcoming meetings.

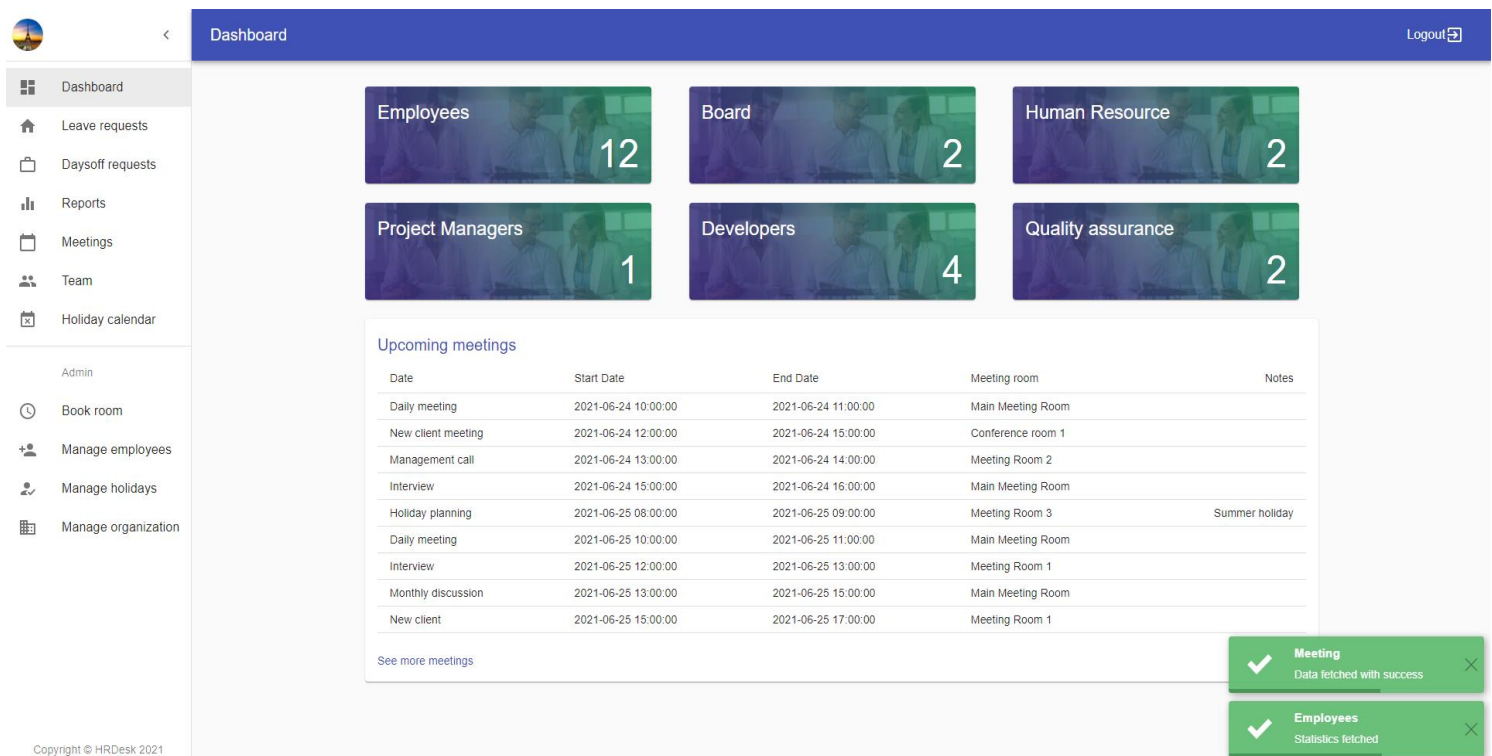


Figure 23. Dashboard module

### 5.2 Leave request module

As mentioned earlier in the presentation a leave request is a temporary necessity to leave the workplace, the most encountered example is going to a doctor. After the user talks with a person which can approve it's leave request he can come to this module and create a leave request. When creating a leave request he needs to specify the interval he will be gone, and also the person which he spoke with. The leave request will be sent to the person he spoke with, and from here the administrator can accept or decline the leave request. This request doesn't impact the number of free days. Let's call them unexpected holidays.

Leave requests

Logout

Short description \* Date \* Start Hour End hour Spoke with \* ADD

ID	Description	Date	Start hour	End hour	Status	Verified By
31	laborator	2021-06-25	22:25:00	22:25:00	Waiting	
24	Laborator	2021-06-28	13:00:00	14:00:00	Waiting	
32	facultate	2021-06-29	14:00:00	16:00:00	Waiting	
33	facultate	2021-06-30	14:00:00	16:00:00	Waiting	
34	work from home	2021-06-30	08:00:00	16:00:00	Waiting	
35	work from home	2021-07-01	08:00:00	16:00:00	Waiting	
36	work from home	2021-07-02	08:00:00	16:00:00	Waiting	
37	work from home	2021-07-05	08:00:00	16:00:00	Waiting	
1	Dentist	2021-06-24	20:00:00	21:00:00	Approved	Marcu Andrei Cristian

1-9 of 9

LeaveRequest Data fetched with success

Figure 24. Leave request module

### 5.3 Dayoff request module

Similar to leave requests, but those are the actual holiday requests of the user. Compared to a leave request a day off can contain multiple days, while a leave request is limited only to hour intervals. Also, when creating a holiday request the user needs to specify the administrator he spoke with, and the administrator will have to accept or decline the dayoff.

If the dayoff is accepted then the number of free days of the user will be automatically adjusted. This number can be noticed on the chart from the top of the page. With blue is the number of free days and with red the number of used days. The number of free days used by a holiday request is calculated without the days that are in weekend, or overlap a national day, or another holiday.

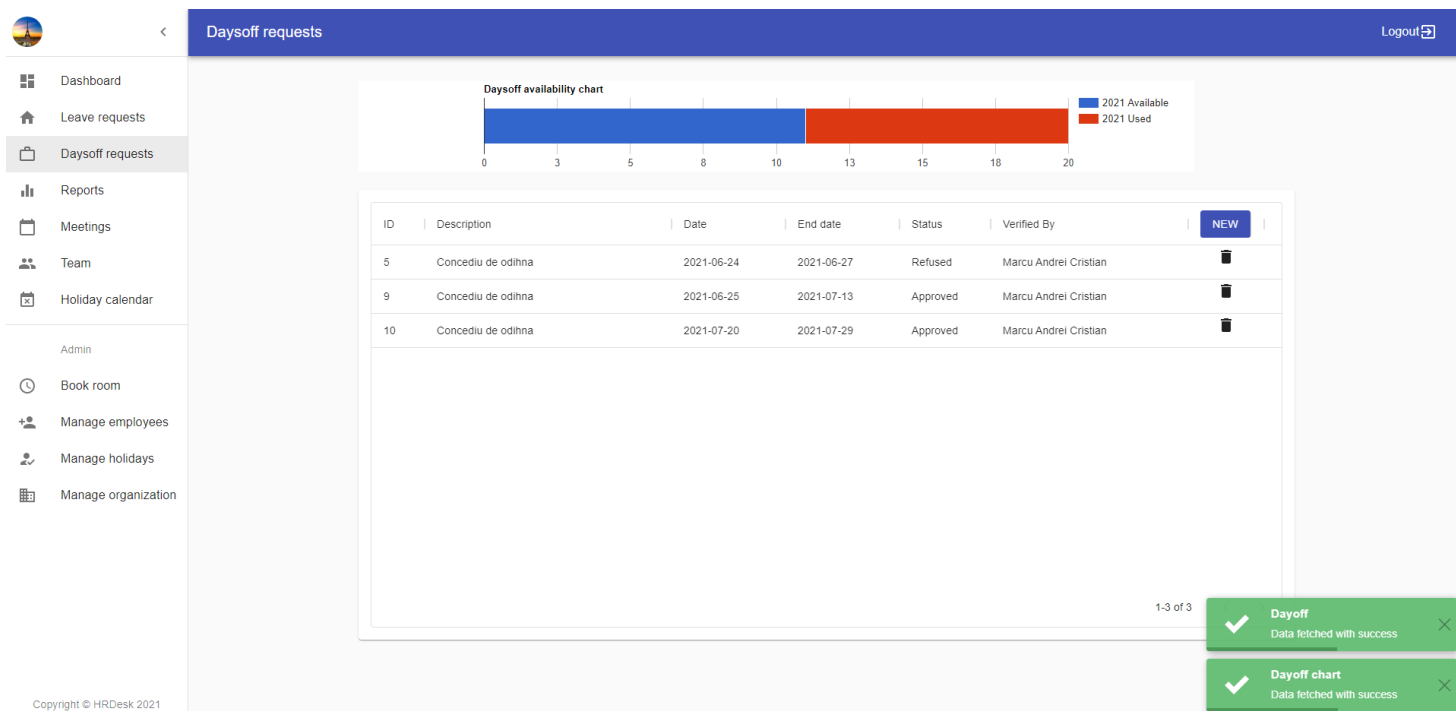


Figure 25. Dayoff module

## 5.4 Hardware requests module

From here the user can request a permanent or temporary upgrade. For example a new monitor or a tablet for temporary testing the application.. These requests need to be approved by an administrator.

Hardware requests

Short description \*

Start date \* 06/27/2021

End date \* 06/27/2021

Request type \*

ADD

ID	Description	Start date	End date	Status	Verified By	Type
1002	IPad	2021-06-27	2021-06-28	Waiting		Project
1003	New mouse	Permanent	Permanent	Approved		Upgrade
1004	Monitor for work from home	Permanent	Permanent	Refused		Personal use

1-3 of 3

Figure 26. Hardware requests module

## 5.5 Reports module

Is composed of three pie charts at the moment of the presentation. On this module there are already multiple tabs available for future use charts.

Right now there are three charts:

- One which displays the number of users between different ranges of age(ex 20-30, 30-40 etc).
- The second one displays the number of users that come from a certain country.
- And the last chart displays the number of users of each function.

Both the functions and the countries of provenience are dynamically retrieve in function of the data inserted, meaning that whenever a function or another country is added this will be automatically available in the charts.

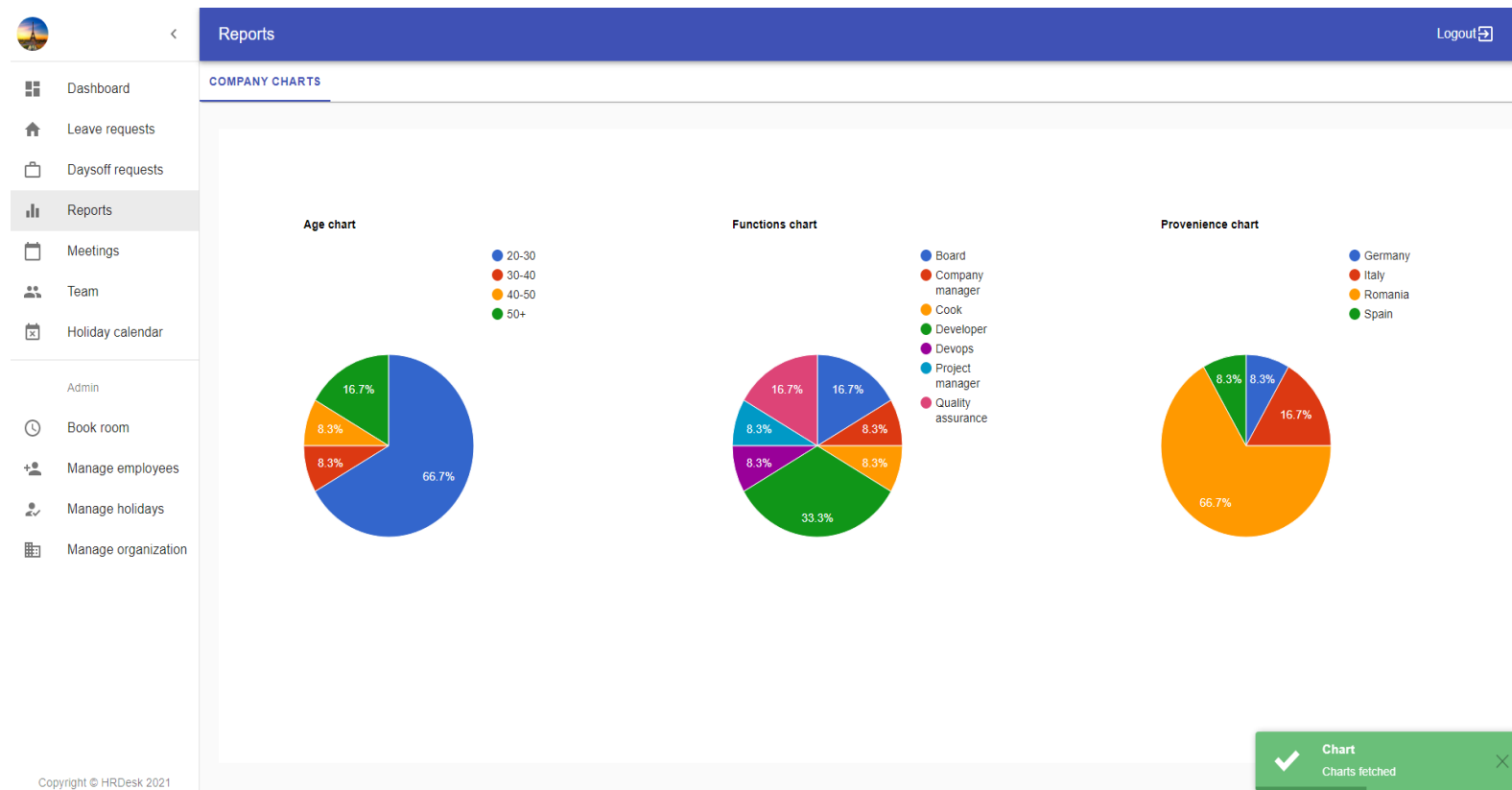
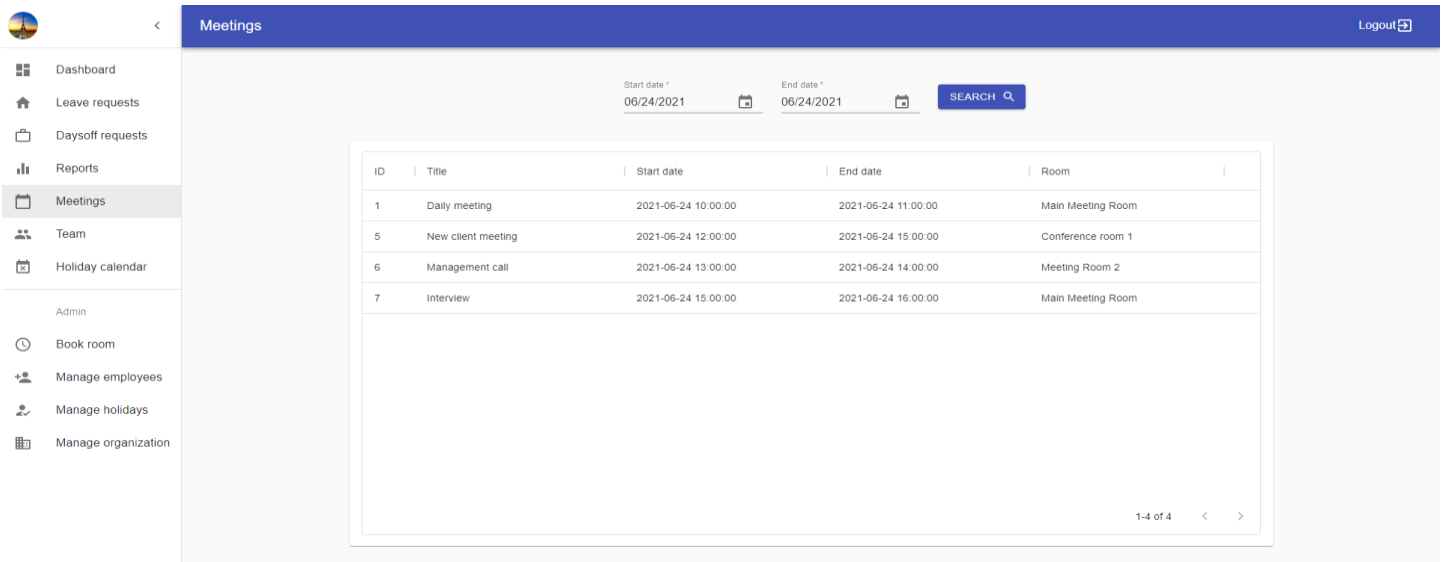


Figure 27. Reports module

## 5.6 Meetings module

It's more of an informative module, the user can see all the meetings between two dates (start and end) which are assigned to it's team. This module is only for display, seeing the meeting room where the call will take place and the description of the call. The meetings will be added from the Booking module which is on the administrative level



ID	Title	Start date	End date	Room
1	Daily meeting	2021-06-24 10:00:00	2021-06-24 11:00:00	Main Meeting Room
5	New client meeting	2021-06-24 12:00:00	2021-06-24 15:00:00	Conference room 1
6	Management call	2021-06-24 13:00:00	2021-06-24 14:00:00	Meeting Room 2
7	Interview	2021-06-24 15:00:00	2021-06-24 16:00:00	Main Meeting Room

Figure 28. Meetings module

## 5.7 Team module

Displays all the employees of the company ordered by the employment date. The entries are displayed in a mobile friendly way using Cards from Material UI. Each card is customized by the data of the user (picture, function, name, etc. ).

Clicking a card opens a more detailed view of the employee, where you can see non-sensitive data like Name, Office, Function, Email, Age, Date of Employment and Avatar.

Both the card and the employee dialog contain the picture of the user. The picture of the user is created by the administrator at the same time with the user, and stored in the database as a byte64 array. Then the picture is cropped by the Material-UI Avatar component.



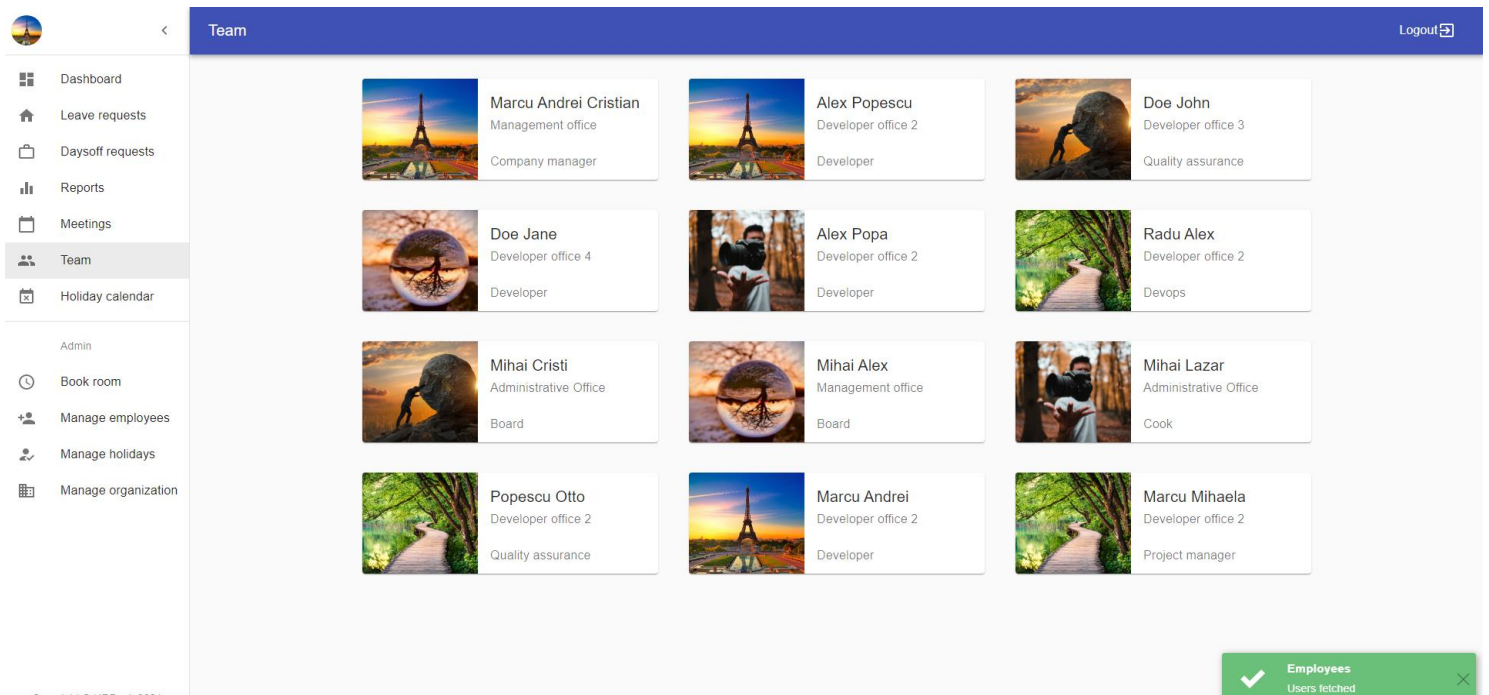


Figure 29. Team module

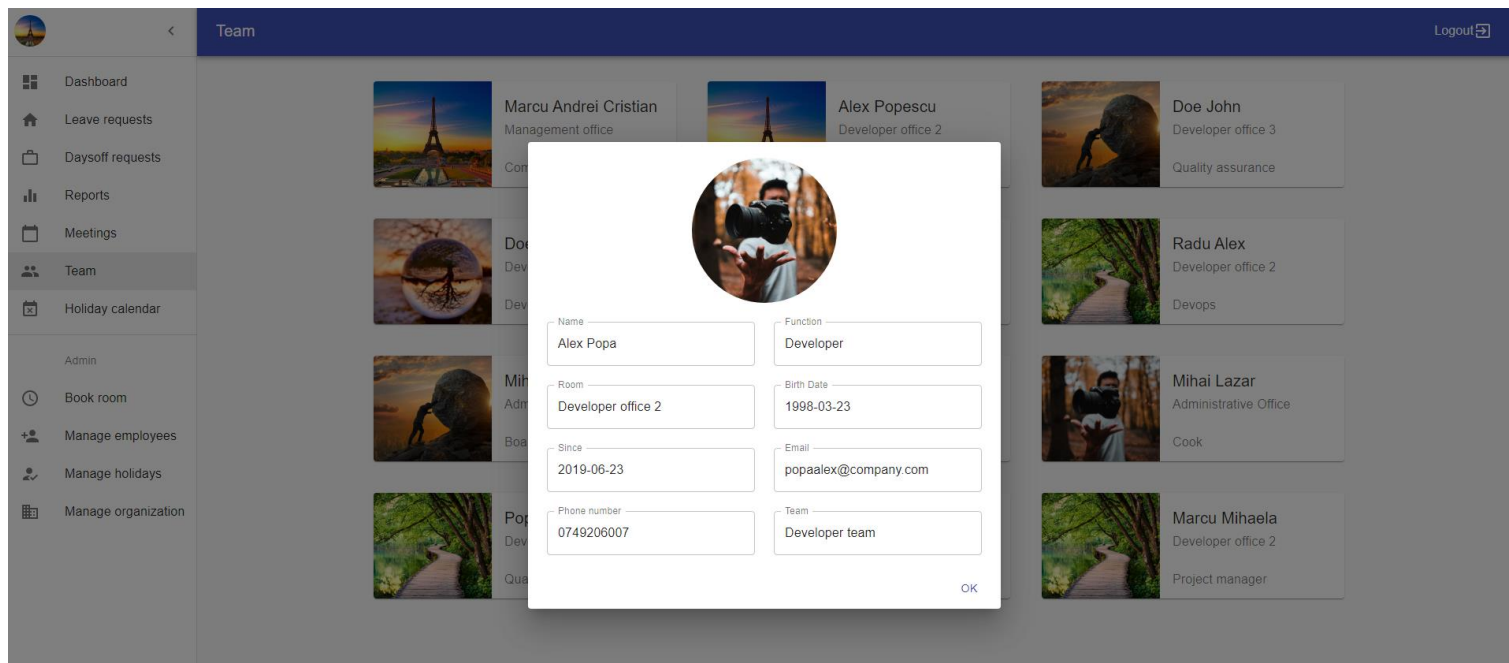


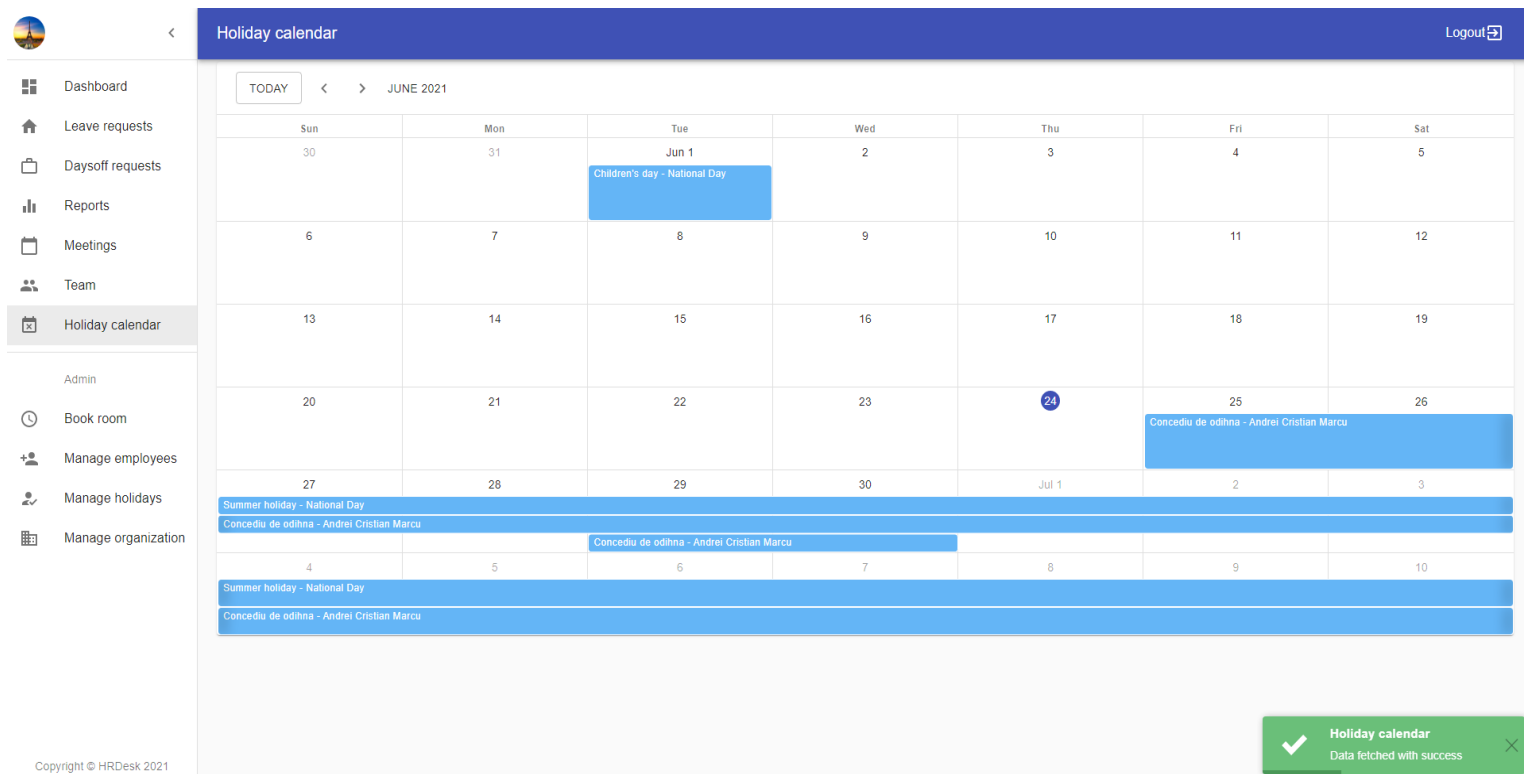
Figure 30. Team module - Employee details

## 5.8 Holiday Calendar module

The module displays a read-only calendar similar to the one available in Microsoft Outlook. This calendar displays three types of entries.

- The first ones are the days off of the current logged in user.

- This user is assigned to a team, so in the calendar the user will also see all the holidays of it's team members.
- In addition to this the national days defined from the administrative level are also displayed. This way the team can make a better development organization, by knowing everytime when a colleague is going to be missing, or when all the team will be gone and the sprint needs to be shorted.



**Figure 31. Holiday calendar module**

## 5.9 Booking module

It's the administrative level of the Module 5(Meetings). It's also a calendar similar to the one in Microsoft Outlook but this one is not read-only. In the calendar we can see all the meeting rooms defined in the application ( the calendar will automatically change when a new meeting room is inserted or deleted).

From here an user which has the permission to access the calendar can book a meeting room for a certain team. This way the planning inside the company will be way easier, there will be no mistaken overlaps in meeting rooms. The application also support future planning and recurrent planning. When a meeting is created for a team, this will be displayed in Module 5 to all team members.

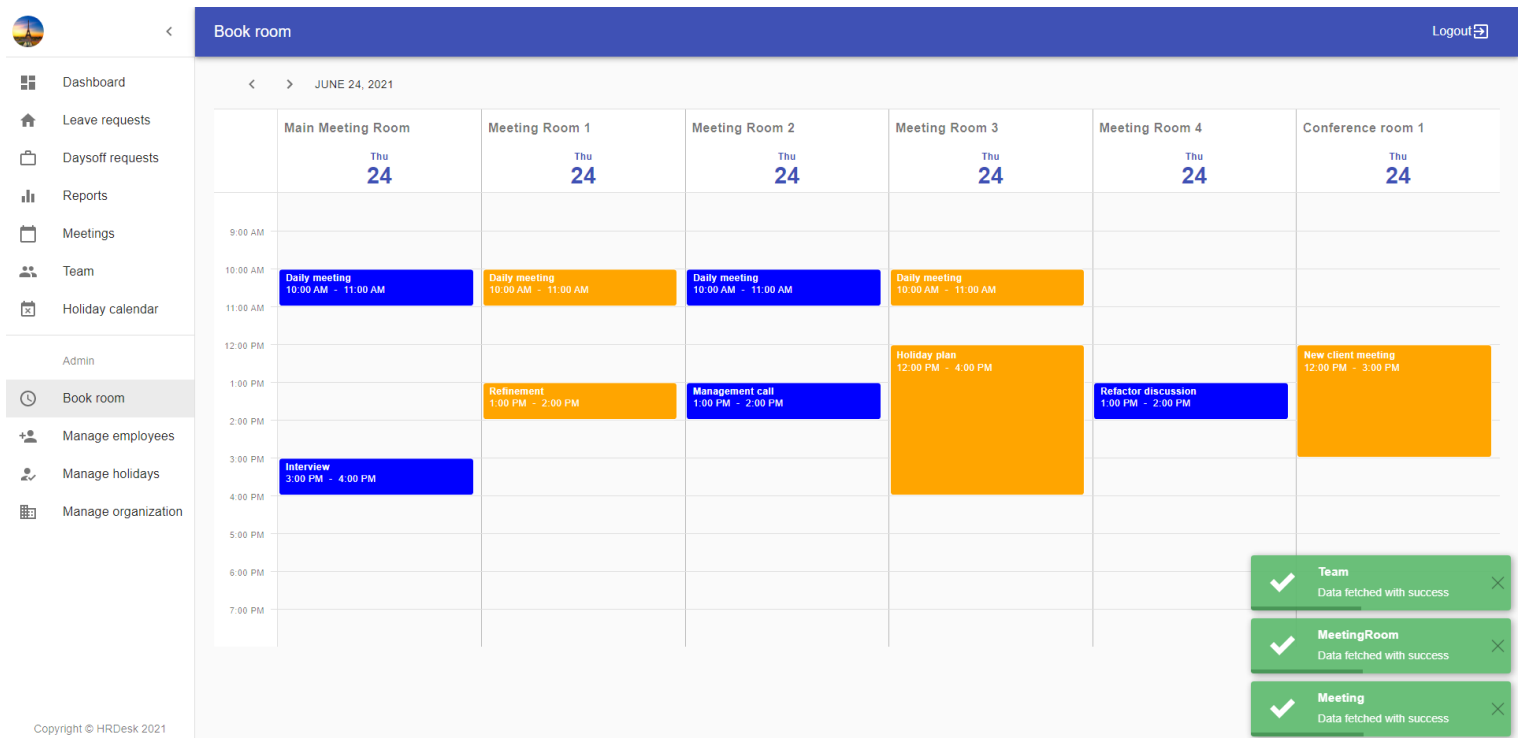


Figure 32. Booking module

If we double click inside a cell from the calendar a menu opens from where we can create a meeting.  
 If we double click a meeting the same menu opens and we can edit or delete the meeting. Also  
 dragging a meeting across the calendar is possible, changing it's hour/meeting room.

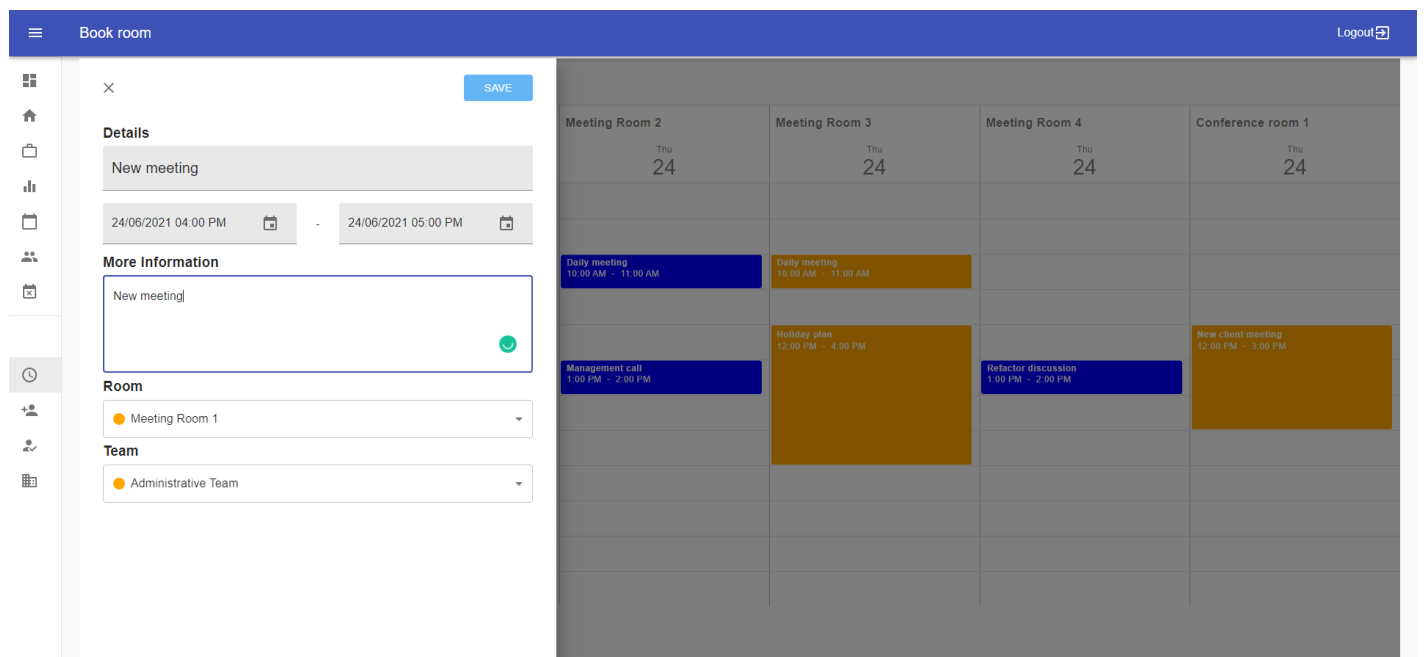
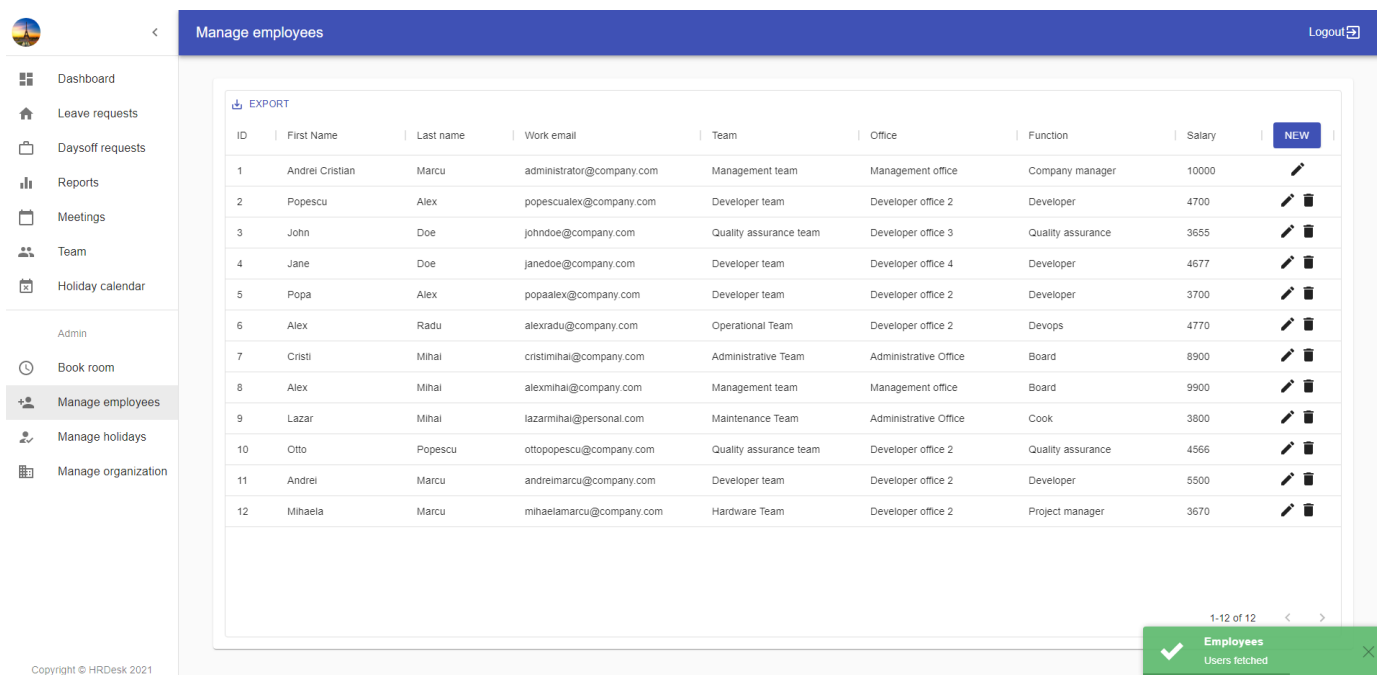


Figure 33. Booking module menu

## 5.10 Manage employees module

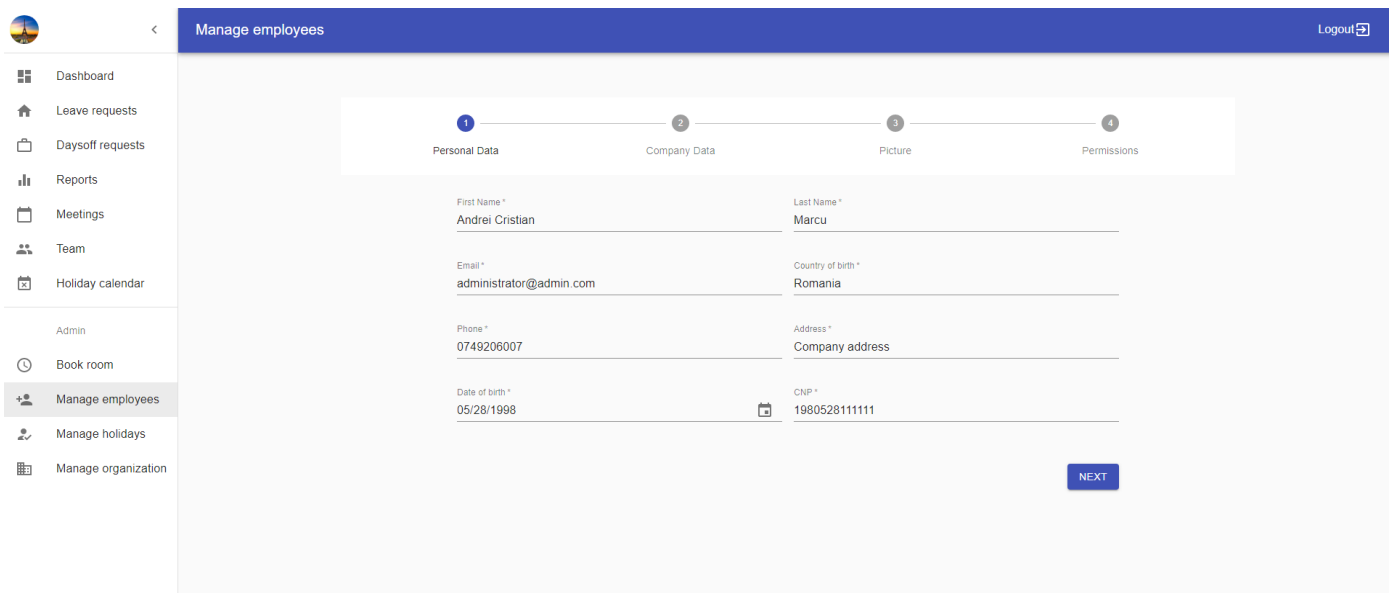
It's the main administrative level, in here all the employees are created, deleted or updated. The creation of an employee is made out of 4 steps.



ID	First Name	Last name	Work email	Team	Office	Function	Salary	NEW
1	Andrei Cristian	Marcu	administrator@company.com	Management team	Management office	Company manager	10000	
2	Popescu	Alex	popescualex@company.com	Developer team	Developer office 2	Developer	4700	
3	John	Doe	johndoe@company.com	Quality assurance team	Developer office 3	Quality assurance	3655	
4	Jane	Doe	janedoe@company.com	Developer team	Developer office 4	Developer	4677	
5	Popa	Alex	popaalex@company.com	Developer team	Developer office 2	Developer	3700	
6	Alex	Radu	alexradu@company.com	Operational Team	Developer office 2	Devops	4770	
7	Cristi	Mihai	cristimihai@company.com	Administrative Team	Administrative Office	Board	8900	
8	Alex	Mihai	alexmihai@company.com	Management team	Management office	Board	9900	
9	Lazar	Mihai	lazarmihai@personal.com	Maintenance Team	Administrative Office	Cook	3800	
10	Otto	Popescu	ottopopescu@company.com	Quality assurance team	Developer office 2	Quality assurance	4566	
11	Andrei	Marcu	andreimarcu@company.com	Developer team	Developer office 2	Developer	5500	
12	Mihaela	Marcu	mihaelamarcu@company.com	Hardware Team	Developer office 2	Project manager	3670	

Figure 34. Manage employees module

The first step is adding personal data like Name, Email, Phone Number, Date of Birth, Country of Provenience.



1

2

3

4

Personal DataCompany DataPicturePermissions

First Name \*  
Andrei Cristian

Last Name \*  
Marcu

Email \*  
administrator@admin.com

Country of birth \*  
Romania

Phone \*  
0749206007

Address \*  
Company address

Date of birth \*  
05/28/1998

CNP \*  
198052811111

NEXT

Figure 35. Manage employees module step 1

The second step contains company specific data like Office, Function, Team, Salary, Number of free days, Work email, Account password.

Manage employees

Logout

Personal Data Company Data Picture Permissions

Team \*  
Management team

Function \*  
Company manager

Office \*  
Management office

Password \*

Number of daysoff \*  
20

Salary \*  
10000

Date of employment \*  
06/05/2021

Work email \*  
administrator@company.com

BACK NEXT

Figure 36. Manage employees module step 2

The third step is adding a picture of the user, after a picture is uploaded the user can see a preview of it.

Manage employees

Logout

Personal Data Company Data Picture Permissions

UPLOAD

BACK NEXT

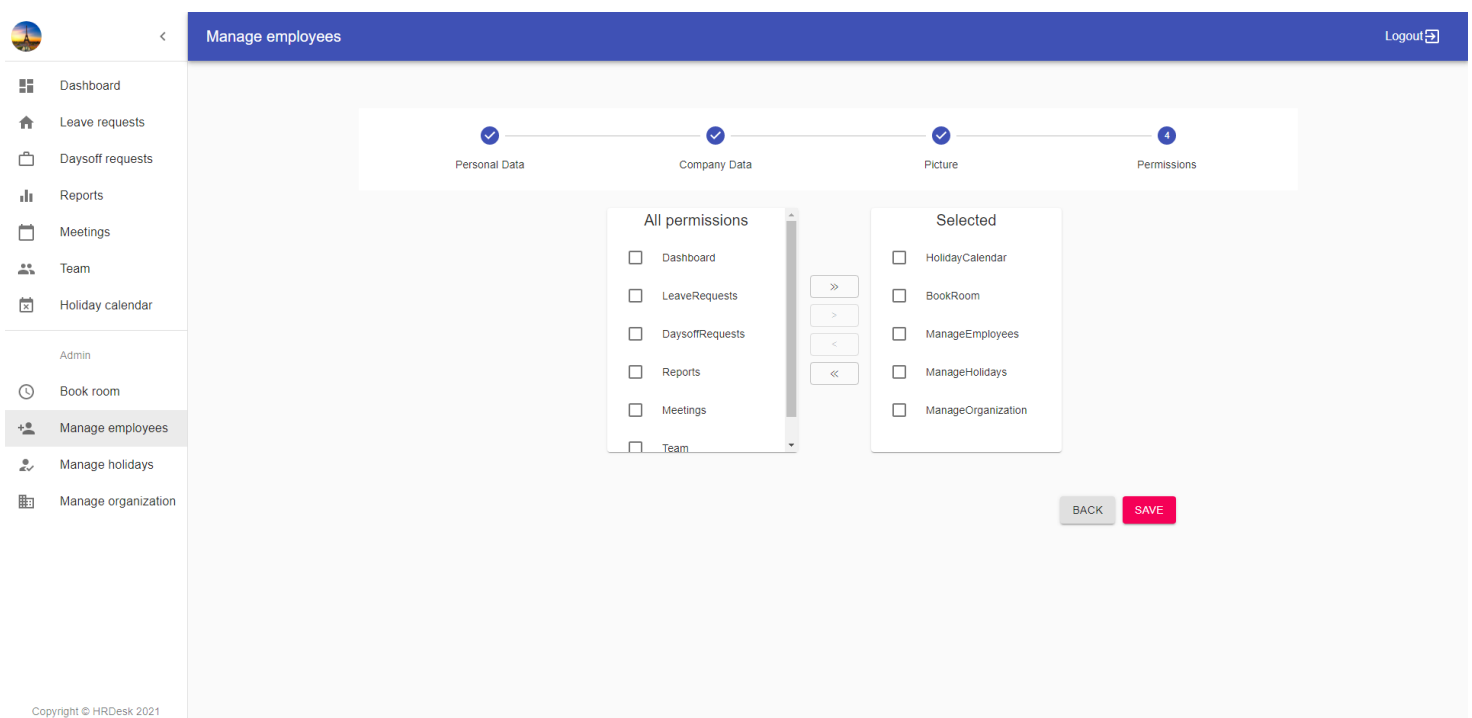
Figure 37. Manage employees module step 3

And the last step, the most important one is the module assigning step. In here the administrator can select which modules of the application the user will be able to see.

If the user is missing a module and he tries to access that page in any way(ex: from url) he will be automatically redirected to the Access Denied page. Also if an user tries to make a request to a module controller that is missing permission he will receive an Unauthorized Http Code.

Between all the steps the data is changed at any time, meaning that if we go back to a step, when we come back the data will always be saved. This way the user can go back and make changes without losing the work.

The modules of an user can be modified at any time by the administrator.



**Figure 38. Manage employees module step 4**

## 5.11 Manage holidays module

It's the administrative module of the requests. It strongly related to the Module 2 and Module 3. In here the person marked as "Administrator" when a leave request or day off is created will see all it's pending request.

There are two button for each request: "Accept" and "Decline". If a leave request is accepted or declined this will be automatically reflected in the status of the request from Module 2 and Module 3.

If a request was accepted and the user changed his mind he can always change it's option. In addition when a day off is accepted the number of free days of the user is automatically calculated.

There are 3 statuses for leave requests and days off:

- Status = 0 -> Waiting
- Status = 1 -> Refused
- Status = 2 -> Approved

The screenshot shows the 'Manage holidays' module interface. On the left is a sidebar with navigation links: Dashboard, Leave requests, Daysoff requests, Reports, Meetings, Team, Holiday calendar, Admin, Book room, Manage employees, Manage holidays (selected), and Manage organization. The main content area has a blue header with 'Manage holidays' and a 'Logout' button. Below the header are three tabs: 'LEAVE REQUESTS' (active), 'DAYOFF REQUESTS', and 'NATIONAL DAYS'. An 'EXPORT' button is located above the table. The table contains 10 rows of leave requests with columns: ID, User, Date, Start hour, End hour, Status, and Description. Each row has 'ACCEPT' and 'DECLINE' buttons. The status of the requests varies: Waiting, Refused, and Approved.

ID	User	Date	Start hour	End hour	Status	Description	Buttons
6	Doe John	2021-06-21	08:30:00	17:00:00	Waiting	Work from home	ACCEPT DECLINE
7	Doe John	2021-06-22	08:30:00	17:00:00	Refused	Work from home	ACCEPT
12	Doe John	2021-06-29	08:30:00	17:00:00	Refused	Work from home	ACCEPT
13	Mihai Alex	2021-06-24	08:30:00	17:00:00	Refused	Work from home	ACCEPT
14	Mihai Alex	2021-06-25	08:30:00	17:00:00	Approved	Work from home	DECLINE
15	Mihai Alex	2021-06-23	08:30:00	17:00:00	Approved	Work from home	DECLINE
16	Mihai Alex	2021-06-22	08:30:00	17:00:00	Approved	Work from home	DECLINE
17	Mihai Alex	2021-06-21	08:30:00	17:00:00	Approved	Work from home	DECLINE
24	Marcu Andrei Cristian	2021-06-28	13:00:00	14:00:00	Approved	Laborator	DECLINE

1-9 of 19

**Figure 39. Manage holidays module leave requests**

There is one extra tab on this module, the National Days tab, which defines the free days across all the company employees. This national days are displayed in each user's holiday calendar. All the grids from this levels offer the user the possibility to generate a report of all the data. The report is generated as excel and contains the data visible in the grid.

The screenshot shows the 'Manage holidays' module with the 'NATIONAL DAYS' tab selected. The table lists the following data:

ID	Description	Start date	End date	CreationDate	
1	Easter holiday	2021-05-02	2021-05-04	2021-06-24	[Edit] [Delete]
2	Christmas holiday	2021-12-22	2022-01-04	2021-06-24	[Edit] [Delete]
3	Children's day	2021-06-01	2021-06-02	2021-06-24	[Edit] [Delete]
4	Summer holiday	2021-06-27	2021-07-14	2021-06-24	[Edit] [Delete]

A green notification banner at the bottom right indicates: 'NationalDay Data fetched with success'.

**Figure 40. Manage holidays module national days**

## 5.12 Manage organization

It's also an important administrative module. In here the administrator can create, update or delete it's own company details. There are three tabs:

- Teams where all the teams are defined(the teams are used all over the application, for example when creating a meeting or an user).
- Functions tab is also needed when creating an user. Whenever we add a Function the Functions chart from Module 4 is automatically updated
- Rooms tab contains the information about offices and meeting rooms

For all these tabs there are some predefined entries created by the DbContext onModelCreating seed method, which is called the first time when a migration is ran on the database. There is also an user with all the modules enabled created on startup. The credentials for this user will be provided later in the presentation.



Manage organization

Logout

Dashboard

Leave requests

Daysoff requests

Reports

Meetings

Team

Holiday calendar

Admin

Book room

Manage employees

Manage holidays

Manage organization

TEAMS

FUNCTIONS

MEETING ROOMS AND OFFICES

ID	Meeting Room Name	Capacity	Location	Number	Creation date	NEW
1	Main Meeting Room	5	Floor 1	1	2021-06-05	
2	Meeting Room 1	10	Floor 1	2	2021-06-24	
3	Meeting Room 2	4	Floor 1	3	2021-06-24	
4	Meeting Room 3	5	Floor 1	4	2021-06-24	
5	Meeting Room 4	6	Floor 2	5	2021-06-24	
6	Conference room 1	30	Floor 2	6	2021-06-24	

1-6 of 6

ID	Office Name	Capacity	Location	Number	Creation date	NEW
1	Management office	0	First floor	1	2021-06-05	
2	Administrative Office	8	Floor 1	2	2021-06-24	
3	Developer office 1	25	Floor 1	3	2021-06-24	
4	Developer office 2	25	Floor 2	4	2021-06-24	
5	Developer office 3	40	Floor 3	5	2021-06-24	
6	Developer office 4	25	Main building	6	2021-06-24	

1-6 of 6

Office

Data fetched with success

MeetingRoom

Data fetched with success

Copyright © HRDesk 2021

**Figure 41. Manage organization meeting rooms and offices**

From the hardware requests tab we can accept the requests made from the hardware request module

Manage organization

Logout

TEAMS

FUNCTIONS

MEETING ROOMS AND OFFICES

HARDWARE REQUESTS

EXPORT

ID	User	Start date	End date	Status	Description	Type	ACCEPT	DECLINE
1002	Marcu Andrei Cristian	2021-06-27	2021-06-28	Waiting	IPad	Project	ACCEPT	DECLINE
1003	Marcu Andrei Cristian	Permanent	Permanent	Approved	New mouse	Upgrade		DECLINE
1004	Marcu Andrei Cristian	Permanent	Permanent	Refused	Monitor for work from h...	Personal use	ACCEPT	
1005	Mihai Radu	Permanent	Permanent	Approved	Laptop WFH	Personal use		DECLINE
1006	Mihai Radu	Permanent	Permanent	Approved	Webcam	Personal use		DECLINE
1007	Mihai Radu	2021-06-29	2021-07-02	Approved	Samsung Tablet	Project		DECLINE
1008	Mihai Radu	Permanent	Permanent	Approved	New monitor	Upgrade		DECLINE

1-7 of 7

HardwareRequest

Data fetched with success

**Figure 42. Manage organization hardware requests**

## 5.13 Unauthorized access

The routing is strongly protected by the permissions, meaning that if the user tries to access a module which it's missing from it's own attributions he will always end on the Access Denied page. The only way to end on a missing permission page is to try to access the page from the URL. Otherwise the menu items won't be available in the drawer.

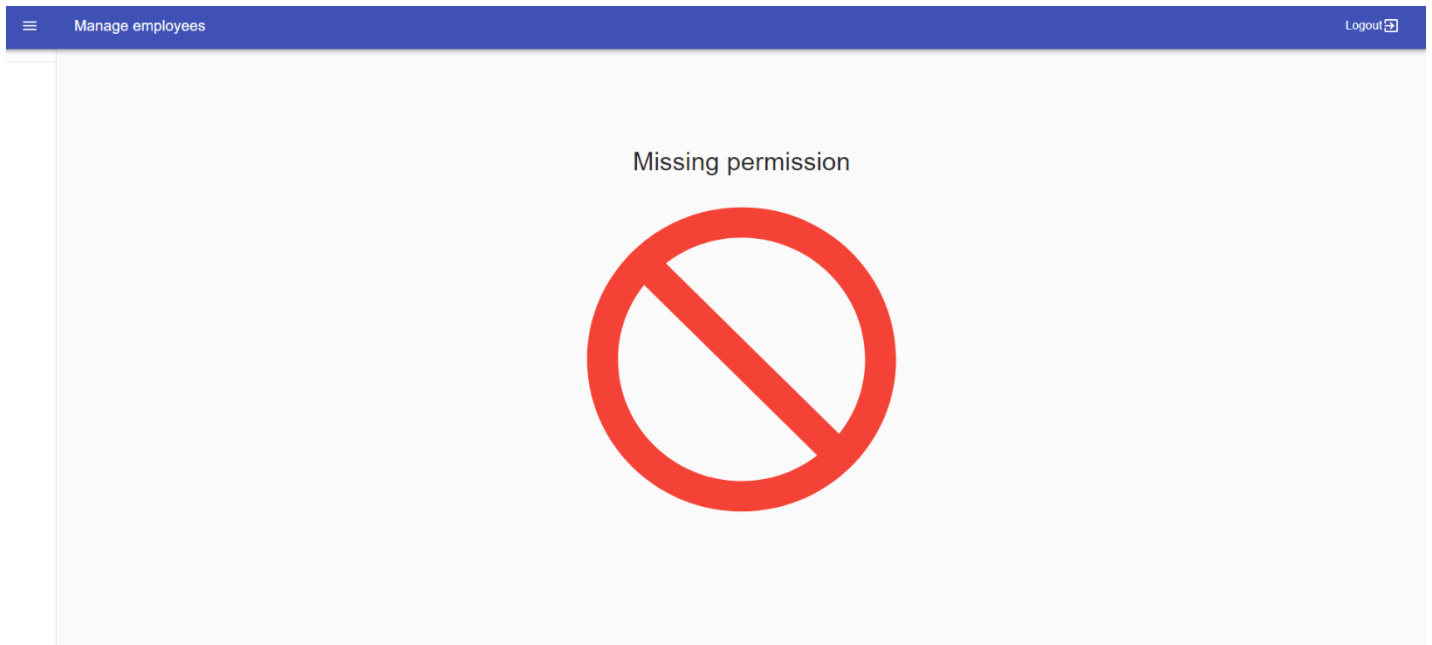


Figure 43. Access denied

## 5.14 Login page

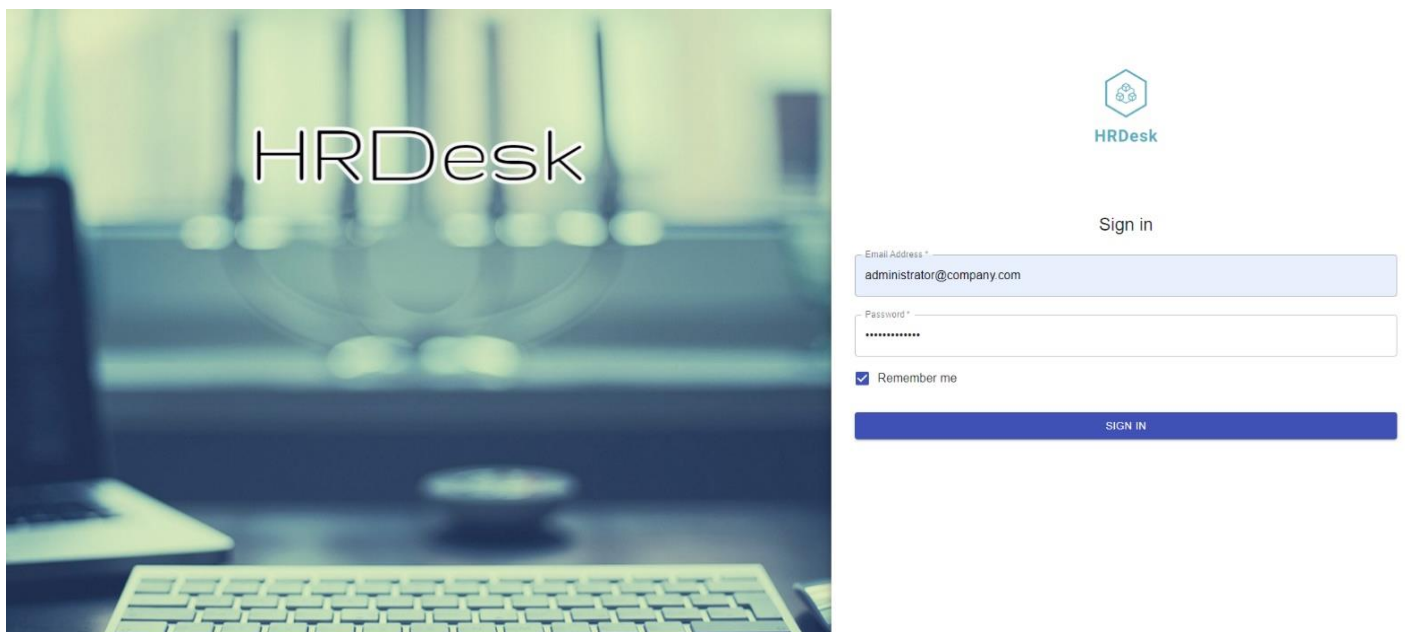


Figure 44. Login page

The login page doesn't have a register possibility, the only person that can create an user is the administrator or a person with Module 9. When the user logs in he has the possibility to check the remember me option which will store it's token to the localStorage instead of sessionStorage. For the time being the token is valid a month. If the token expires the user is logged out of the application.

## 5.15 Excel reports

ID	First Name	Last Name	Work email	Team	Office	Function	Salary
1	Andrei Cristian	Marcu	administrator@company.com	Management team	Management office	Company manager	10000
2	Popescu	Alex	popescualex@company.com	Developer team	Developer office 2	Developer	4700
3	John	Doe	john@company.com	Quality assurance team	Developer office 3	Quality assurance	3655
4	Jane	Doe	jane@company.com	Developer team	Developer office 4	Developer	4877
5	Popa	Alex	popaalex@company.com	Developer team	Developer office 2	Developer	3700
6	Alex	Radu	alexradu@company.com	Operational Team	Developer office 2	Devops	4770
7	Cristi	Mihai	cristimihai@company.com	Administrative Team	Administrative Office	Board	8900
8	Alex	Mihai	alexmihai@company.com	Management team	Management office	Board	9900
9	Lazar	Mihai	lazar@company.com	Maintenance Team	Administrative Office	Cook	3800
10	Otto	Popescu	ottopopescu@company.com	Quality assurance team	Developer office 2	Quality assurance	4566
11	Andrei	Marcu	andrei@company.com	Developer team	Developer office 2	Developer	5500
12	Mihaela	Marcu	mihaelamarcu@company.com	Hardware Team	Developer office 2	Project manager	3670

Figure 45. Employees excel report from HRDesk

ID	User	Date	Start hour	End hour	Status	Description
6	Doe John	2021-06-21T03:00:00+03:00	2021-06-24T08:30:00.61+03:00	2021-06-24T17:00:00.61+03:00	0	Work from home
7	Doe John	2021-06-22T03:00:00+03:00	2021-06-24T08:30:00.61+03:00	2021-06-24T17:00:00.61+03:00	1	Work from home
12	Doe John	2021-06-29T03:00:00+03:00	2021-06-24T08:30:00.61+03:00	2021-06-24T17:00:00.61+03:00	1	Work from home
13	Mihai Alex	2021-06-24T03:00:00+03:00	2021-06-24T08:30:00.345+03:00	2021-06-24T17:00:00.345+03:00	1	Work from home
14	Mihai Alex	2021-06-25T03:00:00+03:00	2021-06-24T08:30:00.345+03:00	2021-06-24T17:00:00.345+03:00	2	Work from home
15	Mihai Alex	2021-06-23T03:00:00+03:00	2021-06-24T08:30:00.345+03:00	2021-06-24T17:00:00.345+03:00	2	Work from home
16	Mihai Alex	2021-06-22T03:00:00+03:00	2021-06-24T08:30:00.345+03:00	2021-06-24T17:00:00.345+03:00	2	Work from home
17	Mihai Alex	2021-06-21T03:00:00+03:00	2021-06-24T08:30:00.345+03:00	2021-06-24T17:00:00.345+03:00	2	Work from home
24	Marcu Andrei Cristian	2021-06-28T03:00:00+03:00	2021-06-24T13:00:00.59+03:00	2021-06-24T14:00:00.59+03:00	2	Laborator
31	Marcu Andrei Cristian	2021-06-25T03:00:00+03:00	2021-06-24T22:25:00.191+03:00	2021-06-24T22:25:00.191+03:00	2	laborator
32	Marcu Andrei Cristian	2021-06-29T03:00:00+03:00	2021-06-24T14:00:00.591+03:00	2021-06-24T16:00:00.591+03:00	2	facultate
33	Marcu Andrei Cristian	2021-06-30T03:00:00+03:00	2021-06-24T14:00:00.591+03:00	2021-06-24T16:00:00.591+03:00	2	work from home
34	Marcu Andrei Cristian	2021-06-30T03:00:00+03:00	2021-06-24T08:00:00.591+03:00	2021-06-24T16:00:00.591+03:00	2	work from home
35	Marcu Andrei Cristian	2021-07-01T03:00:00+03:00	2021-06-24T08:00:00.591+03:00	2021-06-24T16:00:00.591+03:00	2	work from home
36	Marcu Andrei Cristian	2021-07-02T03:00:00+03:00	2021-06-24T08:00:00.591+03:00	2021-06-24T16:00:00.591+03:00	2	work from home
37	Marcu Andrei Cristian	2021-07-05T03:00:00+03:00	2021-06-24T08:00:00.591+03:00	2021-06-24T16:00:00.591+03:00	2	work from home
1	Marcu Andrei Cristian	2021-06-24T03:00:00+03:00	2021-06-24T13:20:00.33+03:00	2021-06-24T14:20:00.33+03:00	2	Dentist
2	Marcu Andrei Cristian	2021-06-16T03:00:00+03:00	2021-06-24T13:20:00.33+03:00	2021-06-24T14:20:00.33+03:00	2	Licenta
4	Doe John	2021-06-22T03:00:00+03:00	2021-06-24T11:30:00.61+03:00	2021-06-24T12:30:00.61+03:00	2	Laborator

Figure 46. Leave request excel report from HRDesk



## 6 UML DIAGRAMS

### 6.1 Class diagram

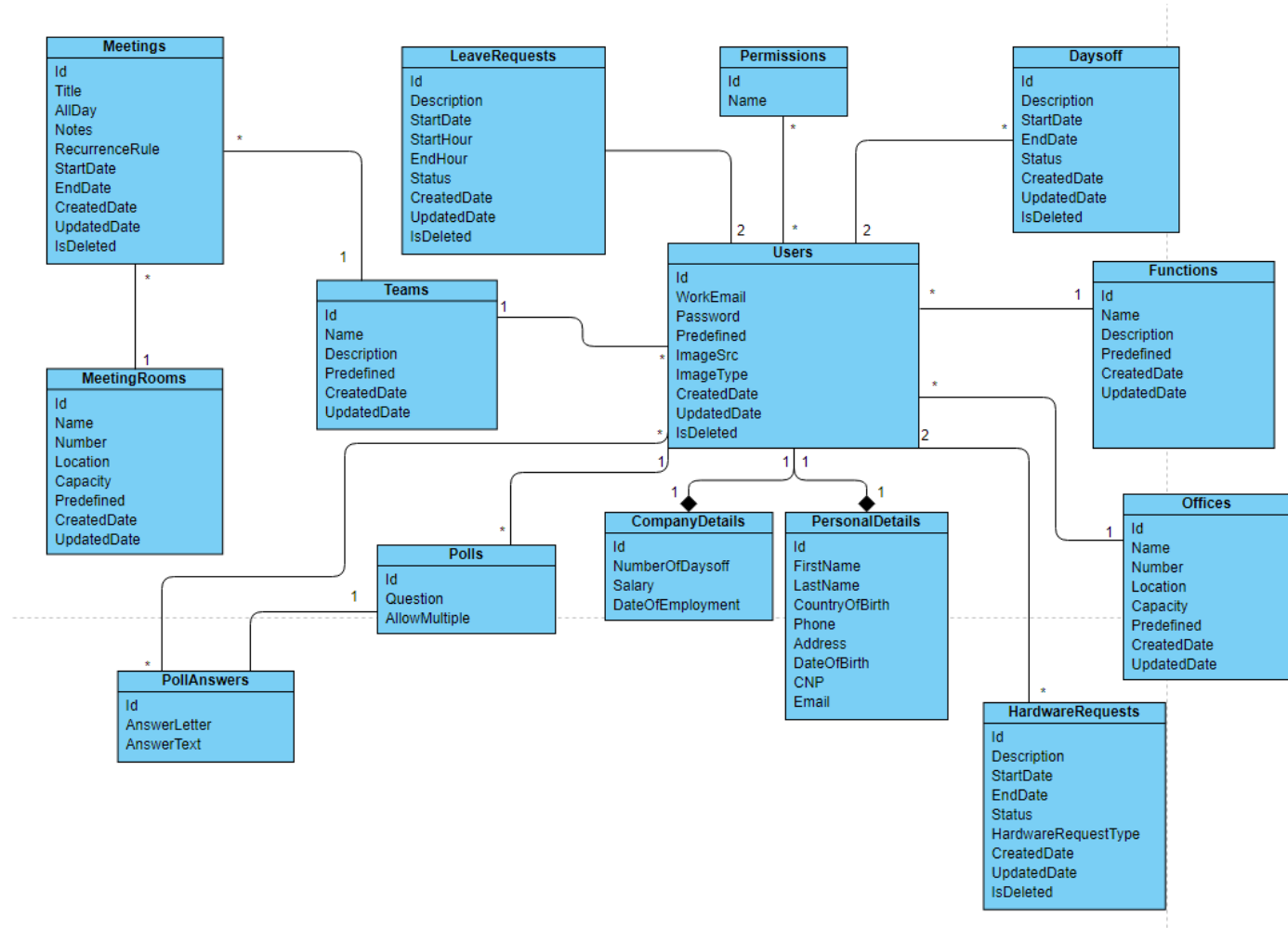


Figure 48. UML Class diagram

## 6.2 Use case diagram – Basic user

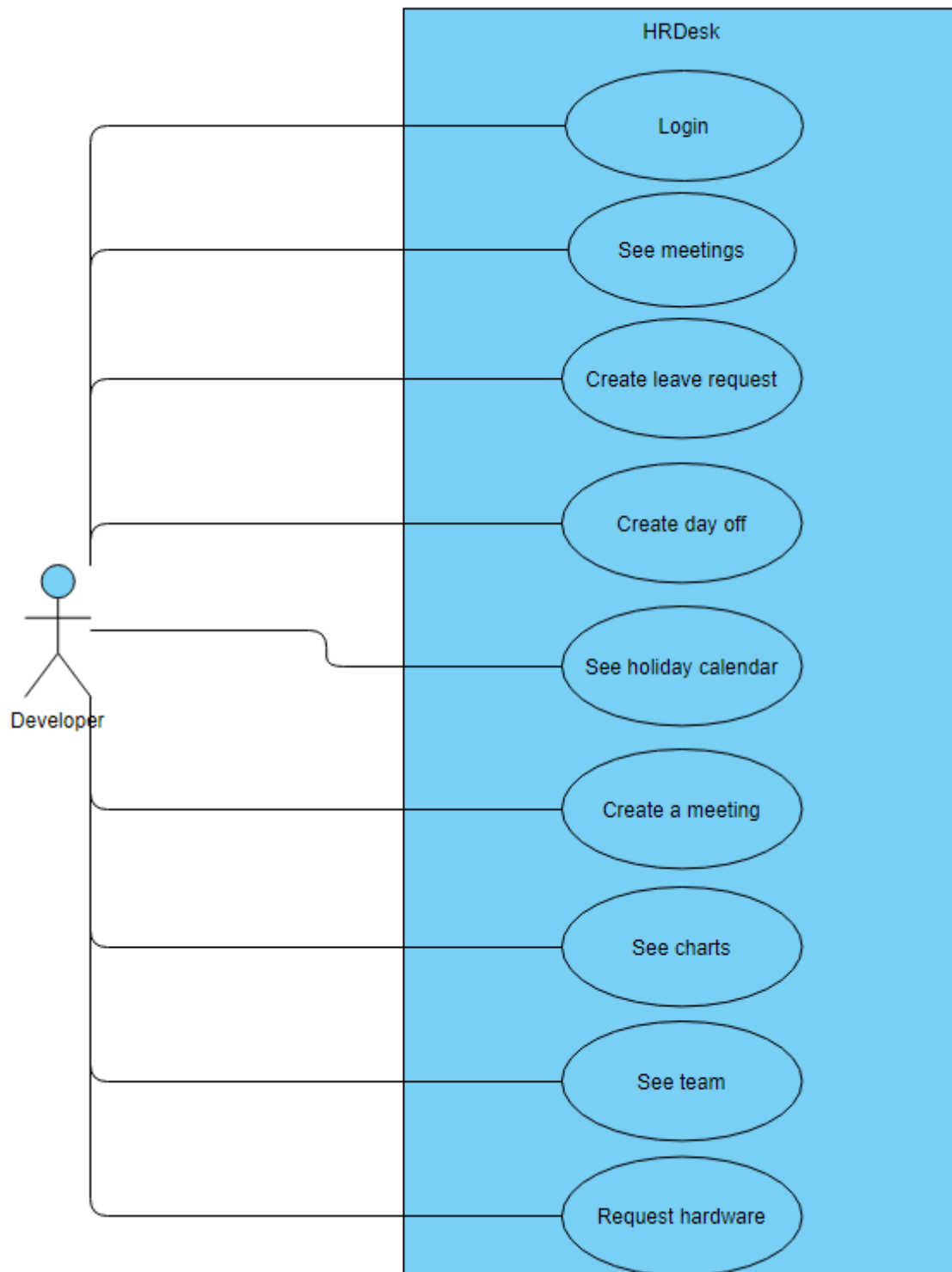


Figure 49. UML Use case diagram – User

### 6.3 Use case diagram – Administrator

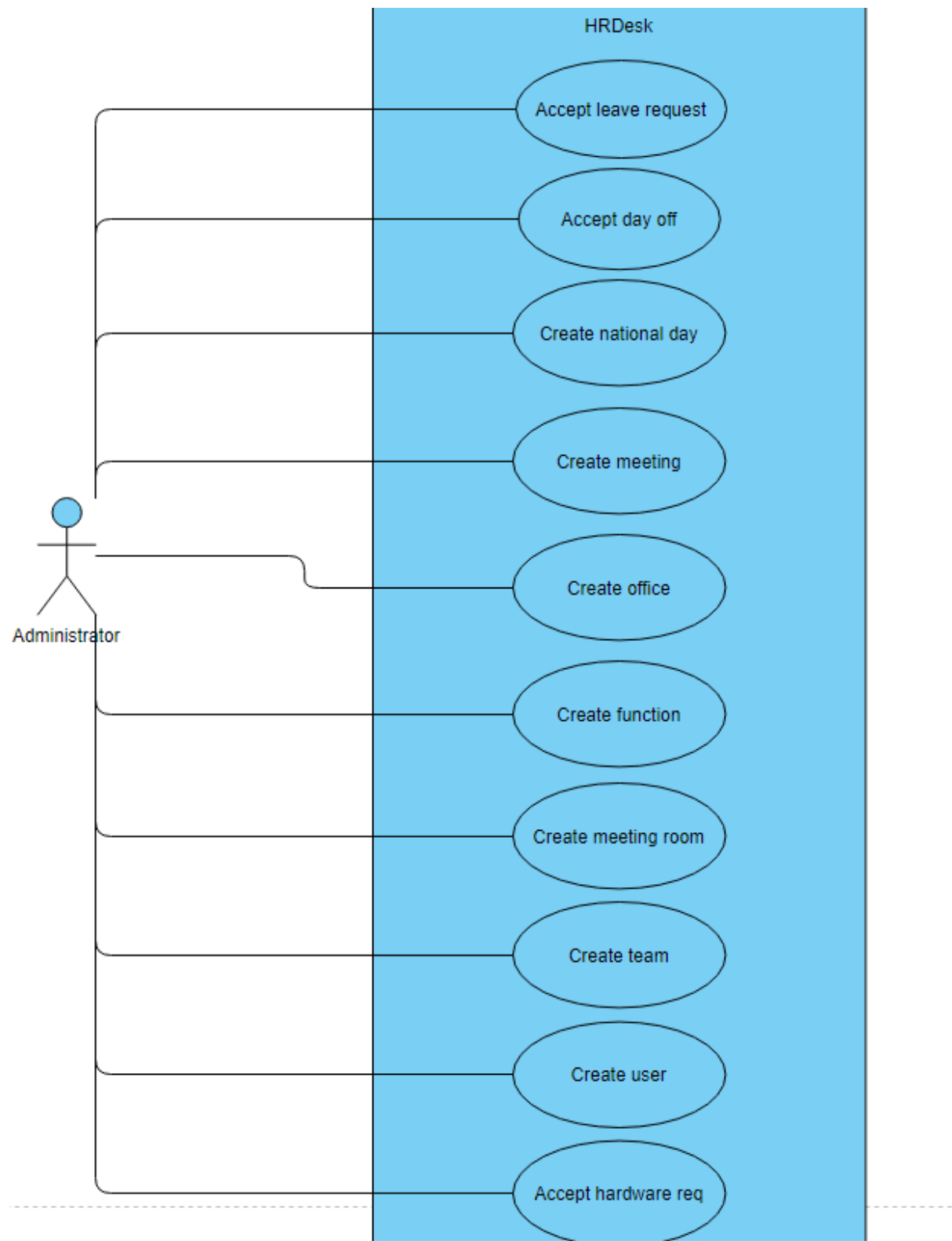


Figure 50. UML Use case diagram – Administrator

## 6.4 Activity diagram – day off

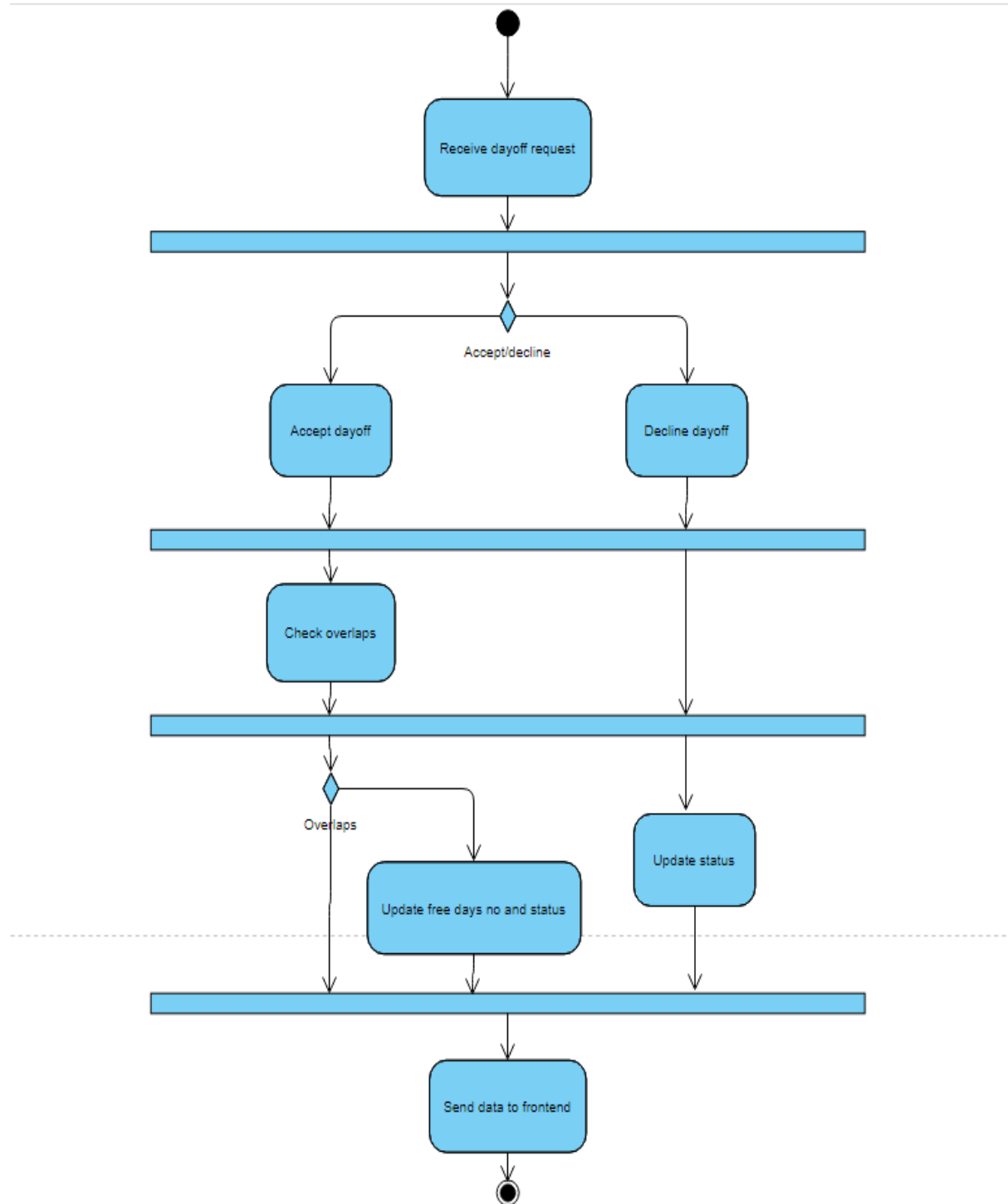


Figure 51. UML Activity diagram - day off



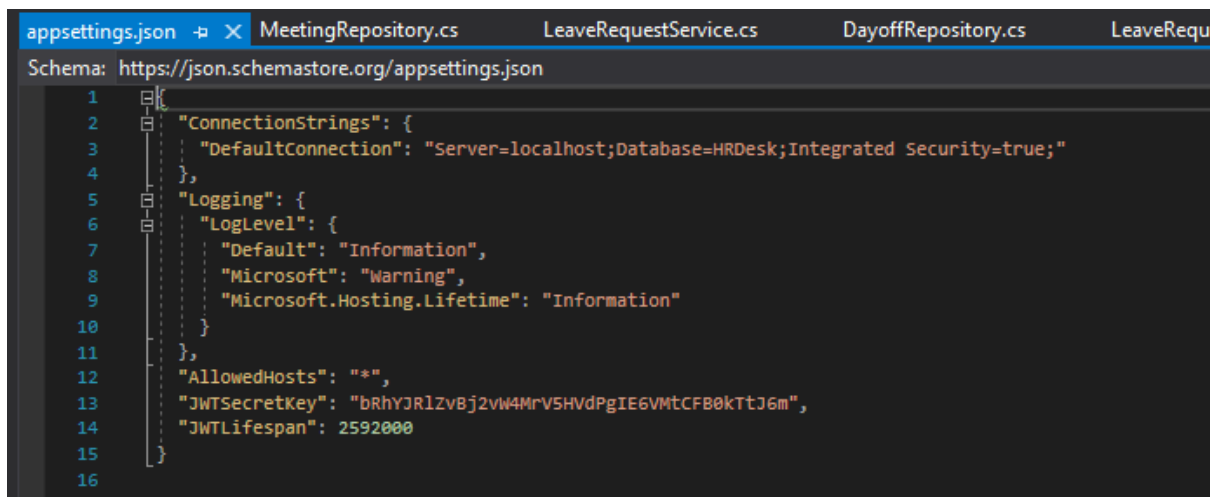
## 7 HOW TO USE

### 7.1 Fetch or download the code

The code can be fetched from Github: <https://github.com/mandrei28/Human-resources-management> from the branch feature/initial\_configuration or main branch / from the CDs. The repository is private due to privacy reasons but I can grant access if needed. In order to fetch it I recommend using Sourcetree, all you need is a Github account, then from sourcetree clone the repository to your local computer. Make sure you're on the feature/initial\_configuration or main branch. Now that we have the code locally we can move on to the next step.

### 7.2 Link and create the database

First of all we need to have SQL Server installed. After we install SQL Server we need to modify the connection string from the "appsettings.json".



```
1  {
2    "ConnectionStrings": {
3      "DefaultConnection": "Server=localhost;Database=HRDesk;Integrated Security=true;"
4    },
5    "Logging": {
6      "LogLevel": {
7        "Default": "Information",
8        "Microsoft": "Warning",
9        "Microsoft.Hosting.Lifetime": "Information"
10     },
11   },
12   "AllowedHosts": "*",
13   "JWTSecretKey": "brhYJRLZvBj2vW4MrV5HvdPgIE6VMtCFB0kTtJ6m",
14   "JWTLifetime": 2592000
15 }
16
```

Figure 52. Connection string

Server = SQL Server hostname(if it's local then you can use localhost instead of the computer name)

Database = Database name, this can be anything.

Now that the connection is made, the next step is to actually create the database. In order to create the database we need to open the package manager console from Tools -> NuGet package manager -> Package manager console. We select the target project as HRDesk.Infrastructure(because the database context is placed there) and then we run the command Update-Database. This will automatically create the tables. Also there are some entries inserted by the seed method, like the first user.

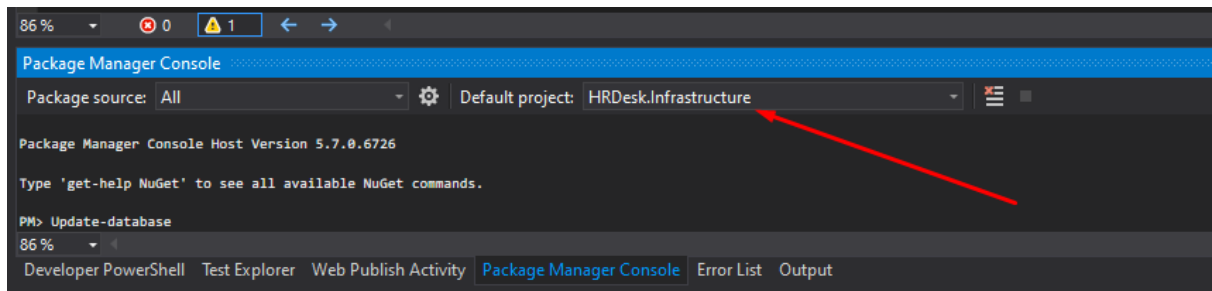


Figure 53. Apply migration

After this the backend solution should be up and running.

```

124     modelBuilder.Entity<CompanyDetails>().HasData(new CompanyDetails
125     {
126         Id = 1,
127         DateOfEmployment = new DateTime(2021, 6, 5),
128         NumberOfDaysoff = 20,
129         Salary = 5000,
130     });
131     // "Email" : "administrator@admin.com",
132     // "Password" : "administrator"
133     modelBuilder.Entity<User>().HasData(new User
134     {
135         Id = 1,
136         CreatedDate = new DateTime(2021, 6, 5),
137         IsDeleted = false,
138         PersonalDetailsId = 1,
139         CompanyDetailsId = 1,
140         FunctionId = 1,
141         OfficeId = 1,
142         TeamId = 1,
143         Password = "AQAAAAEAACcQAAAAECB47G0oGMZ5MBmFGNmX95ff8JEzfsP/77XSzbcpeS60akk3M/CXQ0u10M2SWn/pzg==",
144         WorkEmail = "administrator@company.com",
145         UpdatedDate = new DateTime(2021, 6, 5),
146         Predefined = true,
147         ImageSrc = null,
148     });
149 
```

Figure 54. Default user

The default user has all the modules enabled. Credentials:

Username: [administrator@admin.com](mailto:administrator@admin.com)

Password: administrator(encrypted in the database).

## 7.3 Frontend solution setup

All the packages needed by the frontend solution are stored in the package.json file. Due to this running the “npm install” command will automatically get the dependencies.

While running the commands the path needs to look like this: Licenta-Human-Resources-Management\WebApp\hrdesk

So the steps are: First run the “npm install” command

Then *npm build* to check that everything is ok

And in the end *npm start*, which will cause the website to open at <http://localhost:44332/>

Using the credentials provided above the user can log in the application and begin the manual setup phase. He needs to go to the administrative levels and add offices, teams, functions, meeting rooms and employees. From this point the application can be ran in anyway the user desires. The setup is complete.

## **8 TERMS OF SERVICE**

### **8.1 Author**

This bachelor thesis document is done by Marcu Andrei Cristian, student at Faculty of Automatics, Computers and Electronics in Craiova, Romania.

### **8.2 Terms of use**

The application has no licenses constraints, everything that was used inside the application was free to use. The application was developed for educational purpose.

## 9 CONCLUSION

Both ASP.NET Core API and React are modern technologies that are used by many companies in the development of web applications. These technologies are continuously growing by receiving constant updates. The apparition of React and the other similar framework(Angular, Vue.js) simplified the web programming process by a lot. Since the development became easier a new function appeared „Full stack developer” which handles both backend and frontend of the application. In the past those two were splitted on Frontend Developer and Backend developer.

Given the topic of the application I consider that choosing React and ASP.NET Core was the perfect choice for a web application. While developing the applicaiton I’ve improved my skills with these two technologies and also applied some new knowledge like Redux and axios.

Since the application isn’t working with big sets of data, Entity Framework can easily handle the database query part. If at any time the application will begin to move slow Entity Framework LINQ queries will have to be replaced by classic Stored Procedure, where we can control the code that is actually executed behind. Entity Framework creates it’s own query, which cannot be modified by the actual user.

At the moment of this document there are some limitations regarding the performance, the biggest one being the pagination on the grids. The pagination is implemented on the frontend side but on the backend is not yet added in the queries. An improvement would be, when working with a lot of data, to use skip and take while retrieving data.

The topic of the bachelor thesis, human resources management, can be extended way more in the future, we can add for example possibility to add food menus, to create polls, internal competitions, chat, maybe it’s own project media storage and many more.

I’ve tried, as much as possible, to avoid writing duplicate code. On the backend side especially I’ve applied some patterns(Repository Pattern + Unit Of Work) in order to avoid duplicate queries. Also these gave me an extra layer of abstractization between the database frameworks and the actual code. This way if at any time we will have to change the Entity Framework for example this will be done only by changing the code in the repositories without affecting the actual services and controllers.

The application should also be usable on smaller devices, like tablets and laptops, since the layout was made as responsive as possible. There might be issues on the grids with multiple columns but yet it’s still usable with a scroll option.

From security standpoint I've tried to keep the application as secure as possible since it contains sensitive data, meaning that accessing a module from frontend which is not available for that specific user would redirect the user to the access denied page, also if the user tries to request some data from a module that is not available he will receive a HTTP 401 ERROR Unauthorized, and even more sensitive data like password is encrypted in the database.

In conclusion, this bachelor thesis gave me the opportunity to use the knowledge acquired across the years of university one more time in a single project. This is the most complex application I've developed on my own, and I hope it will mark all the requirements of the Human Resources Management topic.

## 10 BIBLIOGRAPHY

[MAR00] - *Design Principles and Design Patterns*, Robert C. Martin, 2000

[FRE20] – *Head First Design Patterns*, Eric Freeman & Elisabeth Robson, 2nd Edition, O'Reilly Media, 2020

[GAR18] – *Redux in Action*, Marc Garreau & Will Faurot, Manning Publications, 2018

[LOC19] – *ASP.NET Core in Action*, Andrew Lock, First edition, Manning Publications, 2019

[GOR20] - *Practical Entity Framework: Database Access for Enterprise Applications*, Brian L. Gorman, Apress, 2020

[MAR17] - *React Quickly: Painless web apps with React, JSX, Redux, and GraphQL*, Azat Mardan, Manning Publications, 2017

## 11 WEB REFERENCES

- [1] ASP.NET Core logo - [https://commons.wikimedia.org/wiki/File:.NET\\_Core\\_Logo.svg](https://commons.wikimedia.org/wiki/File:.NET_Core_Logo.svg) - Accessed 5th June 2021
- [2] ASP.NET Core Wikipedia - [https://en.wikipedia.org/wiki/ASP.NET\\_Core](https://en.wikipedia.org/wiki/ASP.NET_Core) - Accessed 5th June 2021
- [3] Microsoft – Introduction to NuGet - <https://docs.microsoft.com/en-us/nuget/what-is-nuget> - Accessed 5th June 2021
- [4] Netsolutions – ASP.NET Core vs ASP.NET - <https://www.netsolutions.com/insights/net-core-vs-net-framework/> - Accessed 5th June 2021
- [5] Entity Framework Core Wikipedia - [https://en.wikipedia.org/wiki/Entity\\_Framework](https://en.wikipedia.org/wiki/Entity_Framework) - Accessed 6th June 2021
- [6] Entity Framework Code First Approach – EntityFrameworkTutorial - <https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx> - Accessed 7th June 2021
- [7] React ( JavaScript library ) Wikipedia - [https://en.wikipedia.org/wiki/React\\_\(JavaScript\\_library\)](https://en.wikipedia.org/wiki/React_(JavaScript_library)) - Accessed 8th June 2021
- [8] React logo - [https://en.wikipedia.org/wiki/React\\_\(JavaScript\\_library\)#/media/File:React-icon.svg](https://en.wikipedia.org/wiki/React_(JavaScript_library)#/media/File:React-icon.svg) - Accessed 8th June 2021
- [9] What is JSX? - <https://www.reactenlightenment.com/react-jsx/5.1.html> - Accessed 12th June 2021
- [10] React Redux logo - <https://commons.wikimedia.org/wiki/File:Redux.png> - Accessed 14th June 2021
- [11] SQL Server Wikipedia RO - [https://ro.wikipedia.org/wiki/Microsoft\\_SQL\\_Server](https://ro.wikipedia.org/wiki/Microsoft_SQL_Server) - Accessed 14th June 2021
- [12] SQL Server Wikipedia EN - [https://en.wikipedia.org/wiki/Microsoft\\_SQL\\_Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) - Accessed 16th June 2021
- [13] GitHub logo - [https://en.wikipedia.org/wiki/GitHub#/media/File:GitHub\\_logo\\_2013.svg](https://en.wikipedia.org/wiki/GitHub#/media/File:GitHub_logo_2013.svg) - Accessed 20th June 2021
- [14] GitHub wikipedia - <https://en.wikipedia.org/wiki/GitHub> - Accessed 22th June 2021
- [15] SOLID Principles – Wikipedia - <https://en.wikipedia.org/wiki/SOLID> - Accessed 24th June 2021
- [16] SOLID Principles defined by pictures - <http://web.archive.org/web/20160521015258/https://lostechies.com/derickbailey/2009/02/11/solid-development-principles-in-motivational-pictures/> - Accessed 24th June 2021



- [17] Liskov Substitution Principle - <https://stackoverflow.com/questions/56860/what-is-an-example-of-the-liskov-substitution-principle> -Accessed 24th June 2021
- [18] Dependency inversion principle - [https://en.wikipedia.org/wiki/Dependency\\_inversion\\_principle](https://en.wikipedia.org/wiki/Dependency_inversion_principle) -Accessed 24th June 2021
- [19] <https://jasonwatmore.com/post/2019/10/16/aspnet-core-3-role-based-authorization-tutorial-with-example-api> - Accessed 15th May 2021(used for development)
- [20] <https://geekrodion.com/blog/asp-react-blog/authentication> - Accessed 15th May 2021(used for development)
- [21] <https://stackoverflow.com/questions/43051291/attach-authorization-header-for-all-axios-requests> - Accessed 16th May 2021(used for development)
- [22] <https://stackoverflow.com/questions/47476186/when-user-is-not-logged-in-redirect-to-login-reactjs> - Accessed 17th May 2021(used for development)

## A. SOURCE CODE

Source code can be found on Github: <https://github.com/mandrei28/Human-resources-management> . The repository is private. I can provide access to it if needed

## **B. WEBSITE**

The website is not hosted, but here is the GitHub repository :  
<https://github.com/mandrei28/Human-resources-management> . The repository is private. I can provide access to it if needed

## **C. CD / DVD**

Autorul atașează în această anexă obligatorie, versiunea electronică a aplicației, a acestei lucrări, precum și prezentarea finală a tezei.



# INDEX

## B

Bibliography ..... 54

## C

CONCLUSION ..... 52

CD / DVD ..... 59

## D

Declaratie de originalitate..... iv

Database structure..... 23

## F

Figuri ..... 4

Formulele matematice ..... 4

## H

HRDesk Modules ..... 27

HOW TO USE..... 48

## I

Introduction ..... 1

Index ..... 60

## L

LIST OF FIGURES ..... xv

LIST OF TABLES..... xvii

## P

Project summary ..... ix

Project structure ..... 13

## S

Source code..... 57

## T

Table of contents..... xii

Technologies ..... 3

Terms of service..... 13

## U

UML Diagrams ..... 44

## W

Web References ..... 55

Website ..... 58