Human Resources Allocation Solution

BÂLTAC Mihai-Cristian
The Bucharest University of Economic Studies, Romania
baltacmihai19@stud.ase.ro

As technology advances, so do its applications and standards. We are at a crossroads in a civilization that has grown based on the automation of operations and the development of technology to better human lives. As additional programs that do the same thing arrive, both large and small businesses utilize them, promoting their development. The approach in this paper is to address the major issue, which is the most frequently utilized capabilities in a company, whether it is IT or event production. My work involves minimizing these applications and developing a standard that may subsequently be updated on needs and demand.
Keywords: *HRIS, Tasks, Meetings, Resource Allocation, Management*

1 Introduction

The employees, in addition to the programs they use to do the work, may also utilize time management or managerial applications. According to an analysis by Okta Inc. [1], the number of software applications used by major organizations in all industries increased by 68% between 2014 and 2018, with an average of 129 apps per company at the end of 2018. We define a large firm as one with more than 2000 people and a small business as one with fewer than this amount. And small organizations tend to use more applications, with the average number increasing from 53 in 2015 to 73 in 2017. In 2018, more than 10% of businesses had more than 200 enterprise applications in their portfolio.

According to Bill Swanton, vice president and analyst at Gartner Inc., many firms utilize numerous programs that accomplish the same thing, since each person has a favorite application. Many of them are attempting to adopt "application streamlining" in order to standardize applications and minimize total business application costs.

Ticketing, meeting or planner, task tracking, or the application where each employee fills his activity (either the tasks he worked on or the meetings he attended) are all apps that are not missing from the portfolio of enterprise applications, whether we are talking about small or large businesses. Not all companies have a free day management application; they use the classic way: an application for the human resources department.

According to Luke Marson's post *The 7 most effective employee experience applications* [2], the HRIS application (or The human resource information system) is first and foremost, being the most effective application in his opinion. Collaboration apps, which are related to systems that record meetings, tasks, projects, and important information about them, are second. Ticketing applications are ranked fourth.

HRIS is a kind of business program that allows companies to store employee data, handle typical HR processes, and perform important HR tasks such as payroll and benefits administration.

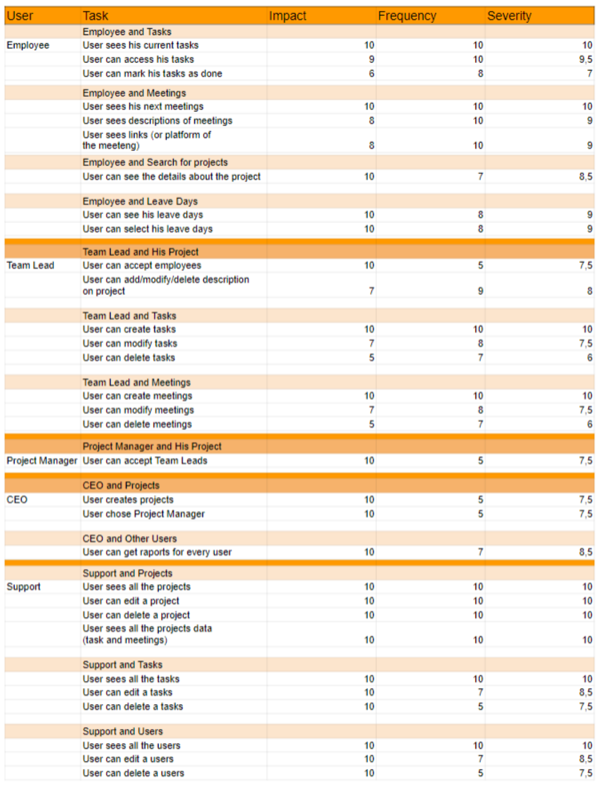
An employee self-service portal, payroll, workforce management, recruitment and hiring, benefits administration, and talent management are all features of HRIS solutions. The individual modules that make up a comprehensive suite of HR solutions are frequently used to provide these features.

By first establishing a document, called a "ticket," a ticketing system documents the interactions on a support or service case.

Both the representative and the customer have access to the ticket, which keeps account of their interactions in one spot.

Either party can return to the thread at any time.
After creating the ticket, advocates can work on the issue on their own. When they have updates or solutions, they can notify the client via the ticket. Meanwhile, if the customer has any questions, they can contact the customer support person by ticket.
The agent is then notified by the ticketing system that a response has been recorded on the ticket, allowing them to resolve it immediately.
Jira [3] is one of the most widely used ticketing applications. Jira is a project management and issue tracking software program. The Atlassian tool, which was created in Australia, is now widely used by agile development teams to monitor bugs, stories, epics, and other activities.
There is no universal template that can be used to all projects; we can see that a variety of Agile time management approaches (Scrum, eXtreme Programming) have emerged, therefore, different programs are utilized for different types of project (Jira vs SpiraTeam) The approaches stated above are only a few examples; they can be applied to technical projects, IT, and programming in general. But there are methodologies and strategies for other areas as well: Marketing, Sales.
The application I am presenting is one that combines the most commonly utilized features of the previous programs. I have listed the most often used features by different types of users. Users, action, impact, frequency, and severity are described in the next paragraphs.
The following people will be considered users:

- *an employee* with access to tasks, meetings, projects, and vacation time;
- *The Team Lead* is in charge of coordinating a department;
- each project can have many departments, which are created by the *Project Manager* in charge of the project to which it is allocated.
- *The CEO* is in control of adding and modifying projects, as well as generating monthly reports for employees.
- *Support* is a person who has complete access to all databases and the ability to add, alter, and delete anything from the database.



User	Task	Impact	Frequency	Severity
	Employee and Tasks			
Employee	User sees his current tasks	10	10	10
	User can access his tasks	9	10	9,5
	User can mark his tasks as done	6	8	7
	Employee and Meetings			
	User sees his next meetings	10	10	10
	User sees descriptions of meetings	8	10	9
	User sees links (or platform of the meeteng)	8	10	9
	Employee and Search for projects			
	User can see the details about the project	10	7	8,5
	Employee and Leave Days			
	User can see his leave days	10	8	9
	User can select his leave days	10	8	9
	Team Lead and His Project			
Team Lead	User can accept employees	10	5	7,5
	User can add/modify/delete description on project	7	9	8
	Team Lead and Tasks			
	User can create tasks	10	10	10
	User can modify tasks	7	8	7,5
	User can delete tasks	5	7	6
	Team Lead and Meetings			
	User can create meetings	10	10	10
	User can modify meetings	7	8	7,5
	User can delete meetings	5	7	6
	Project Manager and His Project			
Project Manager	User can accept Team Leads	10	5	7,5
	CEO and Projects			
CEO	User creates projects	10	5	7,5
	User chose Project Manager	10	5	7,5
	CEO and Other Users			
	User can get raports for every user	10	7	8,5
	Support and Projects			
Support	User sees all the projects	10	10	10
	User can edit a project	10	10	10
	User can delete a project	10	10	10
	User sees all the projects data (task and meetings)	10	10	10
	Support and Tasks			
	User sees all the tasks	10	10	10
	User can edit a tasks	10	7	8,5
	User can delete a tasks	10	5	7,5
	Support and Users			
	User sees all the users	10	10	10
	User can edit a users	10	7	8,5
	User can delete a users	10	5	7,5

Fig.1. Analyzes of Application Activities Table

There are three types of profiles in the app: employee, CEO, and support. The profile for the roles of Team Manager and Project Manager is similar to that of an employee, with the exception that they have additional functionalities on the project where they hold this position.
For each role, the Action column gives a list of available actions. As a result, depending on the type of user, it can perform a variety of tasks.
The Impact column indicates the user's level of need for that activity. The action is graded on a scale of 1 to 10, with 1 indicating that it is not very important and

10 indicating that it is extremely important.
The Frequency column shows how often the user will perform that everyday action. The user is ranked on a scale of 1 to 10, with 1 indicating that the user does not perform the activity frequently and 10 indicating that the user performs the action multiple times per day.
The arithmetic mean of the impact column and the frequency column is the Severity column. This is estimated to determine which activities are necessary and how the design should be conceived to maximize the user experience.

2 Technical Specification

Frontend technologies

For frontend React.js and Sass technologies were used.
Facebook created the React Js user interface library in JavaScript. It offers profound insights on how to work with the DOM (Document Object Model), organize your app's data flow, and consider user interface elements as separate components. [4]
React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript toolkit for creating UI components-based user interfaces.
React can be used to create single-page or mobile applications as a foundation. React, on the other hand, is solely concerned with state management and rendering that information to the DOM, so constructing React apps frequently necessitates the usage of extra frameworks for routing and client-side functionality.
One issue that React solves, to the detriment of traditional web applications, is efficient DOM processing. Each website contains a DOM that shows the page components; they are organized as nodes and objects and may be updated using javascript. It is shaped like a tree. When javascript makes significant modifications to the conventional DOM, it is rendered, which can become inefficient. React js introduces the concept of Virtual DOM, which duplicates the standard DOM and renders only the updated nodes or objects.

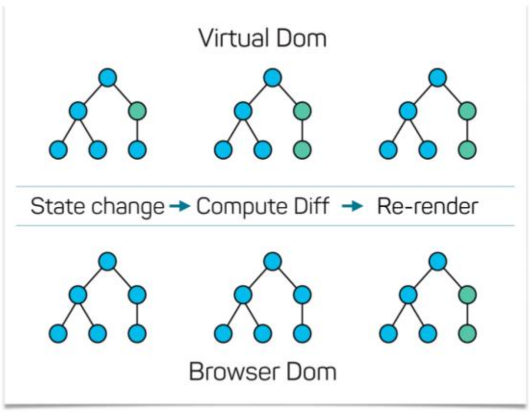


Fig.2. Virtual DOM Example [5]

Fig.2. shows how the virtual DOM modifies a node (State Change) and changes the color from blue to green. Then we modify all the nodes that have the updated node (Compute Difference) as the parent, and we re-render the Virtual DOM in the Browser style (the DOM that the browser sees).
Class React and Functional React are the two types of React. Functional React is becoming increasingly popular because it makes use of the fact that Javascript is a functional programming language. A component of the React class is constructed by extending the React class and rendering HTML code, which is returned by the render() function. The functional React component is given by a function that returns HTML code. It is significantly easier to design a component when using functional react; however, this is a drawback when utilizing a component as a class and want to use the OOP (Object-Oriented Programming) paradigm. Because the two types are compatible, we may have a component in the form of a class that calls another component in the form of a function.

CSS (Cascading Style Sheets) is a fundamental technology of a web page. CSS is the visual component of a website, covering from layout to text color. We may use it to create multiple styles for different

devices or screen sizes.
A CSS preprocessor is a scripting language that allows programmers to write code in one language and then compile it into CSS. Less and Stylus are two well-known examples of preprocessors. Sass is probably the most popular right now.
Sass (short for "Syntactically Awesome Style Sheets”) is a CSS extension that allows us to use variables, nested rules, inline imports, and other features. It also helps organize and allows one to produce style sheets more quickly.
When Chris Eppstein released Compass in 2009, a project specifically designed to handle Sass packages and encourage open-source Sass code sharing, Sass attracted widespread notice.
Eppstein identified an opportunity for Sass to adjust pre-build class names' copy-pasting libraries. [6]
When we compile the sass code, we get the CSS code. As a result, the capacity of Sass to affect the DOM should not be confused; it just helps to create CSS code more effectively. Some CSS constraints are also present in Sass, such as the inability to access an element's parent and unconnected text flows.
There are various ways to compile Sass. Installing an extension that achieves this is the simplest, but also the most recommended option. Another option is to utilize bash scripts to do this. The global Sass mode installation is necessary for the device to recognize these instructions. Uncompiled Sass is often permitted in frameworks. As a result, when the app builds the page code, it also produces the Sass.
The syntax of Sass code is similar to that of the Python language since it employs indentation, no brackets, and no semicolons to conclude a line.
Scss is a Sass variant that has the same capability as Sass but has a code syntax that is similar to conventional CSS. Scss code will be utilized in the provided application.

Backend technologies
For backend Node.js, Express and Sequelize technologies were used.
Node JS can be thought of as a JavaScript runtime environment built on top of Google's V8 engine. As a result, it gives us a context in which we can write JavaScript code on any platform that has Node.js installed.
Ryan Dahl gave a presentation at JSXonf in 2009 that permanently impacted JavaScript. He introduced Node.js to the JavaScript community during his lecture. [7]
Node.js allows developers to utilize JavaScript to create command-line tools and server-side scripting, which involves running scripts on the server before sending the page to the user's browser. As a result, Node.js symbolizes a "JavaScript everywhere" paradigm, bringing online application development together around a single programming language rather than separate languages for server- and client-side scripts.
There are benefits and drawbacks to Node.js. Some of its benefits include fast performance for real-time applications, simple scalability for recent software, a quick learning curve for developers already familiar with Javascript, and improved performance of applications. The disadvantages of using a relatively new technology include: limited support from existing libraries and a lack of experienced developers. Javascript also has other drawbacks that affect node.js, such as an unstable API that encourages frequent code changes, Javascript is also a language with the Asynchronous Programming Model paradigm that makes code maintenance challenging.
The core concept of Node is the usage of non-blocks, which are extremely effective and lightweight in comparison to other technologies because Javascript is an event-driven programming language (anything starts with an event). Node.js seeks to address a market issue rather than try to displace competing technologies. It is useful

for constructing fast and scalable applications because it is efficient and writes relatively quickly. It is not suggested to utilize it for CPU operations or any type of intensive processing, since these will neutralize all of its benefits.

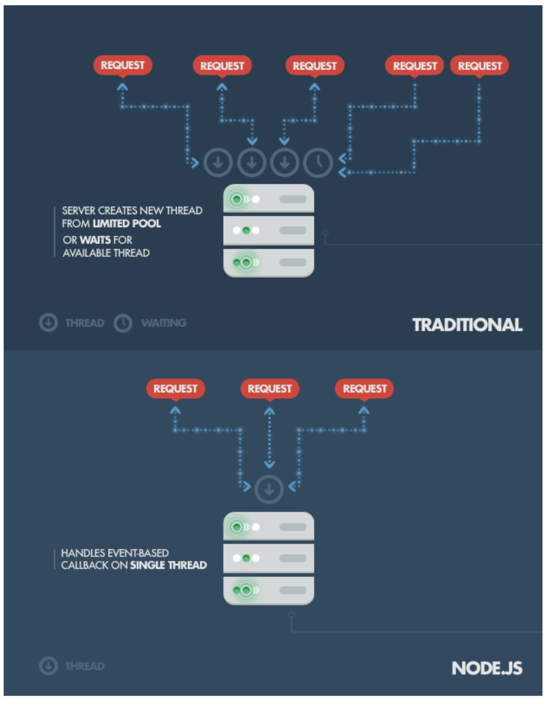

REQUEST
REQUEST
REQUEST
REQUEST
REQUEST
SERVER CREATES NEW THREAD FROM LIMITED POOL OR WAITS FOR AVAILABLE THREAD
THREAD
WAITING
TRADITIONAL
REQUEST
REQUEST
REQUEST
HANDLES EVENT-BASED CALLBACK ON SINGLE THREAD
THREAD
NODE.JS

Fig.3. Traditional Web Server vs Node.js [8]

The architecture of a conventional Web server and Node.js are contrasted in **Fig.3.** Accordingly, in the conventional form, every new request generates a new thread, filling the system memory; when there is no longer any further memory, it waits for a thread to be released. Instead, Node.js uses non-blocking I/O calls and runs on a single thread, allowing hundreds of active connections to be called simultaneously.

Express.js, or simply Express, is a back-end web application framework for Node.js that was distributed under the MIT license as free and open-source software. It is intended for the development of web applications and APIs. It has been dubbed Node.js' de facto standard server framework.
Express. js is a web framework built around the Node.js http module and the Connect components. Middlewares are the term for these components. Developers are the epicenter of the framework's concept, namely, configuration over condensation. In other words, developers are free to choose which libraries they require for a given project, giving them a great deal of freedom and customization. [9]
Express is a simple and adaptable framework that helps in building reliable online applications. It is Node.js' most widely used framework. It offers a range of techniques for quickly and simply building an API. Both SQL and non-SQL databases can use it. Other well-known frameworks like Feathers, KeystoneJs, NestJs, and many others have been developed on top of this framework because of how commonly used it is in industry.

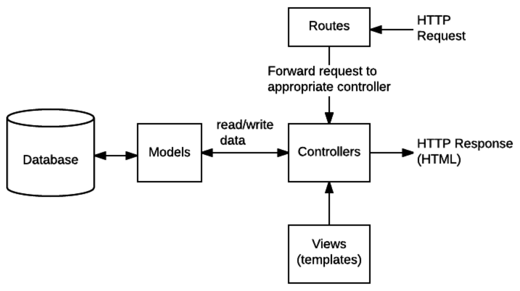


Fig.4. Express Architecture [10]

The architecture of this framework is illustrated in **Fig.10.** Data models, called models, typically define tables. This means that each name of the table is represented by a structure with columns as its elements. Controllers are functions that handle data, act like functions, and receive parameters using a variety of techniques (in the body of a request or as parameters of the function). Routes link the corresponding controller to the request code. As a result, we may specify which requests call which controls on the routes. Controllers render the date using views or templates.

Data conversion between systems that use OOP languages is known as object-relational mapping or ORM. In that

programming language, it generates a "virtual object database" that can be used. In other words, we can use objects to manipulate database data. As a result, we can now use object methods instead of writing SQL code. Therefore, if we change the database, all we need to do is check to see if the ORM we use supports it; if it does, then we don't need to do any code changes.
Prism and Sequelize, which have accumulated over 20,000 github starts each, are the top ORMs for Node js. On the other hand, users use Sequelize more frequently. Both ORMs support six different types of databases. [11]
Sequelize is a promise-based Node.js ORM tool that supports Postgres, MySQL, MariaDB, SQLite, DB2 and Microsoft SQL Server. It includes transaction support, relations, eager and lazy loading, read replication, and other features.
Sequelize uses Semantic Versioning and is compatible with Node v10 and higher.
Sequelize uses objects that are extended from the Model class, as is conventional for ORMs. Thus, object methods are used to carry out actions like Select, Insert, Update, and Delete. The "belongsTo ()" and "hasMany ()" methods specify the connections between tables (in our case, models). Later, these techniques assist us in joining the tables (in Sequelize the "include" method is used).

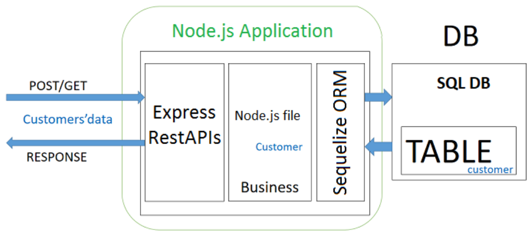


Fig.5. Sequelize Role in Node.js Application [12]

Database Technology

One of the first open-source RDBMSs to be designed and built was MySQL. Although there are many different versions of MySQL available right now, their fundamental syntax is the same. Due to its unique architecture compared to other database servers, MySQL can be used to address a variety of issues. You must understand its design to operate with it because it has a unique architecture.
In contexts with high demand, like web applications, MySQL is flexible enough to operate very effectively. It is adaptable in many aspects, such as you may set it to run effectively on a wide variety of hardware and supports a variety of data kinds. [13]
Because MySQL is based on a client-server architecture, its central component is a MySQL server, which manages all database commands. MySQL was originally built to manage huge databases fast. Various transaction types, including stored procedures, functions, viewers, views, and triggers, are possible. Compared to other databases, MySQL is relatively efficient for read-only commands, but for big testing or sophisticated queries, PostgreSQL is a superior alternative.

3 Implementation of the solution

My objective is to create a web platform that incorporates all of these technologies, allowing managers and human resources to better manage workers on a wide range of projects.
The software has five types of users: employees, team leaders, project managers, CEOs, and support.
The employee can accept assignments, provide comments or be in charge of tasks, look for other individuals, study project strategies, and apply to the project as an employee, team leader, or even project manager.
The team leader may carry out all employee responsibilities, admit people into his project, create a calendar, submit assignments and meetings, and load the strategy.
The project manager not only performs all the duties of a team leader, but also chooses team leaders for the project.
The CEO develops, oversees, and produces monthly reports for each employee.

Support has complete access to all objects and can monitor all projects and events in the application, as well as add, modify, and remove each piece from the database.
A well-defined architecture is required for a better solution to ensure a smooth flow.

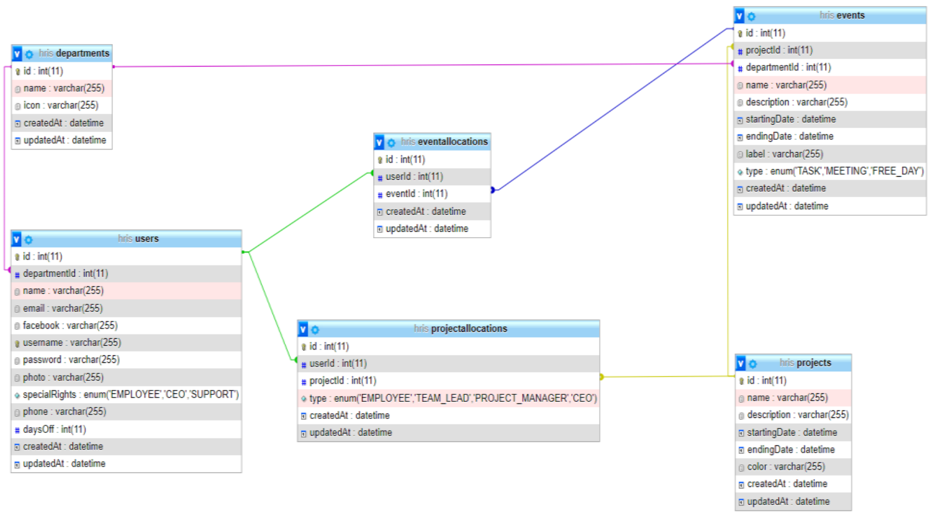


Fig.6. Database Architecture

The User table contains user information such as name, email, Facebook, username and password, photo, phone number, vacation days, and the type of rights, which can be of the following types: employee, CEO, and support. The department is a foreign key for the department table, so we can figure out what interface the user will see based on this information.
Each department's name and icon are listed in the department table.
The Events table has two foreign keys that point to the department and project tables, respectively. Other columns include name, description, label, start and end dates, and type, indicating if the event is a task, meeting, or free day. I chose this format since I didn't want to make tables with the same structure. Aside from the type of event, the only distinction between a task and a meeting is the information in the label column; in the case of tasks, the label column has four possible answers: New, Doing, Done, and Closed. This table, the label table, contains the link to the platform where the meeting is hosted in the case of meetings.
The user and the event are linked through the event allocation table. This table was chosen to prevent the many-to-many relationship between the two tables.
The project table provides the following details: name, description, colour (which is used to visually distinguish the projects), start date, and end date.
In addition to the foreign key relationships between the user and the project, the assigned project table contains a type, which is an enumeration of various types: employee, team lead, project manager, and CEO. Additional activities are established in this project based on these types.

When the application is first launched, a login window displays, requesting the user for login information before sending a request to the backend. If the input is accurate, the frontend receives a json containing the user id and user type. As a result, the program may render the user interfaces and activities. As a result, we have three options for the user: employee, support, or CEO.
The information obtained from json is then

saved as cookies, so that if the user refreshes the page or exits the program and returns later, no additional login is necessary; he may erase this cookie by hitting the Logout button.

The employee interface shows when the user has logged in with his employee credentials. It has access to four different pages: the Dashboard, Tasks, Meetings, and Projects. The Dashboard page is the first and most significant page in terms of User Experience. On this page, the User can view: account information, a calendar for the current month (where he can see which days, he has tasks or meetings), three sections: one for meetings, one for tasks, and one for projects.

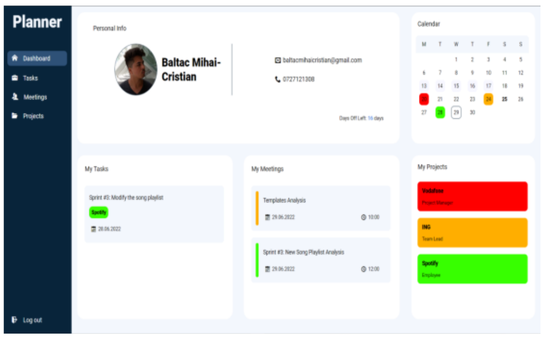


Fig.7. Dashboard

As shown in **Fig.7.**, On the Dashboard page, the user can access a number of sections. A picture, a name, some contact information (phone and email), and the number of free days, they are all found in the first section, which is titled Personal Information. We can see three different types of highlights in the calendar section on the right. If a day is highlighted in colour, it indicates that a task or meeting will take place on that day. This time, the project's colour serves as the background for the event. Days 13 through 17 are highlighted in a very light gray, which indicates that the user is on vacation on those days. On the 25th, which is the current date, there is another kind of highlight. If a date has a border but no background, there will be multiple events that day. We can hover over a day to view more information about the events on that day.

We can add events (tasks, meetings, or free days) directly from the dashboard. When we click on a date in the calendar, a context menu with three options will appear, allowing us to add additional events. When you select the "Add Meeting" or "Add Task" option, a pop-up window will appear with all the details for that event. When we click "Take Vacation," the screen will refresh and we can see that the date we chose is highlighted and that the number of days off has been reduced by 1, indicating that the action was successful.

It takes numerous steps to publish a day of vacation. As a result, we have a project with the id 0 of "FREE DAY" and a department with the id 0 of "FREE DAY". I decided to work with ids of 0 since MySQL permits the usage of primary keys with values of 0 and because the first primary key would automatically start at 1 when the database is reset. Therefore, the value 0 will remain empty.

We have the following logic for posting a day off: We build an event that will receive the start and finish dates, the department id, the description data, and the project id, which will have the number 0 in it. If this event is successfully created, we go to the next step; otherwise, we provide a 500 error code and the words "Server Error." We add our user to the event after it has been created. The user's remaining free days are then determined by accessing him. By deducting one day from these days after this, we can modify the user.

If every step was successfully completed, we return a code of 200 with all of our adjustments. If not, we send the client a code 500 with an error message.

```
postFreeDay: async (req, res) => {
    EventDB.create({
      name: "Off day",
      projectId: 0,
      description: "Free day",
      departmentId:
req.body.departmentId,
      startingDate:
req.body.startingDate,
      endingDate: req.body.endingDate,
      type: "FREE_DAY",
      label: "FREE_DAY",
    })
      .then((event) => {
        EventAllocationDB.create({
          userId: req.body.userId,
          eventId: event.id,
        })
          .then((event) => {
            UserDB.findOne({
              where: {
                id: req.body.userId,
              },
            })
              .then((event) => {
                UserDB.update(
                  {
                    daysOff:
event.daysOff - 1,
                  },
                  {
                    where: {
                      id:
req.body.userId,
                    },
                  }
                )
                  .then((event) => {

res.status(200).send(event);
                  })
                  .catch((error) => {

console.log(error);

res.status(500).send({ message:
"Server error" });
                  });
              })
              .catch((error) => {
                console.log(error);

res.status(500).send({ message:
"Server error" });
              });
          })
          .catch((error) => {
            console.log(error);
            res.status(500).send({
message: "Server error" });
          });
      })
      .catch((error) => {
        console.log(error);
        res.status(500).send({
message: "Server error" });
      });
  }
```

We have three sections at the bottom of the dashboard: one for tasks, one for meetings, and one for projects. Depending on the kind, each part has a variety of cards with useful information. We may click on these cards to get to the card page. If I click on a task, it will take me to the task page; if I click on a meeting, it will take me to the meeting page; and if I click on a project, it will take me to the project page.
The next 2 pages in the user menu contain the tasks list and the meetings list. These are similar, the only difference is the data in the tables.
To add a new event, we simply click the + sign on the page. Depending on the type of event, inputs change. We will create a task, which will include the necessary information, if we are on the task page. Only the projects that the user is enrolled in are visible since the selected project dynamically receives the projects we have.

```
  let { projectId, departmentId } =
useParams();
if (projectId && departmentId) {
    tableDetails = {
      type: "Project",
      userId: userId,
      projectId: projectId,
      departmentId: departmentId,
    };
  } else {
    tableDetails = {
      type: "User",
      userId: userId,
      projects: projects,
    };
 }
```

These two pages, which provide lists of tasks or meetings, can be accessed by the current user or by a department within a project. The same component will be used, with a few user-specific differences. I utilized the URL's parameters for this. If it receives the parameters, it means that a user is accessing the page from a project; if it doesn't have parameters, it means that the current user is accessing the page from the navigation bar.
A task or meeting's page is opened up when we click on it. The dashboard also provides access to an event.
The pages of a meeting and a task are similar, but the information displayed and the actions that can be taken are different. For example, while we can access the meeting link in a meeting page, this action is unnecessary in a task page, because we

don’t have this information, instead, we have a status and the ability to modify that status. New, Doing, Done, and Closed are the available states.

Change event, see participants, and remove event are the common actions. A popup identical to adding an event's opens when the event is changed, except this time the inputs have a default value, which is the current value. We can view, add, or remove participants to that event by choosing the option labelled "See Members." When we click the button to delete an event, a popup window asks us once more whether we are sure we want to do so.

The projects they are assigned to are listed on the projects page along with basic information about them.

We have more options on some other pages depending on the job role we have. We are unable to edit or add departments to a project's details when we have an employee or a team lead position on the project.

When working as the project manager, we can modify a project's details by clicking the pencil icon, which causes a popup to appear with all the project's editable details. If we want to add a new team lead on the project or a department, we click on the plus. A modal will open, in which we are asked for the username of the person we want to add. If we add a team lead, the department to which it belongs is automatically added.

As a Team Lead, you have two options: eliminate the department (by clicking on the trash icon in the first section) or alter the users (add or delete) (by clicking on the pencil to the right of the Members). These features are not available if we have the position of Employee on the project. As a result, they will not appear.

The task and meeting options will open the same component as on the user's task and meeting pages, but this time the department's tasks will be displayed.

```
findDepartmentProject: async (req, 
res) => {
    const { Op } = 
require("@sequelize/core");
    const { projectId, departmentId } = 
req.params;

    await ProjectAllocationDB.findAll({
      where: {
        projectId: projectId,
        type: {
          [Op.and]: [{ [Op.ne]: 
"PROJECT_MANAGER" }, { [Op.ne]: "CEO" 
}],
        },
      },
      include: [
        {
          model: UserDB,

          attributes: ["name", "photo", 
"id"],
          where: {
            departmentId: departmentId,
          },
          include: [
            {
              model: DepartmentDB,

              attributes: ["name"],
            },
          ],
        },
      ],
      attributes: ["type"],
      order: [["type", "DESC"]],
    })
      .then((event) => {
        res.status(200).send(event);
      })

      .catch((error) => {
        console.log(error);
        res.status(500).send({ message: 
"Server error" });
      });
  }
```

There are multiple processes in the above controller. We first look for project allocations and exclude the Project Manager and CEO, because they are automatically in the project. We search all the users of that project for those from the department which is given as a parameter of the route. We provide details like name, image, and ID for these users. The department's name is also included. After completing all of these steps, we have a list of users who belong to our department and do not have the role of Project Manager or CEO on the project. Because the letter "e" from "Employee" occurs in the alphabet before the letter "t" from "Team Lead," it is required to sort the

types of project allocation in descending order.
The interface between the CEO and the employee hasn't changed much. The new "Reports" page and new features are what distinguish them from one another: It features all of the above functionalities in addition to the ability to add and delete projects. Its role is an extension of the Employee's (**Fig.8.**).

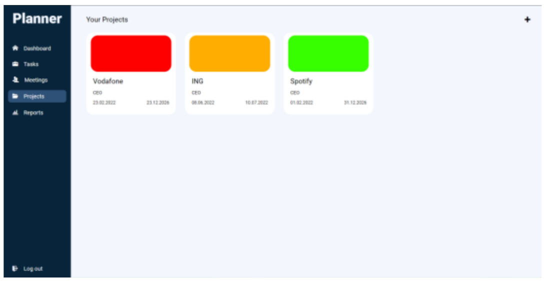


Fig.8. CEO's Projects Interface

The report option, which directs us to a different page, is present in the navbar. The "+" symbol can be seen on the projects page to the right. When the button is touched, a pop-up window appears asking for all the necessary data to establish a project.

```
 postProject: async (req, res) => {
    ProjectDB.create({
      name: req.body.name,
      description:
req.body.description,
      color: req.body.color,
      startingDate:
req.body.startingDate,
      endingDate:
req.body.endingDate,
    })
      .then((project) => {
        ProjectAllocationDB.create({
          projectId: project.id,
          userId: req.body.userId,
          type: "CEO",
        })
          .then((pjAllocation) => {

res.status(200).send(pjAllocation);
          })
          .catch((error) => {
            console.log(error);
            res.status(500).send({
message: "Server error" });
          });
      })
      .catch((error) => {
        console.log(error);
        res.status(500).send({ message:
"Server error" });
      });
  }
```

The logic for starting a project is seen in the above controller. When we initially get the data from the body, we use it to make a project with name, description, color, start and end date. If the project was created successfully then, we assign the user who made the request as CEO.
Whenever we start a new project, the presence of a Project Manager is requested. By selecting the "Add project manager" button in the first section, we can accomplish this. In addition to the project's edit button, we also have a trash button that, when touched, displays a pop-up asking us to confirm that we want to destroy the project. We do this to prevent accidents from occurring if the trash button is unintentionally pressed.

The report page is quite simple, it has two options: departments or users.

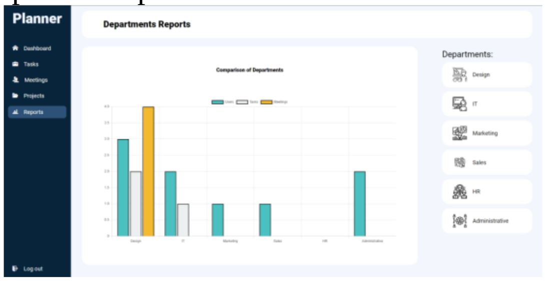


Fig.9. Departments Reports

There are three sections on the department reports page (**Fig.9.**). The page name appears in the first part. A barchart in the second section compares various statistics between departments. The list of departments is represented in the third section. I utilized a third-party library named *chart.js* in the second portion. We can compare tasks and meetings, users and tasks, users and meetings, or even follow the differences between these metrics individually by clicking on one of the rectangles in the legend, which is dynamically erased from the chart. The report page for a department will open if we click on a department in section 3 of the screen.

There are two bar charts on each department's page of individual reports. One describing the distinction between meetings and tasks and another describing projects and members. The functionality of these bar charts is identical to that of clicking on the legend. To make it simpler to calculate additional types of statistics, there is a summary below this. The "See Members" option also directs us to a list of the department's employees. The only distinction between this page and the one in which is reached by selecting the alternate option (the Users on the Reports page), is that on this page, only the users in the department from which we select "See Members," are shown.

On this page, which we access by clicking on "Users" on the "Reports" page, there is a list of all the users on our page. It can be seen that there are users with the color yellow text (which means that person has the role of CEO) and green (indicating that the person is Support). We also have the choice of adding a new user; however, we are not asked for the username or password because those are produced automatically in the backend.

When we click on a user on the page with all users it will lead us to that user's report page. The first two dashboard sections are found on this page; the only difference is that, on the calendar, there is no longer a context menu. Therefore, nothing happens if we click on the date.

The following two ("All time" and "Last month") show the reports from meetings and assignments throughout various time periods. The final panel matches the dashboard's panel for Projects.

```
getStatsLastMonth: async (req, res)
=> {
    const { Op } =
require("sequelize");
    const lastMonth = new Date(new
Date().setDate(new Date().getDate() -
31));

    const { userId } = req.params;
    UserDB.findOne({
      attributes: [],

      include: [
        {
          model: EventAllocationDB,
          attributes: ["eventId"],
          include: [
            {
              model: EventDB,
              attributes: ["type",
"endingDate"],
            },
          ],
        },
      ],
      where: {
        id: {
          [Op.eq]: userId,
        },
      },
    })
      .then((event) => {
        event.dataValues =
event.dataValues.EventAllocations?.map((
e) => {
          if
(e.dataValues.Event.dataValues.endingDat
e > lastMonth)
            return
e.dataValues.Event.dataValues.type;
        });

        let noOfTasks = 0;
        let noOfMeetings = 0;

        event.dataValues.forEach((e) =>
{
          if (e == "TASK") noOfTasks++;
          if (e == "MEETING")
noOfMeetings++;
        });

        event.dataValues.forEach((e) =>
{});
        event.dataValues.Task =
noOfTasks;
        event.dataValues.Meeting =
noOfMeetings;

        res.status(200).send(event);
      })
      .catch((error) => {
        console.log(error);
        res.status(500).send({ message:
"Server error" });
      });
  },
```

We have a bit more difficult process in the controller shown above. The month's past date is the first thing we obtain (we assumed that each month has 31 days). The following stage incorporates gaining access to the database and discovering all the Events that our user attended. If the activity proceeded as intended after getting this information, we compare the deadline for each event. Those that occurred less than a month ago will alter by store only the type of the event, and the other data is discarded as unnecessary. Following the completion of this phase, we go through our list once more and count the instances of events of type "Task" and "Meeting" to add to the variables noOfTasks and noOfMeetings, respectively. The data must then be removed and the calculated numbers for tasks and meetings added.

As an extension of the CEO user, the Support user gets access to all of the CEO's application features. It can also add, edit, and remove departments. It can also alter and delete users. It can also reset passwords. He can view all tasks, meetings, and projects on the dedicated pages, which is another distinction.

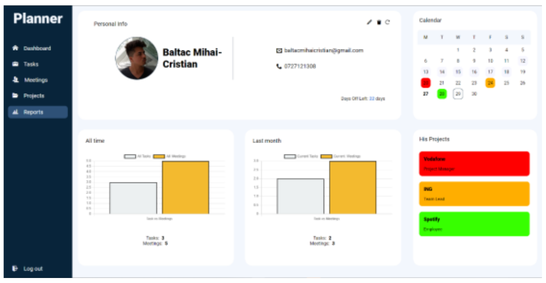


Fig.10. User Reports Support Interface

4 Conclusions

The topic of human resources management is extremely complex, and tactics are always evolving. We aim to allocate resources as effectively as possible so that each project can be structured utilizing a methodology. Numerous applications are continually changing, and new standards are always being released. Both new jobs and modified new processes are established. The application that I've described in my paper is a way to enhance teamwork and effective time management within an organization, assisting both staff members and business owners. The application aims to combine the functionalities currently utilized by the majority of businesses while giving users complete control to develop and alter plans in order to reach their objectives.

By allowing users to customize the sequence of events, project management tactics are made easier to utilize, from choosing which departments we include on each project to structuring tasks and meetings. To make it simpler to build an effective report, the application tries to take into account every event that a user has. Business leaders can track and organize their human resources with the use of the numerous reports produced on various matrices.

Although it has a friendly interface and is a complex tool, anyone can use it. By employing applications that can perform more and can manage these components more simply, I hope to reduce the number of programs needed by businesses.

These standards are always evolving; therefore, the program needs to adapt and provide new functionality as needed. Considering the possibility to extend the application and to make it as efficient as possible, I can state some improvements that the application must have. These are:

- Developing a mobile app (currently the application can only be used on the desktop).
- The user's ability to personalize the application, from the colour scheme to the grouping of dashboard parts.
- Connecting to other calendars, such as Google Calendar and Outlook, so that when an event is added to one, it should also appear in the other.
- Linking up with other user-used applications.
- The ability to send event notifications through email and phone number.
- Making it possible for users to write comments on events would make user communication more effective.
- The ability to connect several events.

• The ability to upload files, including movies and text documents.

References

[1] Loten, A., "*Employees Are Accessing More and More Business Apps, Study Finds"*, The Wall Street Jurnal, 7 February 2019, https://www.wsj.com/articles/employees-are-accessing-more-and-more-business-apps-study-finds-11549580017

[2] Marson, L., "*The 7 most impactful employee experience apps".* Tech Target 31 August 2020, https://www.techtarget.com/searchhrsoftware/tip/The-7-applications-that-impact-employee-experience-the-most

[3] Atlassian. 2022. "Jira | Issue & Project Tracking Software | Atlassian." <https://www.atlassian.com/software/jira?bundle=jira-software&edition=free&tab=release>.

[4] A. and Bush, A., "React.js essentials", Packt Publishing, 2015, p.5.

[5] DEV Community. 2022. "React Virtual DOM Explained in Plain English https://dev.to/adityasharan01/react-virtual-dom-explained-in-simple-english-10j6

[6] Giraudel, H. and Suzanne M., „Jump start Sass"., SitePoint, 2016, p.139.

[7] Satheesh, M., D'Mello, B. and Krol, J. "Web development with MongoDB and NodeJS" , Packt Publishing., 2015, p.2.

[8] Endeev.com. 2022. "Sails.js – Beginning with Node.js – Endeev". <http://www.endeev.com/blog/sails-js-beginning-with-node-js/>

[9] Azat M., "Express.js Guide". Lean Publishing, 2014, p. 2.

[10] Developer.mozilla.org. 2022. "Express Tutorial Part 4: Routes and controllers - Learn web development | MDN". <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes>].

[11] Openbase. 2022. "10 Best Node.js MySQL ORM Libraries in 2022 | Openbase". <https://openbase.com/categories/js/best-nodejs-mysql-orm-libraries> .

[12] Thakkar, R., 2022. "Mechanisms of Sequelize.js". DEV IT Journal. Available at: <https://www.blog.devitpl.com/sequelize/> .

[13] Schwartz, B., Zaitsev, P., Tkachenko, V., Zawodny, J., Lentz, A. and Balling, D., 2008. "High performance MySQL". 2nd ed. United States of America: O'Reilly, pp.1,2.

BÂLTAC Mihai-Cristian is a graduate of the Faculty of Economics Cybernetics, Statistics and Information at the Bucharest University of Economic Studies, bachelor's degree in Computer Science, mostly interested in web development and automation.