

IoT Management of Human Tasks at the Level of Smart Homes

Nicolae-Gabriel VASILESCU
Bucharest University of Economic Studies
gabriel.vasilescu@csie.ase.ro

This paper presents a prototype to help manage tasks in smart homes by connecting to the IoT network that allows the interconnection and communication between devices, robots, and objects inside the house to facilitate the activity carried out by the inhabitants and to emphasize the automation of processes, the secure use of at a distance and their achievement. Both technical details from the implementation are presented, as well as the visual appearance of the interface to help residents by using it as easily as possible, the processes that take place regarding the tasks are described, from when it starts and until the execution reaches the end.

Keywords: IoT, Automation, Human tasks, Smart homes, Security

DOI: 10.24818/issn14531305/29.2.2025.02

1 Introduction

The Internet of Things (IoT) is a system that connects several smart devices, objects, digital machines through an Internet network [1], which helps to reduce the human effort to perform manual tasks, automation being very current in solving tasks that take place in smart homes.

Applications for managing the tasks that take place in smart homes are very important in everyday life as they help to manage the time spent on a specific task or problem of interest over a long period as efficiently as possible.

They have been around since ancient times when people asked the question of how they can efficiently maximize their time and how they can be more productive to solve as many problems as possible in a proposed, preferable, possibly predictable time, and, also staged to divide into small parts and distribute them according to the capacity of each member of the family.

The question was raised from the beginning of how to create a context through which to realize an overview of all the responsibilities that exist at a given moment knowing that they will be completed to be carried out successfully.

Also, communication is an essential step that makes one realize that to achieve something tangible, discussions are needed regarding the devices and the IoT network that facilitate the degree of understanding of the tasks and,

eventually, to create a superior depth and as clear as possible of the problem.

It is desired through the prototype developed for the management of tasks at the level of smart homes to outline a scenario through which the details related to what has been done or what is to be done in the daily activity of smart homes are always known, where things are a little different because there are consistent differences from a component intelligence to another, and people are willing not to retain for an average period what changed today or what was thought about in the past and is going to achieve in a future period, for example, and in this way, if an agenda is created they remain the information somewhere recorded. Benefits are brought both from the point of view of a certain person's program and from the point of view of the house because it is possible to visualize the problems that may intervene or that have already appeared.

The prototype fulfils all these criteria and brings to the fore the management and monitoring of tasks in smart homes with secure access for the family members who live in that place and contributes to the efficient realization of the activities performed every day.

This environment has always been governed by innovative ideas to bring lightness and minimal effort to use to the forefront of all residents, but it is known that technology has a galloping pace, and major changes occur from

one day to the next, wanting adapting to all of this as easily as possible.

Through the problems that have arisen, a context is always created that modernizes what has been achieved so far and that brings new solutions for what has been found to be not good and is known to occur again in the future. Here it can be said that although there have been some discussions in the past to move to automating and reusing what was before in order not to waste time again to manage these inconveniences caused only by the fact that you have to create another task for a task identical to the one in the past or recently completed, not all applications respect these good manners that are the basis of an application.

2 Literature review

There are different tasks both inside the smart home and in the yard, such as cleaning, washing the dishes, home security, gardening, activities that many years ago required a lot of manual work performed by humans, but which are currently carried out with the help of automatic processes, remotely programmed robots. All these tasks need an IoT network capable of connecting all devices, robots, and other elements inside and outside the smart home.

The idea of the prototype started from the fact that the residents need to quickly manage what they must do in the next period, and they don't remember or remember what they wanted to achieve in the past and that's why an application is needed which provides this use case, namely that of managing task management.

Currently there are several popular applications to be able to manage problems and needs known and used in the current period [2].

These applications offer solutions to different problems, but basic because they aim to standardize them so they can be used by as many entities as possible and bring the same benefits to everyone.

Smart homes as it appears in [3] are described as a set of all the automated processes that facilitate the actions of the inhabitants using

various applications or devices that manipulate and control windows, lights, doors, etc.

The prototype starts from the premise that the user must have maximum use in a way that is as easy to manage as possible without wasting time to get familiar with giant applications. A brief retrospective was made in the initiation of this application also following what exists in the market now.

Another aspect for which the implementation of the prototype was wanted is that, according to the current trend, technology evolves and develops both the infrastructure and new programming languages to ease the work of the developer but also of the end user, especially at the level of use of resulting applications.

The task management application combines the degree of novelty of the technologies used with the friendly way of use given by the very easy-to-use web interface, by the persistence of the component parts of the application: both the logic part but also the visual part, of the stored code and especially of the fact that everything is responsive. The existence of some vital functionalities is needed that would automate the life cycle of a task.

This market of task management applications must bring novelty and above all a degree of ease of use to facilitate the daily activity especially in smart homes where it is desired to focus on data, on completed tasks, on the problems of the inhabitants and especially on their needs [4].

Also, the other applications described above also have some organizational problems, i.e. in the case of maintenance or changes to a functionality, they are unavailable for a few hours or even a day, and in these situations activities that are closely related to those changes are disrupted.

Related to the response mode to the client, it moves quite slowly, especially when you drag-and-drop from one state of a task to another. This was avoided by the created prototype and all the logic was implemented for the application to respond in a better time to maximize the time of the end user and gain in performance.

The important point in the application is the resident who, with a few clicks, must be able

to see the tasks instantly, to be able to see their evolution over time or, above all, to announce with a single button that the task has been completed.

As similarities with what exists on the current market and the prototype created, the fact that all these applications deal with the management of tasks and want the realization of the cycle of a task from the very beginning is respected.

As clear differences, the task management application brings a lot of responsive frameworks, new technology, graphs that are drawn in real time, notification when a task is ready and other competitive advantages related to the application's infrastructure.

3 Research Methodology

The prototype ecosystem consists of components that lead to the realization of a final form of the application because it is desired to group functionalities into small parts to highlight very clearly what a component wants to do, all of which have a single responsibility, and the great advantage of its reuse in different contexts in the future.

Many components have been used that help to draw on the main background a sequence that is modified in real time by the user and that does not influence the way of viewing when resizing the screen or the phone used in the management of the task management application from a smart home.

This ecosystem came to life with the idea of being used very easily by the inhabitants and every incorrect use movement brings to the fore exceptions and windows that outline very clearly what was entered incorrectly or does not respect the format required by the application to re-enter the desired information and then to recheck. Here the resident is also alerted when he does something right and the flow is allowed to continue through the life cycles of the tasks by displaying notifications that quickly disappear after reading.

Each user can connect and view their own situation as well as the general situation in the home at the IoT level, as each resident has a different context depending on the chores or tasks they must do.

Also, all this information flow encountered in the prototype is built for task management, and everyone will have access to the situation of the other inhabitants. One can see very clearly what is happening in the house very quickly and in a very efficient way.

The tasks are made in such a way as to be as easy to understand as possible during them but also after they are finished, being recorded in a record that helps to create some statistical forecasts to estimate what will happen in the future period and which will be the flow to continue with the activity object.

The prototype must provide an infrastructure as easy as possible and understandable by all to gain notoriety in the market of smart home task management applications, and therefore this system on which it is created brings reuse of code and components to solve tasks repetitive.

Through the connection with the Amazon server and the persistence of the data found and stored on GitHub [11], it is desired to access the application regardless of the area, the only condition being the connection to the Internet network to allow both the connection to the interface and to be able to respond to the requests or needs of the inhabitant using it to perform the full flow of specific actions.

Everything is done dynamically, all changes in the interface appear immediately after a request is executed, there is no component that has any static characteristics or depends exclusively on anything other than the Internet.

The prototype brings a degree of novelty, diversity through the different programming languages used and through all the services used and offered by it exposing the ideal development framework.

A friendlier environment is needed for this application and therefore it implements all the proposed functionalities and is based on the needs of the end users simplifying the whole process of managing and planning activities.

The idea of an application to manage the management of human tasks in the home comes as a solution to the problems of planning the future flow of activities that want to be implemented, especially from the desire not to miss the ideas that may appear, but also for the

effective planning for a period next.

The resident always has a minimal step of connecting to the application and each page is quite suggestive and brings very clearly the functionalities created to benefit him in his activity, all he must do is to ensure that he can start his actual work and endeavor necessary to work, passing the respective task into the "IN PROGRESS" state.

A task can have 5 main states: "CREATED", "ASSIGNED", "IN PROGRESS", "FAILED" and "DONE" depending on the stage it is at after being proposed by a resident.

During each state there are a lot of specific actions that the user needs to consider and that will greatly help him to complete this activity. You can see in figure 1 below the flow and the stages that a task can go through.

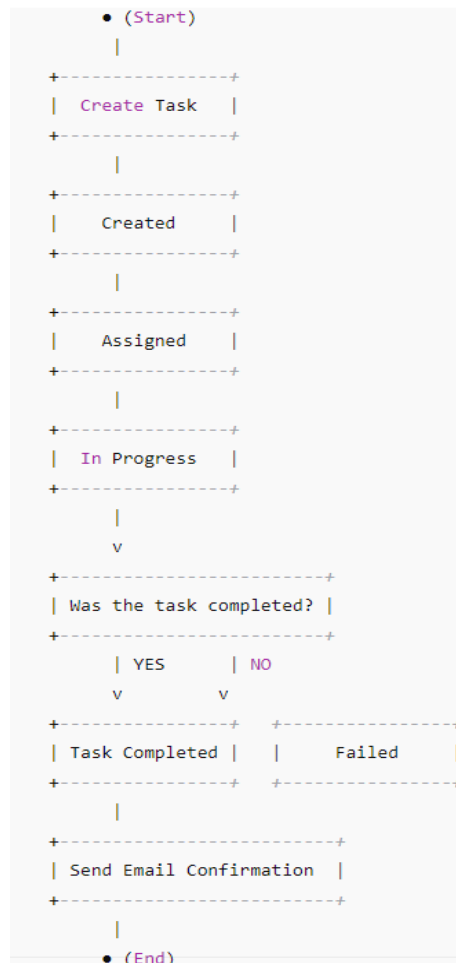


Fig. 1. Possible states of human tasks

When a task is created, it can be assigned by anyone, because it can be given to someone who could solve this task according to certain criteria.

When a task is assigned to a person, it will be present in the personal list that will appear on the own tasks page. From there the person assumes that they can solve that problem(s) and must estimate how long they can solve that problem. After this stage, it can start actual work and move the task to the in-progress

state.

When it is on the "in progress" status and after the actual work is finished, the result of the task is desired, which can be done or failed. If it has failed, it is a problem of achieving it and you can see what can be achieved, perhaps asking for help.

If the done state is reached, then it can be considered that the problem has been successfully solved and all residents are notified, receiving an email with the title of the task that has been

solved.

The proposed application is addressed to smart homes that want to have the activity organized from all points of view, having a planning process that is very easy to manage because you must know very well what problems or changes are to be made and then it can be used extremely quickly in the process of solving them.

This application gains in performance, real-time drawing of information even while it is being changed, plus new and innovative functionalities that automate the process of solving human tasks.

From the idea stage, the application proposed a high degree of performance and focusing on the residents, and for this reason an environment conducive to its activity on the platform was created, being very intuitive everything that happens in that space, without needing an understanding guide or any such document.

After the internal rules are established, the application is up and running to manage all the tasks of a resident and the whole process related to them, the user gaining a lot from the point of view of keeping information about what he did, what he is doing and what is next to achieve in the next period.

To concretize the aspects related to the performance of the application, 2 JavaScript frameworks are used, i.e. React.js to create the Web interface and Node.js to implement the functionality part.

React.js helps to create interfaces that bring an interactive contour to the users who use the application. It focuses on the pieces of code that have Single Responsibility, these are called components, essential parts and on which the entire infrastructure is based since the language was conceived. It layers all the components from the ones inside to the highest in the hierarchy, i.e. the parent that manages all the elements, shapes, icons by configuring them through an important part called state. By this state is meant all the attributes related to a certain component and which greatly impact what happens during its display on the screen. These attributes or elements can change depending on the parent or child element they are in via another important part

called props or properties. These props are properties that are inherited when a component is split, and only through them it can make changes to other specialized child components [5].

This framework also helps to efficiently display and update those states with very high performance, which gives its great strength. It makes the code easy to follow and debug if there are problems compiling or running a piece of it, or even all of it.

Quite complex UIs are formed through these encapsulated components that manage their own state, and because the logic is written in JavaScript, the state can be easily removed from the DOM, thus rich data can be transferred inside the application, the user being thus greatly helped in his activity [6].

Also, for state management between the different components, Real Time Go uses another Js framework, namely Redux. It helps a lot with its very consistent behavior when running on different environments such as client/server. Moreover, as a main advantage, the centralization of the application logic and states particularly allow modification, persistence, undo, and redo of the states very easily [7].

This framework even helps in sending complex reports to the server related to what is happening at the various states and points in time where the application is. It can be used at any level of the UI and has a wide variety of applications that can suit a programmer's needs to build something more different or innovative in today's Web application market.

For the server part, Node.js was used, a framework that brings to the fore the fact that it eliminates the problems related to the asynchronous part of things, an aspect that is predominantly found in JavaScript. It is based on some synchronous events and is designed so that scalable network applications can be created. On the synchronous side of things, as opposed to OS threads, the Node will wait until certain operations are done to move on to the next step, doing other operations that don't wait for other threads.

Users who use Node.js can't have problems with process deadlocks because they don't

exist, there is no function to perform direct I/O, so these processes can never deadlock. This makes scalable systems quite reasonable to develop in Node.

As a system, Node.js is very similar to Ruby and Python, except that it picks up the event model a bit faster. Events are performed in loops instead of libraries as in the case of the 2 mentioned above [8].

HTTP is a major pawn in shaping the requests that Node.js makes because it creates a conducive framework for a Web client. Since there are no threads in Node, that doesn't mean you can't take advantage of multiple child cores. It allows the use of sockets to interact between processes, resulting in the balancing of activities within them.

To use persistence, Node.js can interact very well with databases, Mongo DB, MySQL or any other using a dedicated ORM, namely Sequelize, which takes care of all the connection between the server and the database.

All entities are automatically created if they do not exist at a given moment, and the update, select or other CRUD operations are performed by only one line of code, which gives great ease when implementing the actual requirements of the application proposed to be developed since its initialization phase.

By intercorrelating all these aspects and using the 2 JavaScript frameworks, the end user is given all the favorable conditions to be able to estimate and note everything he will have to undertake in the next period.

The client can view the changes that appear on the screen instantly, in real time, without wasting a few seconds as happens in the case of established and old interfaces when the execution time can reach a few seconds.

The React - Node connection and the addition of Redux and Sequelize provide security and speed in data transmission without confusing the process of programming tasks and their execution in any way.

For the data persistence part, the prototype uses the MySQL database, which achieves high levels of scalability, has a high degree of

security, offers a very good response time and reliability, whose connection to the server was made with an ORM, Sequelize, which makes working with databases quite easy.

At present, this database management system, MySQL, is the most popular open source used for this purpose at the present time, which integrates very well with any operating system, especially Linux, and through which you can build highly complex applications using any major known programming language [9].

Sequelize creates all the entities very quickly without having to build SQL scripts to do all these things, which is the main deal used. It handles everything related to the relationships between entities, the mapping between the elements of the classes defined on the server and those in the database, and all the CRUD operations that are performed [10].

The rate of development of an application is greatly accelerated because it is gained in the database chapter, and all manipulation of the tables can be done directly from the code. Thus, the developer no longer needs to ensure that he has not respected a certain relationship or that integrity exceptions have happened/thrown.

Sequelize comes with a manual that specifies all the necessary information starting from the basics, such as those to declare and validate the attributes of a class when they are mapped to the database.

This server-sequelize-mysql link is done without impacting the application's activity in any way and behaves like when CRUD operations are performed but being hidden in the server logs.

The application server is written in Node.js, the JavaScript framework, and all the functionality implementation and its core can be found there. From here all processes are launched and all routes are exposed to be received by the Web client. By creating all the functionalities on the back-end side, the effective serving is reached through certain exposed endpoints. In figure 2 below you can see some used and exposed endpoints.

```
const router = express.Router()

router.route('/signin').post(authController.signin)
router.route('/signup').post([authController.signup])

router.route('/mytasks').post(tasksController.mytasks)
router.route('/create_task').post(tasksController.create_task)
router.route('/my_created_tasks').post(tasksController.my_created_tasks)
router.route('/assign_task').post(tasksController.assign_task)
router.route('/change_task_status').post(tasksController.change_status)
```

Fig. 2. Exposed endpoints

Through all these routes the actual connection with the React part is made and thus the great benefit of very fast backend-frontend communication is realized, these 2 essential parts of the application bringing increased performance to the prototype.

To greatly strengthen the speed of the user to realize the data entered in the wrong format, validations are built especially on the server side, but also on the client.

Everything related to the interface was made using the React.js framework that allows very friendly interaction with the end user and that draws without influencing the state or size of other components in real time. The advantages of this relatively new framework brought only pluses to the developed application and put the ease of use of all developed applications in the foreground.

It allows users to declaratively describe the interface and be able to model all the states that occur during these interfaces. This means that developers describe all existing interfaces as a final state, and this framework models them according to context and state.

Reusing components, states and interfaces helps a lot in the time a developer spends on development and by using states and props, the advantages come in large numbers as you can manipulate data very easily depending on the current context or what is to take place.

The same component can be used in multiple user interfaces, with components containing other components, each influencing the state of the other based on input from a user at a given time.

A component can still have a private state that can contain data that can change over time and cannot be modified by other components outside of it.

To manage all this application flow very well, it used GitHub as a persistence method, the most known place to save the code. This is where development teams can create processes to review and improve the quality of code. The best way to do pair programming, this environment is perfect for communicating the changes made in the project because you can quickly see in the History/Commits tab which developer made a change, when and what it consisted of [11].

GitHub is perfect for any environment because it can be cloned on any operating system without any problem.

To have public access to the developed interface, the entire application is uploaded to Amazon, an environment that provides storage services for different projects. In connection with the space offered, data is also stored at petabyte level, an enormous amount these days. It offers backup services for applications uploaded there in the cloud and public IPs that can be accessed from anywhere in the world [12].

To be able to switch between the two environments, the test environment and the public one on Amazon, the GitHub had a major contribution in this regard because through a single line of code the code written on a Windows environment is cloned on a Linux environment.

Also, the Amazon server comes bundled with

a MySQL server, where only the created schema must exist on this client, and Sequelize will take care of the rest.

This close connection between the Amazon server and the GitHub side helps enormously in the application life cycle because when bugs appear, the transition to the dev/test environment must be made, where pieces of code from the application are added or changed, git being an important pawn here as it shows all changes including added/deleted/changed lines of code so you know in the future what changed.

The application has a complex architecture, but at the same time quite easy to understand because both on the Node.js side and on the client side all components and classes are divided into packages with very suggestive names, thus making any developer understand what has been worked in the case of creating new functionality or changing pieces of code in this regard.

On this whole part of the infrastructure, the application offers many advantages because each piece of written code only deals with a small part of the functionality, which is a plus since a small or large update does not inconvenience the application in any way, having each one responsibility.

Real Time Go is a SPA, Single Page Application, i.e. a web application that dynamically interacts with its users by rewriting the page according to the functionality provided by the server and requested by the client [13].

Both the server side and the client side interact very well with each other and are connected using axios, a promise based on the HTTP client on the browser side but also on Node.js.

This bridge between the two parties makes it possible to wait in real time for some events while others are executing to put together all

the information to send as quickly as possible to the client [14].

A web application can be said to be a collection of interrelated web pages that provide different functionality for a customer to get the best out of their business by following a well-planned plan. These users communicate through the UI interface to enter the information needed to save all the data in this task storage process.

Asynchronous HTTP requests are initiated through the so-called front-end that are exposed to the server that will come with a quick response that will display or record the data provided. The Web interface helps a lot with the flexibility offered to both the end user and the developer as one can track the evolution of the application over time, the number of users and other performance aspects.

Through the component diagram, all the component parts of the application that participate in solving the proposed problem are brought to the fore, both the database, the back end and the front-end, all of which form the core that provides everything that appears in the description related to the functionalities.

Figure 3 shows all the components used in the application, starting with the database, all the scripts being generated by Sequelize, the entities with all the attributes and characteristics being created in the Application Repository. On the line with the services appear all the functionalities that help to realize the entire flow, and through the controllers all the functions found in the repositories are used, being used to expose endpoints used by the Web client.

As an overview, Figure 3 summarizes, by visualizing the components used, the actual flow of the application.

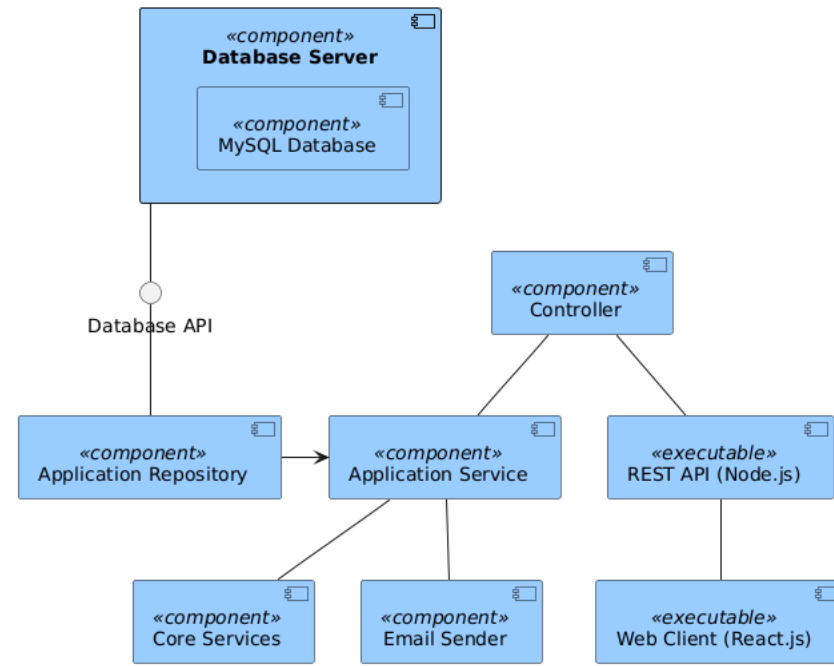


Fig. 3. Component diagram

This application flow starts from the moment the user creates an account in the application and ends when the user logs out from here. He can carry out different activities while maintaining the logged in state.

A resident can create tasks for himself or have tasks created for him, and in the latter case assign a certain user to a set task.

After the moment of taking it over and verifying the capabilities necessary for its completion, the work on it begins by moving into the "in progress" state. Here a user can estimate how long this whole process will take.

In the end there are two possibilities, such as that everything ends successfully and then it has the done state or no success and then it has the failed state. If everything goes according to the estimate and the task ends successfully then all residents are notified that it has ended successfully.

A rather important aspect is that of the possibility to view different graphs regarding the activity carried out by the person who is connected to this dashboard.

Also, throughout maintaining the connected status, when refreshing the page, the respective page is kept without being redirected to log in to reconnect.

Exceptions are thrown when something doesn't work or the server can't respond

immediately, and the user can instantly realize the problem even if this happens very rarely. All these models, but especially their methods, are further used in the controller modules where the endpoints are exposed to the client to be able to access them and solve their problems.

An important role is also played by the statuses that help both the programmer and the user to realize whether the dashboard is working at the desired capacity.

Every time it is checked if there is something in the database and it is different from null and then the data obtained from MySQL is manipulated. Every time a status code is sent to the client to be able to understand what happened to his query or request. the prototype uses the following codes:

- 200, in the case of a successfully completed operation, from class 2.
- 400, when there is a query or insertion problem in the database related to the format or other aspects regarding the respective objects used, from class 4.
- 500, when there are problems caused by the server, such as the fact that the connection hangs or some methods there, in class 5, cannot be accessed.

All the logic is in the server and that's why the actual core is done here, and minor checks are

done in the interface when the call itself is made because there is already testing on the server side, sending different statuses each time from which produces the expected result. To make it possible to draw the assigned tasks in real time, the prototype uses web sockets, and by means of some callbacks it is notified in real time if any task has been sent, and if so, it is drawn when is saved in the database.

The interface is the overall picture that users see when they connect to the application, it is made with the help of and using the React framework as a set of functionalities for customers to benefit from.

Everything is very interactive; components redraw very quickly, and everything is done in real time. It starts by creating an account to

use the application. This is done through a new form when pressing the sign-up button by filling in the email, the username and by entering the password twice to check if there is a match between the two. Here there is also the functionality to see the password by pressing the buttons on the right.

Once registered, the user is redirected to the log in page, where after entering the username and password, he enters the application.

The application itself allows the credentials to be retained in the browser cache until the actual logout.

The application flow starts on the task management tab, where residents can create a task by pressing the icon on the top right. Figure 4 highlights the task creation activity.

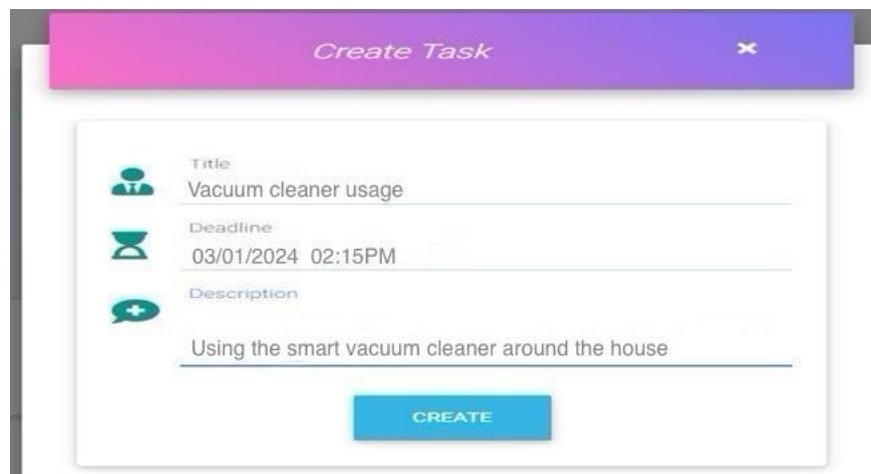


Fig. 4. Task creation activity

At this moment, depending on the status of the task, it can be set to progress and then to fail or done as it can be seen in Figure 5. A task in the assigned state cannot directly reach the done state.

If it is done successfully and everything is fine, when the done status is selected, the task ends, all team members are automatically notified using the service created to send mails.

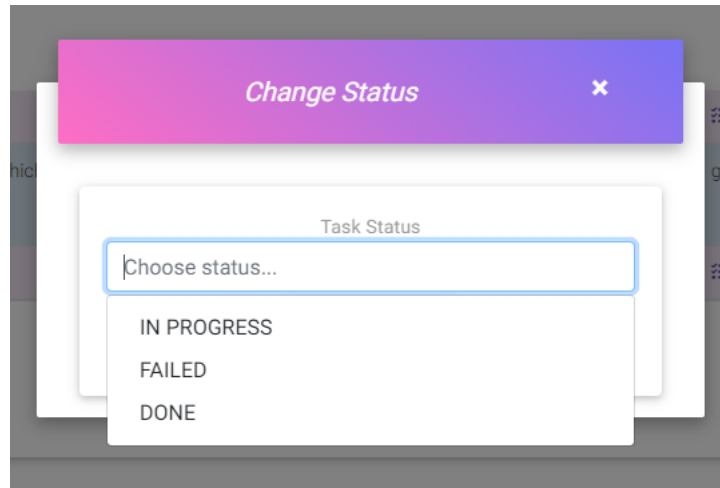


Fig. 5. Task statuses

To have the widest possible picture of the status of the tasks according to the current user, on the Reports branch real-time graphs are created with all the tasks according to the different states in which they are.

As can be seen in figure 6, these statistics appear for the months of the year, and when you click next to each month, the tasks at that time in the 4 states will be counted.

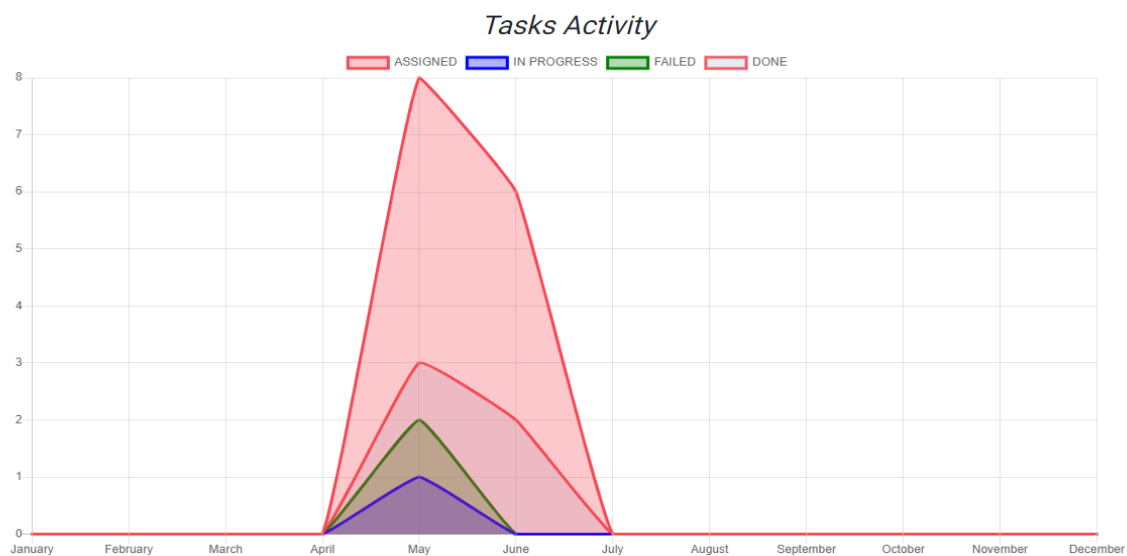


Fig. 6. Human Tasks chart

For the activity in the smart home to be as interactive as possible, the prototype also allows the writing of comments for the respective human tasks.

Figure 7 shows that any resident can contribute to the completion of a task by adding new comments.



Fig. 7. Add comments

Also, a task can be edited if it was created incorrectly. This is done very easily just by changing the respective fields.

For everything to happen in real time, when a task is given to another inhabitant, it appears instantly in the list of own tasks.

The prototype provides a notification manager when an activity is successful or not to alert residents whether they have entered correct data into the application.

To help in the management and states of human tasks, the application uses RxJS, a library that uses observers, which facilitates the composition of asynchronous callbacks [15].

At the architectural level, the application communicates with the user through the graphical interface (frontend), and all functionality is carried out at the services level on the backend side, with all entities being mapped to the database level, as can be seen in figure 8.

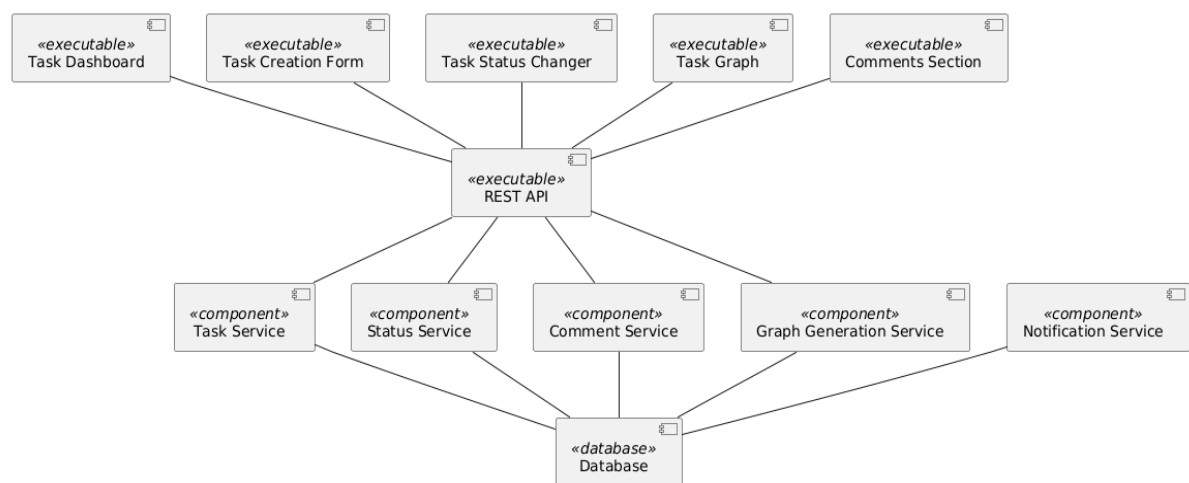


Fig. 8. Task management architecture

The data obtained from the sensors located in the smart home, also based on the work [16], can be processed in such a way as to very well identify the actions that can be automated based on the identified daily activities.

Starting from [17], new challenges arise every day, considering that technology evolves so that identifying the needs of residents is the basis of the other stages so that the tasks performed in smart homes at the management level can be categorized and reduced through automation, thus improving the quality of life.

Smart home security is influenced by how human tasks are managed and starting from [18], and people are reluctant to pass as many tasks as possible in an automatic way through various automated technological processes. There are different platforms that document existing security issues and even identify new versions that fix them.

As presented in [19], the IoT architecture is divided into several layers, such as application, communication, perception, and all of this form the entire network that is found in different security states, depending also on the

behavior in the daily activities carried out by the inhabitants.

Including the prediction part of some events can be handled in an IoT network as reported in [20], and the management of the tasks carried out should not be impacted, while also avoiding various dangers that may arise both based on older technology and influenced by other ongoing actions.

4 Future Work

The application offers a lot of benefits, but surely in the future it can offer even more as the market of task planning dashboards is dynamic and constantly moving, which shows the fact that residents want the automation of all repetitive daily processes as much as possible.

A first aspect introduced will be the log in with face-recognition that can make it easier to enter the application without having to enter the credentials every time to enter the created account.

This feature would enormously improve the quality of such an application and lead to the market introduction of AI technologies within the developed application. It can be considered a big plus compared to the rest of the interfaces.

Also, there will be a lot of emphasis on process automation, as there is the automatic sending of emails, which is proving to be a very often undertaken activity.

Another aspect to modify can be considered detailing the tasks, which will make it possible to visualize the requirements, residents being able to write on a task if they encounter problems or if they need additional resources to finish it.

A final aspect to improve would be that of the connectivity between users, i.e. the integration with Google Maps to view in real time where the other inhabitants are placed, in the respective city or country, to know the distance between them.

5 Conclusions

The application offers an interactive web interface where residents can do a lot of actions because it offers a lot of functionality

achieved through new frameworks that come to the aid of the end customer who uses it to be able to plan their daily activity inside a smart home.

Time is important and especially the degree of estimation of some actions that can be divided into quite small pieces, each individual human task.

Also, automation should play a very important role in family life because there are repetitive actions at every step that require time to solve. If it follows this trend, it comes to realize that it will have to put more emphasis on the core of a task, i.e. on its solution, but not on the ways of transmitting its stages.

An application of this type is very useful because there are still times when ideas and problems arise to be solved within a smart home and once noted they will remain saved in the database, and in the following period it will be determined who oversees solving it in the shortest possible time and understand very well what needs to be done.

The prototype comes as a necessity to always know what the situation of a smart home at a given moment, what happened in the last period, and here it is the graphic representation, and to predict, starting from the current situation, the future actions that follow to be undertaken within the dwellings.

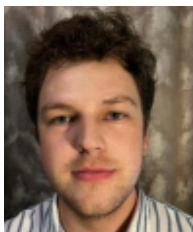
Therefore, tasks are indispensable in our life, especially the everyday one, at home, where human tasks appear and which, to be solved, require a phasing that helps the responsible person outline his activity in small but sure steps.

References

- [1] A.A. Laghari, K. Wu, R.A. Laghari. et al., "A Review and State of Art of Internet of Things (IoT)," in *Arch Computat Methods Eng*, vol 29, 2022, pp. 1395–1413.
- [2] Future US, Inc., „Best Task Management App,” Future US, Inc. Available: <https://www.techradar.com/news/best-task-management-app>.
- [3] L. Biljana, S. Risteski, V. Kire and Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," in *Journal of Cleaner Production*, vol 140, 2017,

pp. 1454-1464.

- [4] Process Street, „Introduction to Task Management,” Process Street. Available: <https://www.process.st/introduction-to-task-management/>.
- [5] FreeCodeCamp, „React is taking over front-end development,” FreeCodeCamp. Available: <https://www.freecodecamp.org/news/yes-react-is-taking-over-front-end-development-the-question-is-why-40837af8ab76/>.
- [6] Facebook Inc., „A JavaScript library for building user interfaces,” Facebook Inc. Available: <https://reactjs.org/>.
- [7] D. Abramov, „A predictable state container for JavaScript apps,” Dan Abramov. Available: <https://redux.js.org/>.
- [8] Node.js Foundation, „About Node.js,” Node.js Foundation. Available: <https://nodejs.org/en/about/>.
- [9] Oracle Corporation, „MySQL Enterprise,” Oracle Corporation. Available: <https://www.mysql.com/>.
- [10] Sequelize INC., „Manual Sequelize,” Sequelize INC. Available: <http://docs.sequelizejs.com/>.
- [11] GitHub, „Write better code,” GitHub. Available: <https://github.com/features/code-review>.
- [12] Amazon Web Services, „Documentation,” Amazon Web Services. Available: <https://aws.amazon.com/>.
- [13] Wikimedia Foundation, Inc., „Wiki-media Foundation, Inc.,” Single-page application. Available: https://en.wikipedia.org/wiki/Single-page_application.
- [14] GitHub, Inc., „Promise based HTTP client for the browser and node.js,” GitHub, Inc. Available: <https://github.com/axios/axios>.
- [15] Apache-2.0 License, „Reactive Extensions Library for JavaScript,” Apache-2.0 License. Available: <https://rxjs-dev.firebaseapp.com/>.
- [16] L. Babangida, T. Perumal, N. Mustapha, R. Yaakob, „Internet of Things (IoT) based activity recognition strategies in smart homes: A review”. *IEEE Sensors Journal*, 22(9), 2022, 8327-8336.
- [17] D. Bouchabou, S. M. Nguyen, C. Lohr, B. LeDuc, I. Kanellos. „A survey of human activity recognition in smart homes based on IoT sensors algorithms: Taxonomies, challenges, and opportunities with deep learning.”. *Sensors*, 21(18), 2021, 6037.
- [18] H. Touqeer, S. Zaman, R. Amin, M. Hussain, F. Al-Turjman, M. Bilal. „Smart home security: challenges, issues and solutions at different IoT layers”. *The Journal of Supercomputing*, 77(12), 2021, 14053-14089.
- [19] P. Franco, J. M. Martinez, Y. C. Kim, M. A. Ahmed. „IoT based approach for load monitoring and activity recognition in smart homes”. *IEEE Access*, 9, 2021, 45325-45339.
- [20] O. Taiwo, A. E. Ezugwu. „Internet of things-based intelligent smart home control system”. *Security and Communication Networks*, 2021(1), 2021, 9928254



Nicolae Gabriel VASILESCU has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2019. In 2021 he has graduated the IT&C Security Master program at the Bucharest University of Economic Studies and starting from 2021 he is a PhD student in Doctoral School of Economic Informatics at the Bucharest University of Economic Studies. Currently he works as Java Developer at Cegeka Romania, Bucharest. He is interested in Java programming, new technologies and IoT security.