Organizer

Project Management, Task Administration, and Team Collaboration Application

Developed by Carlos Dominguez

July 2024

Table of Content

Challenges and Solutions

- Keep it Simple
- User Experience

Code Highlights

- Structure, Organization, and Readability
- Validation of Input Data
- Compact and Clear Method Definitions

Use of Policies

- Flexibility and Centralized Permission

Use of Vue.js

- Enhancing User Interactivity
- Key Components and Functionalities
    - Updating Task Status
    - To-Do Lists
    - Time Tracker

Conclusion

- Summary
- Future Prospects
- Final thoughts

Organizer

Overview:

The "Organizer" project management, task administration, and team collaboration application is developed using PHP (Laravel) and Vue.js. This project aims to enhance team productivity, streamline project tracking, and facilitate efficient task management. It offers a comprehensive platform for various team and project needs, integrating features such as calendar management, time tracking, task administration, and project oversight. The platform serves as a powerful tool for team collaboration and resource management.

Background:

My experience in developing project management applications spans several years and multiple iterations, driven by my interest as a developer and my understanding of team collaboration needs. While working as a FIeld Service Telecommunication Engineer, in charge of field service teams, supervisors, working in collaboration with colleague engineers, and reporting to my immediate manager, I recognized the necessity for a unified tool to coordinate activities, collect data, and maintain a history of completed tasks. This need prompted me to develop an application to enhance work efficiency, facilitate collaboration, and keep records of company resource usage. The development of this application evolved alongside my career as a Telecommunication Engineer.

- **Task Assignments Application at Ufinet Guatemala (2009 to 2011):** I created a simple task manager platform to coordinate workforce activities. This version focused on reporting daily activities and recording periodic maintenance duties in the network. It enabled teams to report activities efficiently through user-friendly interfaces, despite their limited experience with computer systems.
- **Project Management Software at Tigo Guatemala (2011 to 2014):** Developed for the Operation and Maintenance division, this software improved task coordination for up to 150 team members, reducing response times and enhancing data generation for efficiency analysis. It included functionalities for managing teams, projects, tasks, and radio base maintenance reports, with features for task history, team calendars, inventory management, and file organization. Tasks were classified into categories such as maintenance, improvements, installations, and corrective tasks.
- **Project Manager Application at Ufinet Guatemala (2017 to 2023):** This enhanced version, used by field service groups in southern Mexico and western Guatemala, was optimized for SQL queries and PHP code. It included inventory management, tracking materials and devices by serial numbers, and maintaining historical records of stock movements. The application also featured network site profiles, team tasks and calendars, and comprehensive project management.

I would like to emphasize that, although my role was a Telecommunications Engineer, I had a strong personal interest in developing applications. Despite making an application that was not part of my official job responsibilities, I dedicated my personal time to create these applications.

By leveraging my professional insights and incorporating feedback from team members, I aimed to enhance the efficiency and coordination of our work processes.

Due to data privacy policies and regulations of the companies I worked for, I was unable to retain copies of the code developed during these projects. Nevertheless, the insights and expertise gained from these experiences have helped me develop a logical programming mindset, with a strong focus on data management and the importance of user-friendly interfaces.

Following these principles, I developed Organizer from scratch using Laravel. Although still under development, the project aims to create a versatile platform capable of supporting personalized applications. This endeavor has enhanced my proficiency in modern programming techniques, methodologies, and standards, reflecting my dedication to continuous improvement and professional growth.

Team Organizer built in Laravel

Technologies Used: Laravel Framework, Vue.js, and Bootstrap library.

Organizer offers comprehensive functionalities to manage projects, track tasks, and facilitate team collaboration. Users can create and track projects, manage tasks and their time investments, handle documents and files across the platform, focussing on sharing information, tasks and duties in a collaborative way.

Use Case: Understanding the importance of a robust software solution for coordinating teams, managing activities, and controlling tasks and projects, I chose to build the platform in Laravel, due to its elegant syntax, comprehensive set of tools, and robust features, providing a strong foundation for developing scalable and maintainable applications making it a flexible and future-proof choice for evolving project requirements.

Key Features Include

Project Management:
Effortlessly create, track, and oversee multiple projects, ensuring alignment with goals and deadlines across teams.

Task Administration:
Assign, monitor, and manage tasks with precision, balancing workloads and meeting deadlines efficiently.

Team Collaboration:

Foster seamless collaboration through shared task lists, file attachments, and integrated calendar views. Profiles allow for dynamic interaction among team members and clients.

Customizability and Integration:

Benefit from a user-friendly interface that can be customized to fit specific needs and integrated with other applications as required.

Client Section:

Provide clients with access to relevant project information, enhancing transparency and communication.

Time Tracking:

Provides tools for tracking time spent on tasks and projects, helping manage workloads and optimize productivity, allowing to display reports of each collaborator

Document sharing:

Supports attaching and sharing files, images, and documents related to tasks and projects, centralizing crucial information.

Profiles:

Extends user profiles for team members and clients, making a distinction of a project collaborators, and project clients.

Roles and Permissions:

Define roles such as collaborators, team leaders, and project owners, each with different permissions (edit, create, delete). Currently, the platform uses basic permission logic, but Laravel policies allow for easy modifications as needed.

User Dashboard:

The User Dashboard provides a comprehensive view of tasks and projects, consolidating key information into a single page. It displays task statuses and a to-do list, allowing users to efficiently monitor their workload and priorities for the day. This centralized view simplifies task management and helps users stay organized by presenting all relevant information

Calendars:

Calendars are displayed in the collaborator dashboard and projects.

Planned Integrations:

AI Assistants:

Integration with AI tools to provide smart task management and automated assistance.

Google Calendar:

Syncing with Google Calendar for seamless scheduling and time management.

Clockify:

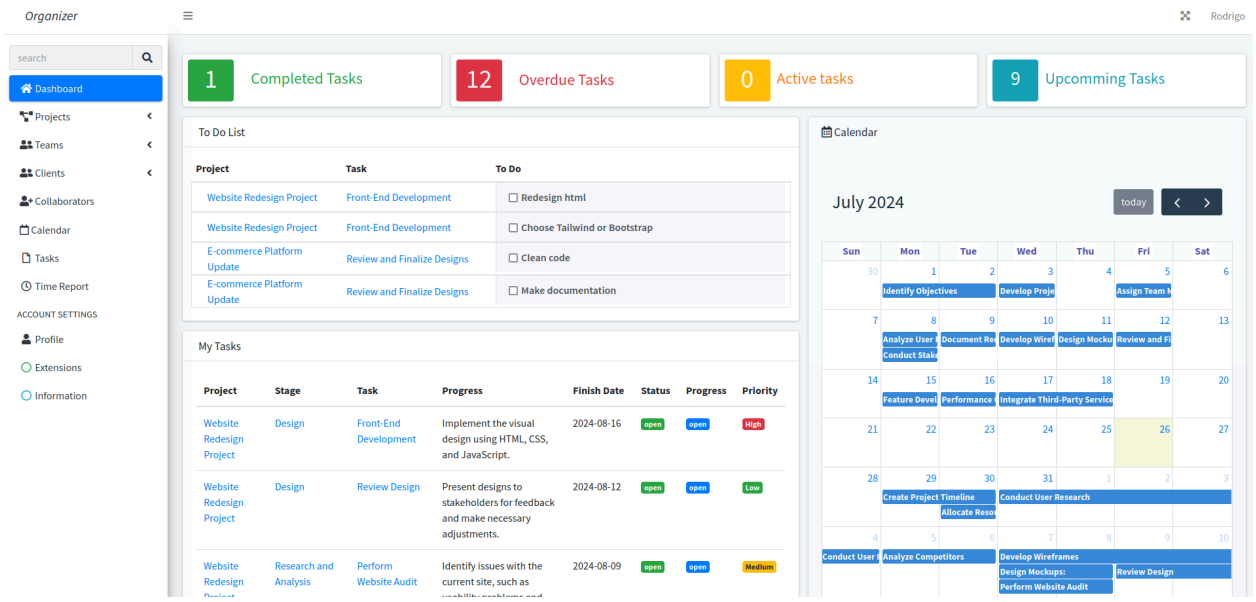
Integration with Clockify for advanced time tracking and reporting features.

Structure and Design

The application's structure emphasizes functional efficiency and streamlined operations. Each component is designed to facilitate smooth task management and project coordination, ensuring that users can quickly access and manage their workflows. The layout supports intuitive navigation and effective functionality, prioritizing practical use over aesthetic design to enhance overall productivity and ease of use.

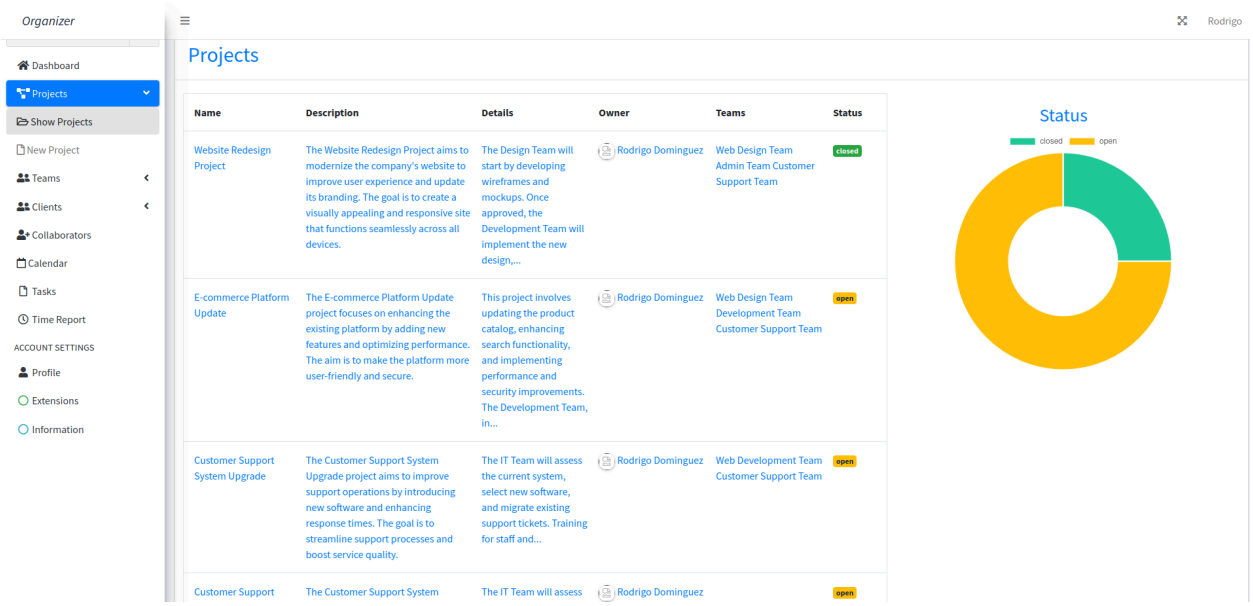
Dashboard:

The dashboard centralizes task and project management with a **Calendar** for scheduling, a **To-Do List** for daily priorities, and an **Open Task List** for tracking ongoing work. These features provide a clear overview of deadlines and tasks, helping users stay organized and focused.



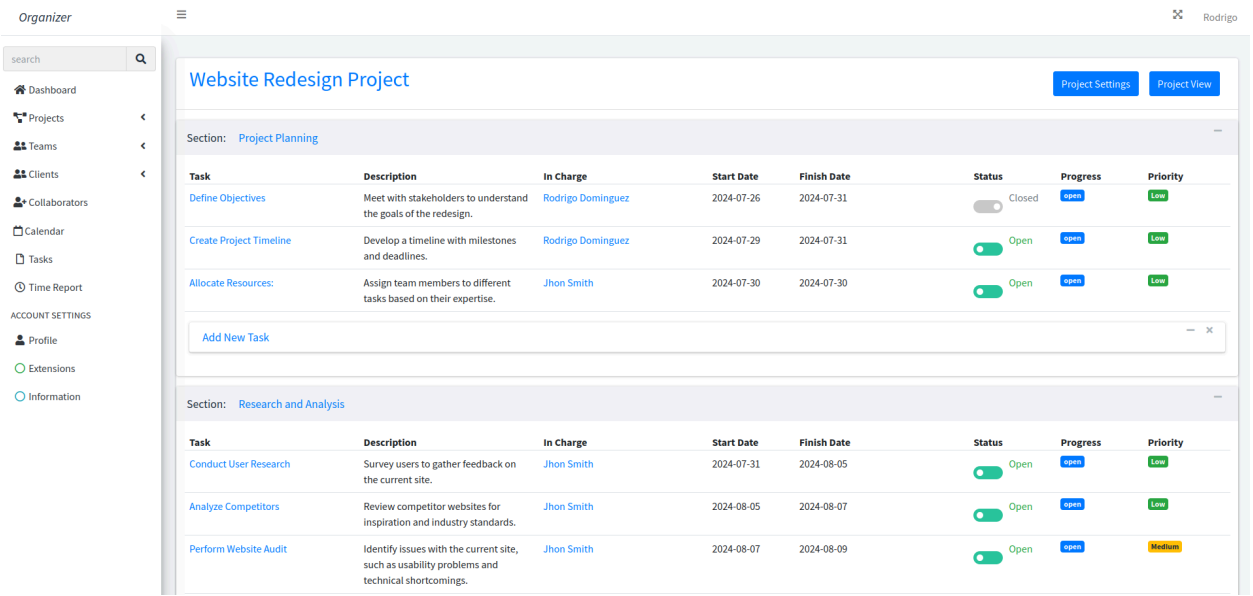
Projects:

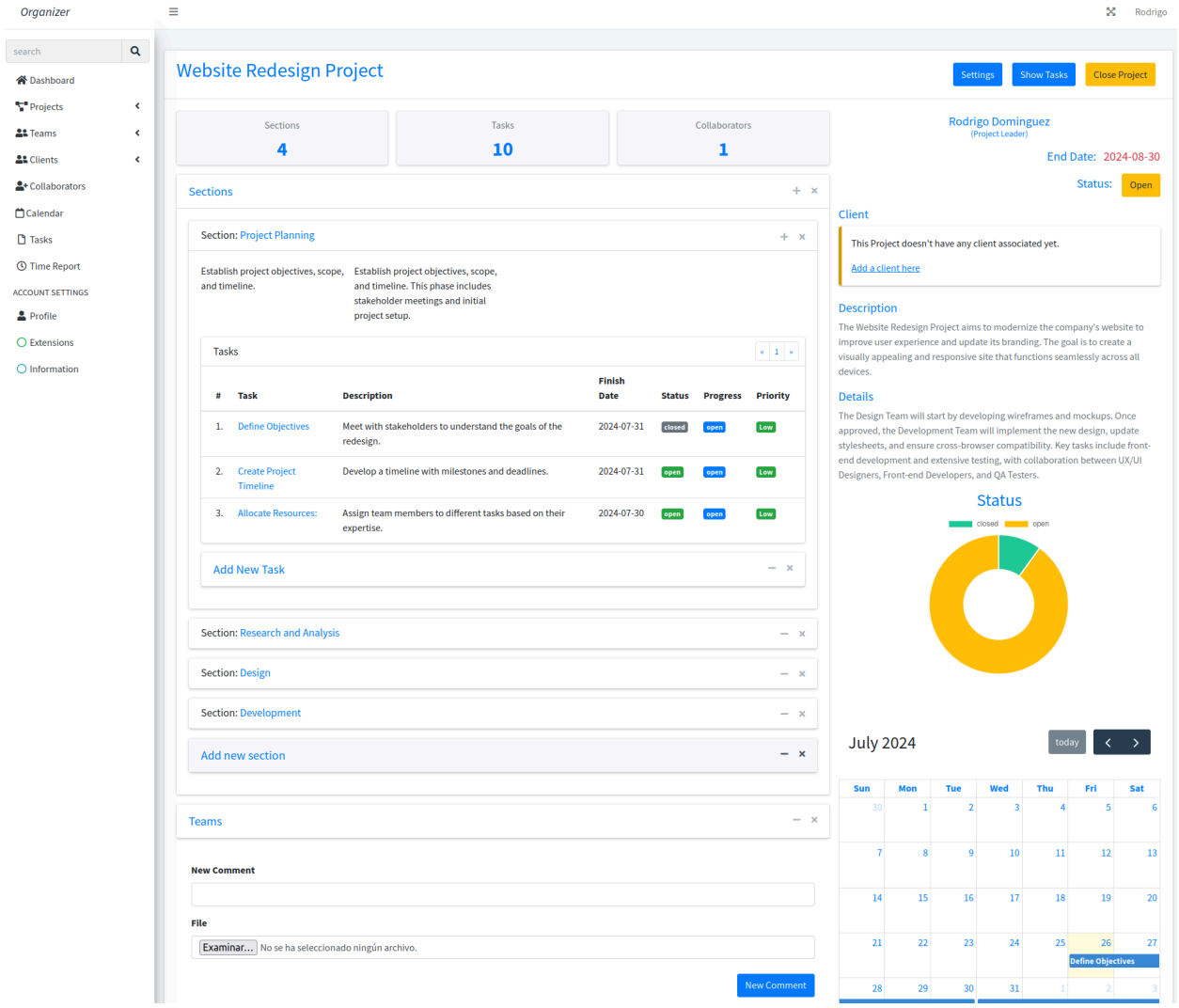
The Project List provides an overview of active projects that the collaborator is participating in. It enables users to quickly view project details, and monitor their progress. This centralized view helps in efficiently tracking project status and accessing key information



Project layouts:

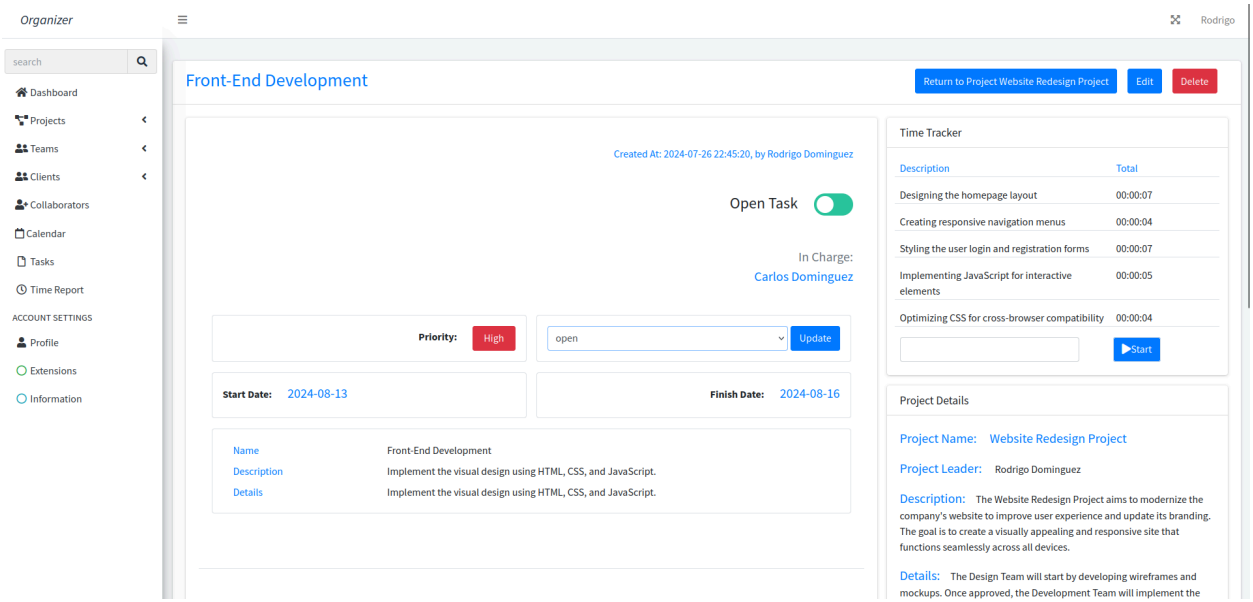
The **Project Layouts** section provides two distinct views to enhance usability. The first layout presents tasks and stages in a list format, making it easy to manage and track individual items. The second layout offers a comprehensive overview, including project details, comments, files, stages, task status, and general information, for a clear view of the project's progress and components.




Organizer
Rodrigo
search
Dashboard
Projects
Teams
Clients
Collaborators
Calendar
Tasks
Time Report
ACCOUNT SETTINGS
Profile
Extensions
Information
Website Redesign Project
Settings
Show Tasks
Close Project
Sections
4
Tasks
10
Collaborators
1
Rodrigo Dominguez
(Project Leader)
End Date: 2024-08-30
Status:
Open
Client
This Project doesn't have any client associated yet.
Add a client here
Description
The Website Redesign Project aims to modernize the company's website to improve user experience and update its branding. The goal is to create a visually appealing and responsive site that functions seamlessly across all devices.
Details
The Design Team will start by developing wireframes and mockups. Once approved, the Development Team will implement the new design, update stylesheets, and ensure cross-browser compatibility. Key tasks include front-end development and extensive testing, with collaboration between UX/UI Designers, Front-end Developers, and QA Testers.
Sections
Section: Project Planning
Establish project objectives, scope, and timeline.
Establish project objectives, scope, and timeline. This phase includes stakeholder meetings and initial project setup.
Tasks
Task Description Finish Date Status Progress Priority
1. Define Objectives Meet with stakeholders to understand the goals of the redesign. 2024-07-31 closed open Low
2. Create Project Timeline Develop a timeline with milestones and deadlines. 2024-07-31 open open Low
3. Allocate Resources: Assign team members to different tasks based on their expertise. 2024-07-30 open open Low
Add New Task
Section: Research and Analysis
Section: Design
Section: Development
Add new section
Status
closed
open
July 2024
today
Sun Mon Tue Wed Thu Fri Sat
Define Objectives
Teams
New Comment
File
Examinar...
No se ha seleccionado ningún archivo.
New Comment

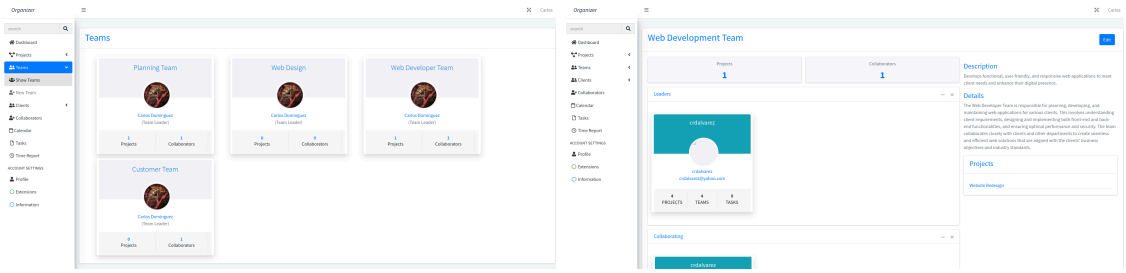
Tasks:

The **Tasks** section provides a comprehensive overview of all tasks associated with projects. It allows users to view, manage, and track the status of tasks, including their deadlines, priorities, and assigned team members. This section helps in organizing work efficiently, monitoring progress, time reports, make comments, upload files and update task status.

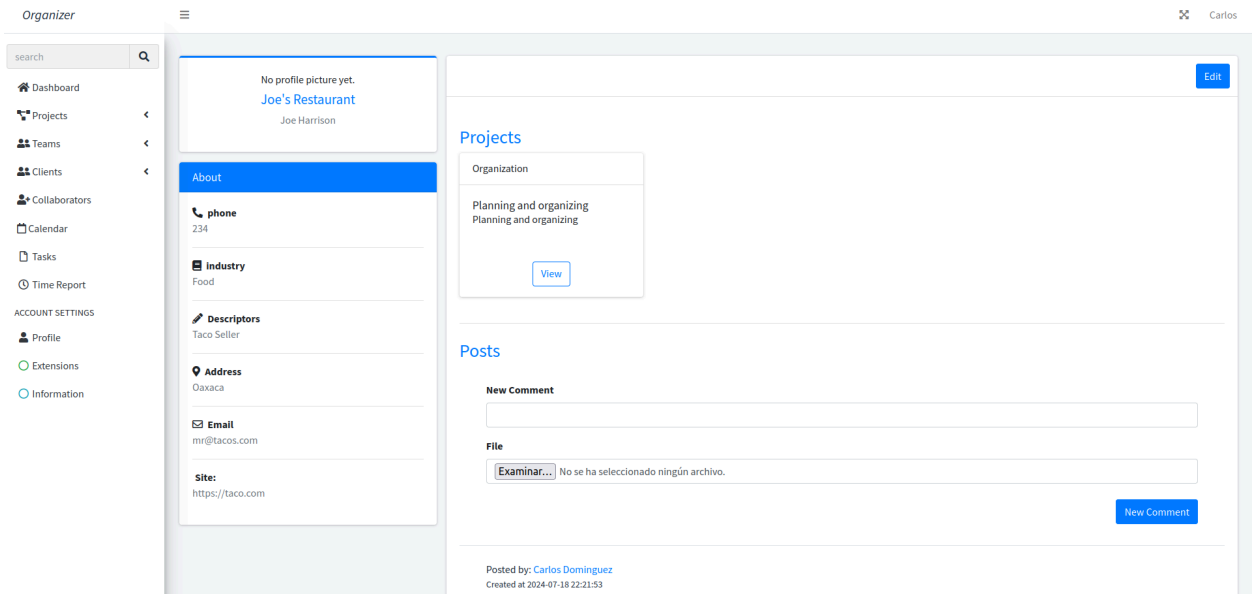


Teams:

The **Teams** section allows the creation and management of team members. This section is essential for tracking team activities, coordinating efforts, and ensuring that all team members are aligned with project goals.



Clients:
The **Clients** section offers a centralized view of all client-related information. It allows users to access client profiles, and manage client-specific details, view clients projects, and Allows to post comments and share files.



Challenges and Solutions

Challenges

Keep it Simple:

One of the key challenges I encountered was to ensure the application included only the necessary functionalities, making it scalable and adaptable without introducing a bias toward a specific use case. By focusing on simplicity and core features, the platform remains flexible, allowing for future customization to meet diverse business needs. This approach ensures that the application can be easily extended and adapted for various purposes without being constrained by an overly complex or narrowly focused initial design

User Experience:

Creating an intuitive and user-friendly interface that could accommodate various use cases without becoming overwhelming involves designing a layout that is clear and accessible. This means implementing a navigation structure that is easy to understand, using consistent visual elements, and ensuring that users can efficiently access all necessary features. The interface should be adaptable to different tasks and workflows while maintaining simplicity and avoiding clutter, thus supporting a wide range of user needs without compromising ease of use.

Code Highlights

Structure, Organization, and Readability:

Clean, well-organized, and readable code is a priority, to ensure maintainability and scalability. Adhering to best practices, I employed a consistent naming convention for routes and controllers, which enhances the clarity and predictability of the codebase. Each route and method is named descriptively to reflect its purpose, facilitating easier navigation and understanding for future collaborators.

I structured the code into clearly defined sections, grouping related routes and functionalities together. This organization improves the overall readability of the file and streamlines the process of locating and modifying specific routes or methods. By keeping the code well-documented and self-explanatory, I aimed to make the development process smoother and more efficient for any team members who may work on the project.

```php
// ---------------  TASKS SECTION

Route::get('/tasks', [TasksController::class, 'index'])->name('tasks');
Route::get('/task/new', [TasksController::class, 'new'])->name('task.new');
Route::get('/task/{task}', [TasksController::class, 'show'])->name('task.show');
Route::get('/task/{task}/edit', [TasksController::class, 'edit'])->name('task.edit');
Route::get('/task/status/{id}', [TasksController::class, 'getStatus'])->name('tasks.status.get');
Route::post('/task', [TasksController::class, 'create'])->name('task.create');
Route::post('/task/delete/', [TasksController::class, 'delete'])->name('task.delete');
Route::post('/task/status', [TasksController::class, 'updateStatus'])->name('task.status.update');
Route::post('/task/timer/update', [TimerController::class, 'updateRecord'])->name('task.timer.update');
Route::post('/task/progress/update', [TasksController::class, 'updateProgress'])->name('task.progress.update');
Route::patch('/task/{task}', [TasksController::class, 'update'])->name('task.update');

// ---------------  CLIENTS SECTION
Route::get('/clients', [ClientsController::class, 'index'])->name('teams');
Route::get('/client/new', [ClientsController::class, 'new'])->name('client.new');
Route::get('/client/{client}', [ClientsController::class, 'show'])->name('client.show');
Route::get('/client/{client}/edit', [ClientsController::class, 'edit'])->name('client.edit');
Route::post('/client', [ClientsController::class, 'create'])->name('client.store');
Route::patch('/client/{client}', [ClientsController::class, 'update'])->name('client.update');

// ---------------  TEAMS SECTION
Route::get('/teams', [TeamsController::class, 'index'])->name('teams');
Route::get('/team/new', [TeamsController::class, 'new'])->name('team.new');
Route::get('/team/{team}', [TeamsController::class, 'show'])->name('team.show');
Route::get('/team/{team}/edit', [TeamsController::class, 'edit'])->name('team.edit');
Route::post('/team', [TeamsController::class, 'create'])->name('team.create');


// ---------------  COMMENTS SECTION
Route::get('/comment/new', [CommentsController::class, 'new'])->name('comment.new');
Route::get('/comment/{comment}', [CommentsController::class, 'show'])->name('comment.show');
Route::get('/comment/{comment}/edit', [CommentsController::class, 'edit'])->name('comment.edit');
Route::post('/comment', [CommentsController::class, 'create'])->name('comment.store');
Route::patch('/comment/{comment}', [CommentsController::class, 'update'])->name('comment.update');
```

Validation of Input Data:

The controller ensures data integrity through a multi-layer validation process. Front-end validation checks user inputs before submission, while the controller validates data once it reaches the server. Additionally, Laravel's policies are used to verify that the user has the proper permissions for the requested action. This combined approach helps prevent errors, maintain the system secure, and ensure that only authorized users can access or modify data.

```
<div class="col-3">
    <x-input-label for="name" :value="__('Task Name')" />
    <x-text-input id="name" class="form-control" type="text" name="name"
        value="{{ old('name') }}" required autofocus autocomplete="name" />
    <x-input-error :messages="$errors->get('name')" class="mt-2" />
</div>
```

```
public function create()
{
    $this->authorize('create', Task::class);

    $data = request()->validate([
        'name' => ['required'],
        'description' => ['required'],
        'responsible_id' => ['required'],
        'details' => ['required'],
        'start_date' => ['required', 'date'],
        'finish_date' => ['required', 'date'],
        'stage_id' => ['required'],
        'status' => ['required'],
        'priority' => ['required'],
        'progress' => ['required'],
    ]);

    $task = auth()->user()->tasks->create($data);
    return redirect()->to(url()->previous());
}
```

Compact and Clear Method Definitions:

Methods are clearly defined and separated based on functionality, enhancing code readability and maintainability. Each method handles a specific task related to its designated service, ensuring a clear separation of concerns.

```php
public function index(Task $task)
{
    $this->authorize('viewAny', Task::class);
    $tasks = Task::where('user_id', auth()->user()->id)->where('status', 'open')->get();
    return view('tasks/index', compact('tasks'));
}
```

Use of Policies:

Leveraging policies for CRUD operations exemplifies a best practice for managing permissions and ensuring application security. By implementing policies across Blade files, the application centralizes permission management, allowing for streamlined control over viewing, collaborating, editing, and deleting functionalities. This approach minimizes the need for changes in other parts of the application when permissions need to be adjusted, as functionalities are managed within the policy files. Consequently, this setup facilitates the flexibility to grant or restrict access based on client requirements, making it easier to maintain and update permissions.

ProjectPolicy.php

```php
public function update(User $user, Project $project)
{
    return ((
                ($project->user_id === $user->id) ||
                ($project->teams->contains('leader_id', auth()->user()->id) !== null)
            ) &&
            $project->status === 'open');
}


public function open(User $user, Project $project)
{
    return (
                ($project->user_id === $user->id) ||
                ($project->teams->contains('leader_id', auth()->user()->id) !== null)
            );
}

public function collaborate(User $user, Project $project)
{
    $userId = $user->id;

        return (
            (($project->teams->contains('user_id', $userId) !== null) ||
            ($project->teams->conatains('leader_id', $userId) !== null) ||
            $project->teams->collaborators->contains('id', $userId) ||
            ($project->user_id === $userId)) &&
            ($project->status !== 'closed')
        );
}
```

projects/show.blade.php

```
@section('content_header')
    <section class="mt-1">
        @if ($data->module->teams->isEmpty())
            <div class="callout callout-danger">
                @can('update', $data->module)
                    <h3 class="text-danger">This Project has not any associated collaborators. Assign a team</h3>
                @endcan
                @cannot('update', $data->module)
                    <h3 class="text-danger">This Project has not any associated collaborators.</h3>
                @endcannot
            </div>
        @endif
    </section>
@stop
@section('content')
    <div class="card" id="app">
        <div class="card-header">
            <div class="row">
                <div class="col-8">
                    <h2 class="text-primary">{{ $data->module->name }}</h2>
                </div>
                <div class="col-4 text-right">
                    @can('update', $data->module)
                        <a class="btn btn-primary m-2" href="{{ route('project.edit', ['project' => $data->module->id]) }}">Settings</a>
                    @endcan

                    <a class="btn btn-primary m-2" href="{{ route('project.tasks', ['project' => $data->module->id]) }}">Show Tasks</a>

                    @can('open', $data->module)
                        @if ($data->module->status === 'closed')
                            <button class="btn btn-warning m-2" onclick="openProject({{ $data->module->id }})">Open this Project</button>
                        @endif
                    @endcan
```

```
                    @can('collaborate', $data->module)
                        @include('comments.new')
                    @endcan
                    @can('view', $data->module)
                        @include('comments.show')
                    @endcan
```

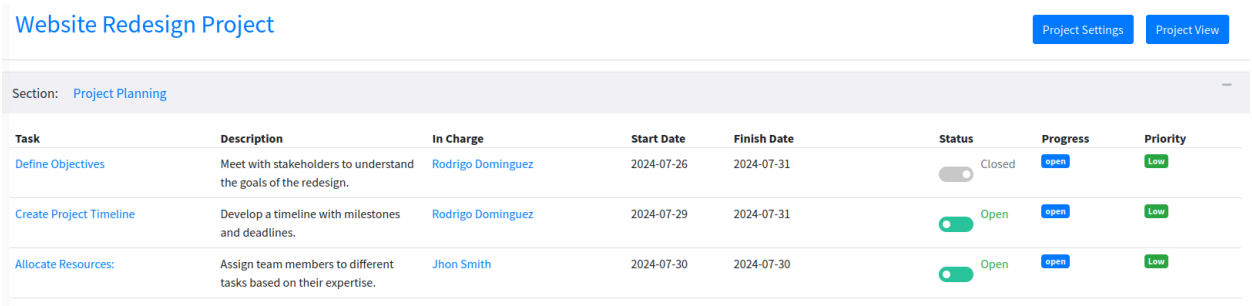
Use of Vue.js

Organizer is primarily developed using PHP, leveraging the powerful Laravel framework for backend operations. However, to enhance user interactivity and provide a more dynamic experience, few key components and functionalities are built using Vue.js. This integration of Vue.js is aimed at making the application more responsive and user-friendly. Some features built in vue are:

Updating Task Status:

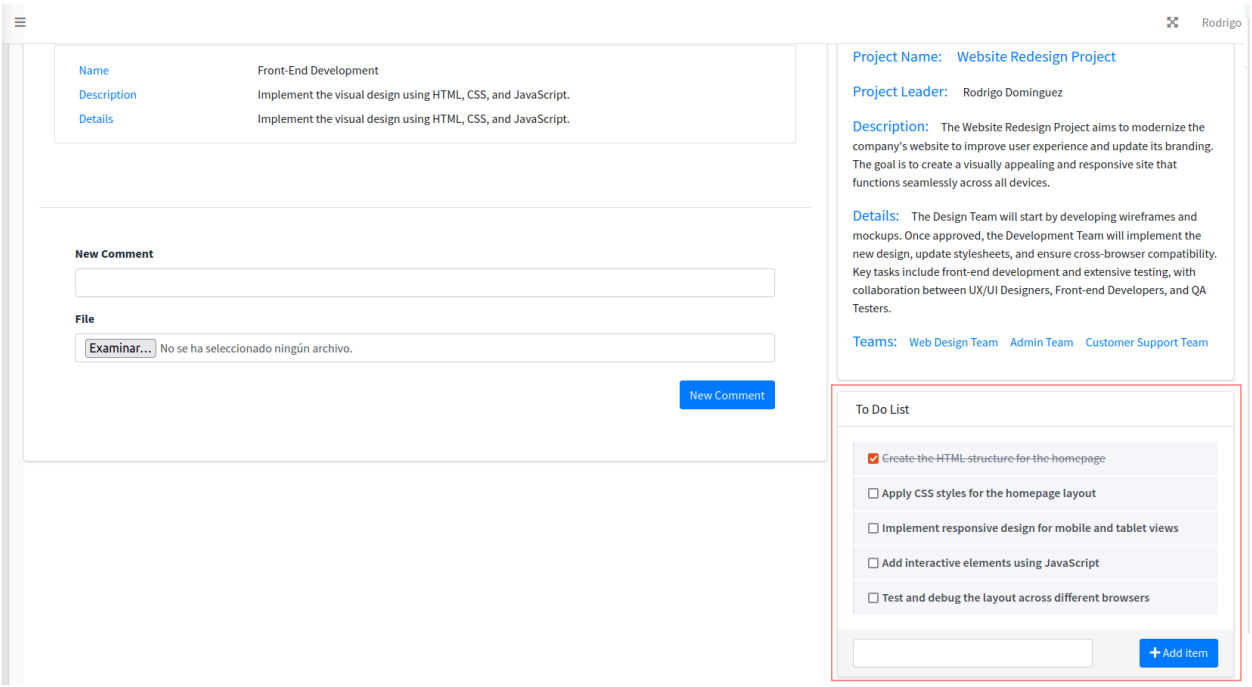
Task statuses can be updated in real-time without requiring a full page reload. This is achieved through Vue components that handle the status changes and communicate with the backend via Axios, ensuring that updates are reflected immediately in the UI.

In the Project Task list view, the task status can be updated through a toggle button. This button calls Axios to post a status update method. This approach allows for real-time status updates without requiring a full page reload, making the user experience more dynamic and efficient.



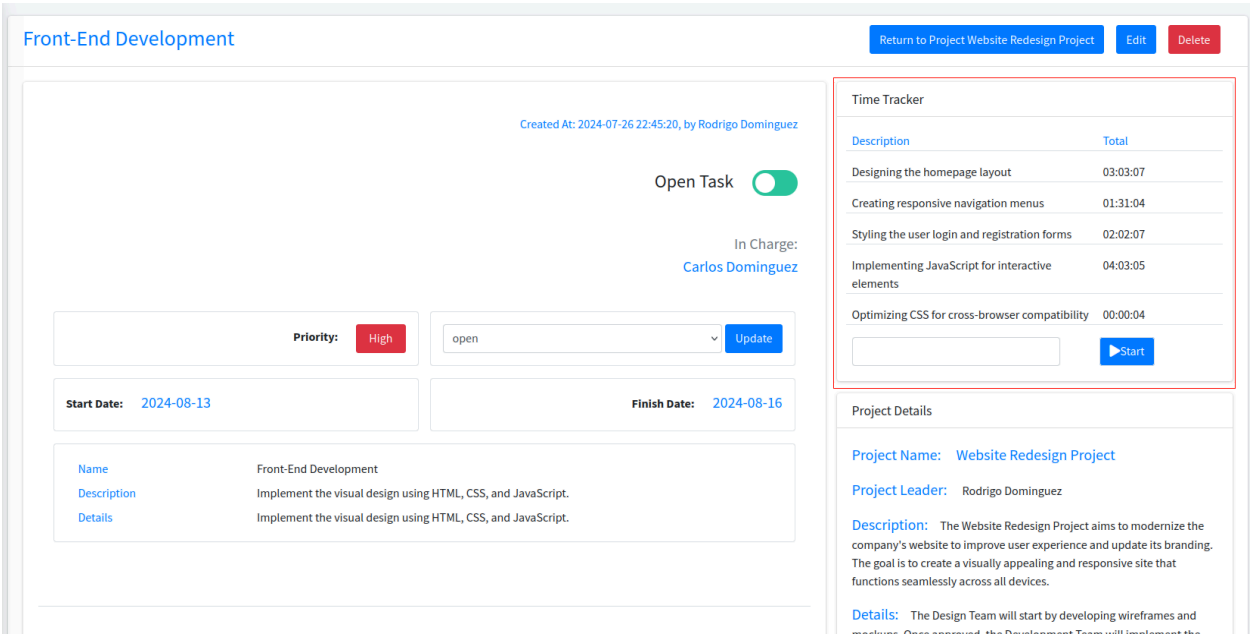
To-Do Lists:

Users can create and manage to-do lists with a more interactive interface. Vue.js components facilitate the dynamic addition and removal of list items, enhancing the overall task management experience. This action is present in the Task view and user Dashboard .



Time Tracker:

The time tracking feature utilizes Vue.js to provide a responsive and real-time tracking interface. Users can start, pause, and stop timers with immediate visual feedback, making it easier to manage and record time spent on tasks.



Conclusion

Summary

Organizer is a project management and team collaboration platform developed with PHP (Laravel) and Vue.js. It's designed to help teams stay productive and manage projects efficiently. With features like project tracking, task management, time tracking, and document sharing, it's built to support a wide range of team and project needs.

Future Prospects

Looking ahead, I see the potential for Organizer to integrate AI tools, Google Calendar, and Clockify. These additions will make the platform more versatil, useful and integrated with popular applications. The way I've set it up with Laravel policies means it can easily adapt to different needs, keeping everything organized and making updates straightforward.

Final Thoughts

Working on Organizer has been a fantastic experience for me. It's not just a project but a reflection of my passion for software development and my journey of learning. My background in telecommunications gave me a unique perspective as an Application User, Team Leader, Problem Solving approach. I've enjoyed using my personal time to dive into development and create something that really works.

I want to share that I have a genuine enthusiasm for learning and tackling new challenges. I'm driven by a personal interest in developing software and improving my skills. I love getting involved in projects that are interesting and push me to grow. My journey with Organizer has been a great way to channel my energy and curiosity, and I'm excited to bring that same passion and drive to future opportunities.