Proceedings of the International Conference IMCS-60, MSU
October 10–13, 2024, Chisinau, Republic of Moldova

Orion – Smart Generation of Work Schedules

Tudor Tescu, Adrian Iftene

Abstract

In today's work environments, many businesses, especially those active in the HoReCa (Hotel, Restaurant, and Café/Catering), stay away from fixed schedules. Since most of such companies are open every day, there exists the problem of employees having to work weekends and, by having a variable schedule, no employee will have to work every weekend. The use of variable schedules, however, creates the burden of creating an optimal arrangement of shift assignments very often, and business owners are tasked with spending many hours considering each employee's preferences, capabilities, and contractual details. The present paper presents a solution that comes to the aid of these owners and helps them manage their employees' activities more efficiently.

1 Introduction

While there already exist a vast amount of applications dedicated to helping business owners and their teams create schedules faster and easier, most of them only allow the creation of schedules and the assignment of employees, with no extra details or capabilities. The applications that can generate schedules often generate an arrangement of shifts and assignments that approximately match previously created schedules but without any customization. Thus, the idea of creating a better application with powerful features for business owners emerged.

The selling point of this application is the automatic schedule generation feature. This functionality works hand-in-hand with the employee profile customization capability, allowing the employer to set

©2024 by Tudor Tescu, Adrian Iftene

specific conditions for each employee, such as limiting them to certain shift areas (for example, a cook should only work in the kitchen and should not be assigned to the serving area), set the maximum number of hours allowed per day, give the employee several days off after working for several consecutive days (many businesses such as restaurants prefer to have their serving staff work for two consecutive days, often for 12 hours each, and then give them one or two days off), or to explicitly provide the employees time off. As for the employees, they can select (either manually or from the defined shifts in the latest and upcoming schedules) their unavailability intervals. These features work hand-in-hand to offer the employer and their team an efficient method of generating schedules based on imposed conditions.

2 Similar applications

Deputy[1] is an app that offers scheduling features and the ability to create and swap shifts. Furthermore, it tracks attendance and working hours and helps businesses comply with state laws. Deputy also offers an automatic generation feature, which, however, only creates an approximate arrangement of shifts based on previously created schedules.

Coast[2] is an app that markets itself as an alternative to Deputy. It offers the same features as the aforementioned Deputy app while also giving employers the means of messaging their team.

Like the previously mentioned applications, **Homebase**[3] aims to help business owners with scheduling. It offers advantageous features such as calendars, time off, and compliance with local law, but it lacks automation, which is the key point of the application presented in this paper.

[1] www.deputy.com
[2] www.coastapp.com
[3] www.joinhomebase.com/employee-scheduling/

3 Architecture and Implementation

Orion, the application presented in this paper, consists of the frontend application and the backend services. The frontend application is built using Angular and Taiga, and the backend consists of two separate microservices - Identity, which is responsible for handling authentication, and Administrative, which is responsible for handling everything related to the administration of the business and the schedules. Both of the aforementioned microservices are built following the Domain Driven Design architecture [1] and the Test Driven Design pattern to ensure consistency and reliability. Furthermore, to provide increased security, performance, and scalability, the backend services follow the CQRS design pattern. The infrastructure layer holds and implements abstractions of services declared in the application layer. Furthermore, it holds the implementation of the repository and manages the connection to the database. While the layer itself does not perform many actions, it is crucial for implementing a backend application following the DDD approach, as it is also responsible for most configurations that involve the use of services or the handling of credentials. The presentation layer is the layer responsible for communication with the outside world. In this case, the presentation layer is the intermediary between the Angular frontend application and the rest of the backend layers. For the development purpose of Orion, the presentation layer consists of a series of Azure Functions HTTP Triggers, which are methods that intercept and handle HTTP requests by sending commands or queries to the respective handlers. While Azure Functions also supports other useful features, such as a Service Bus (which would have been helpful in this application in specific scenarios), only the HTTP triggers have been used, as the other features would have required more infrastructure and costs.

3.1 Use Cases and Benefits

While the application's selling point is the intelligent generation feature, it was designed to be user-friendly and easily understandable.

Thus, the most common use case is viewing the created schedules in a human-friendly calendar format. Furthermore, statistics and information for each employee can be observed, such as contract hours, worked hours, and various conditions. The feature can be observed in Figure 1 below.

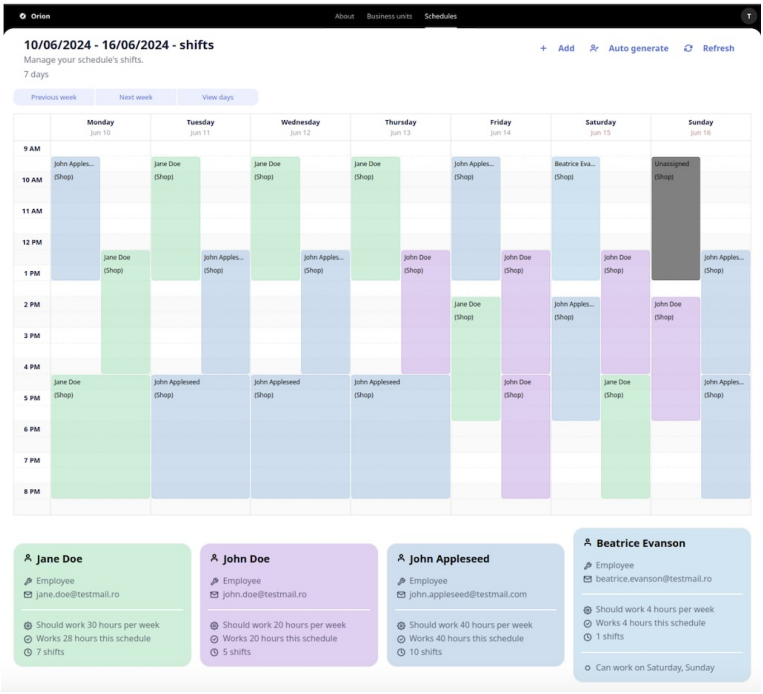


Figure 1. The schedule overview feature

The chosen design displays all the necessary information regarding the schedule. Assigned shifts can be easily observed and are color-coded uniquely to each employee. Furthermore, unassigned shifts have a distinctive look, making them difficult to miss. Statistics can be seen for each employee, making it easier to identify potential issues.

4 Conclusion

The presented application can serve as a powerful tool in a business owner's arsenal by allowing them to easily schedule their employees, either manually or by leveraging the built-in automation feature. During the development of the presented application, we faced various challenges, ranging from choosing the tech stack, architecture, and libraries to figuring out the best possible way of generating shift assignments.

While this application can be considered complete, well-structured, and reliable, further improvements can be made in the future. First off, additional employee conditions can be created, such as shift pairing restrictions to promote better dynamics between certain employees, mandatory days off, and various others. Additionally, certain pages could benefit from redesigns, such as the page that displays a specific schedule. This particular page has proven to be a great challenge, as developing a calendar from scratch would have been a challenging and highly time-consuming task, and libraries that could have provided a better experience are, unfortunately, hidden behind paywalls, which would have been detrimental. Last but not least, the generation algorithm would also benefit from further research and improvement, as an algorithm can never be too efficient. By creating further versions of the algorithm, more cases could be considered, such as the additional conditions mentioned earlier.

References

[1] E. Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*, Addison-Wesley Professional, 2003.

Tudor Tescu, Adrian Iftene

"Alexandru Ioan Cuza" University, Faculty of Computer Science, Iasi, Romania
E–mail: {tudor.tescu, adiftene}@gmail.com
ORCID: https://orcid.org/0000-0003-3564-8440