Development of a web application for managing the job market using TALL stack

Mario Milunović
Information Technology School
Belgrade, Serbia
mario.milunovic@gmail.com

Selver Pepić
Academy of Professional Studies
Šumadija
Trstenik, Serbia
s.pepic@uninp.edu.rs

Goran Miodragović
Academy of Professional Studies
Šumadija
Trstenik, Srbija
gmiodragovic@asss.edu.rs

Abstract - **This is document presents the key elements in the process of designing and developing a web application intended for managing communication between employers and workers. The design process is based on Larman's method [1] of software development and is done in the following steps:**

- **Verbal description of the application**
- **Constructing ER diagrams and relational database schemas**
- **Identification of use cases**
- **Detailed description of all use cases in the form of a scenario**
- **Defining system operations**
- **Creating sequence diagrams for key system operations**
- **User interface design**
- **Coding**
- **Deployment**

Application is developed using TALL stack.

Keywords – TALL; PHP; MVC; Laravel; Livewire; Tailwind; Alpine.js; MySQL; Larman; DigitalOcean

I. INTRODUCTION

The application presented in this paper is a vision of the mechanism by which the supply and demand of part-time jobs can be managed. The idea was inspired by well-known Internet sites with the same goal, such as Freelancer.com, Upwork.com, Fiverr.com, etc.

The architecture of the application is based on the MVC model [2] and the TALL stack was used for development. TALL is an abbreviation made up of the initials of 4 popular frameworks that are most often used in combination: Tailwind, Alpine.js, Laravel, Livewire.

Tailwind is a CSS framework consisting of a set of predefined classes with which you can style a html elements with the ease of a Bootstrap framework but with much more freedom when it comes to design. Tailwind has very good support for creating html pages that automatically adapt to different display sizes (responsive design).

Alpine.js is a JavaScript framework that allows the user to create simple interactive components by writing code directly within html tags.

Laravel is currently the most popular PHP full-stack framework for developing web applications based on the MVC architecture.

Livewire is a framework that, in addition to Laravel, offers developers the ability to create a modern dynamic graphical interface while remaining in the domain of the PHP language. Livewire and Alpine.js come from the same author.

II. VERBAL DESCRIPTION

The application should enable employers to publish jobs and then choose the one that suits them best from the received offers sent by workers. On the other hand, application should enable workers to search for job postings and submit their offers. The application allows the user to be both in the role of employer and in the role of employee. In other words, user can make offers for other people's jobs and collect offers for their own jobs.

The app offers employers a display of real user skills. The user can not only state that he has a certain skill. After each successfully completed job, user receives one point for the skill that the job required. On the other hand, the application offers protection to workers because it reserves the employer's funds at the moment when the employer hires a worker. In other words, the employer must have funds in advance to cover the agreed costs. The application has its own internal currency that the user can buy or pay to his bank account.

The owner of the selected bid starts working and after the completion of the works he declares his bid delivered. The employer must then declare that the offer is accepted if

he considers that the employee has successfully completed the task. When he does that, the reserved funds are transferred to the worker's account and the worker is awarded one point per skill.

III. DESIGN

A. ER DIAGRAM AND REALTIONAL MODEL

Fig. 1 shows the Entity Relationship diagram [3]. This diagram started the design process by identifying the most important entities and defining their relations based on the on the analysis of the verbal description.

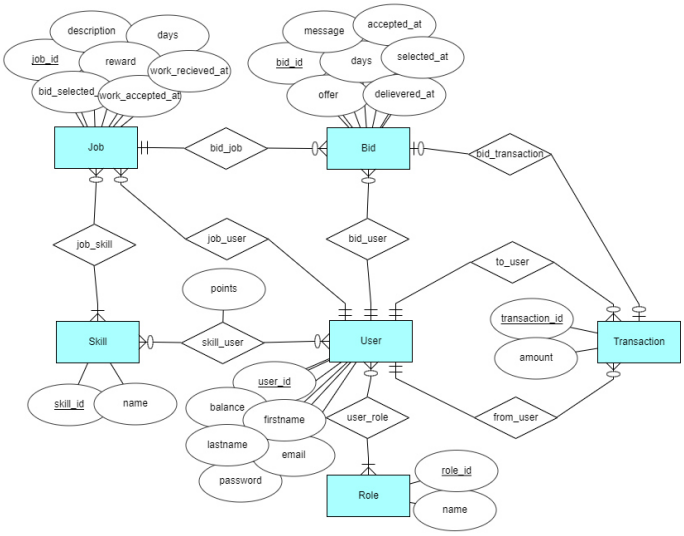

Figure 1. ER diagram

Based on the ER diagram, the relational scheme shown in Fig. 2 was formed.

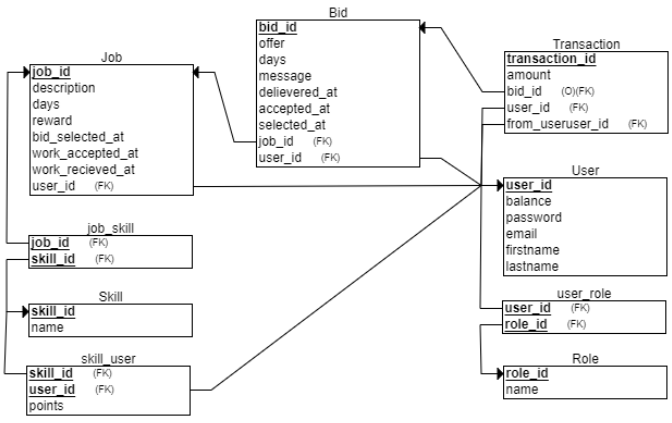

Figure 2. Relational schema

B. USE CASES

Fig. 3 shows all use cases grouped into 6 categories.

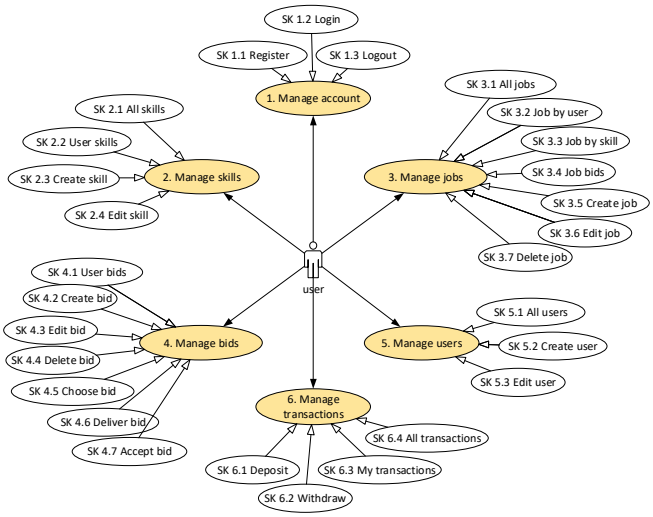

Figure 3. - Use case diagram

The use cases [1] are then described in the form of a scenario. An example of a scenario for the use of SK 4.5 is given. The names of the files containing the code for the graphical interface used within the scenario are also given during the description.

Use case 4.5 - The employer chooses the bid

DESCRIPTION:
Declaring that the offer was chosen as the most favorable.

PREREQUISITES:
• Web browser displays a form with detailed information about the bid (components.bid)
• The business owner must have enough money in his account to cover the cost of the bid
• The bid must have the status “Created”.

BASIC SCENARIO:
1. By clicking on the "Choose" command, the employer confirms the selection of the best bid
2. The application performs the bid selection process
3. The application displays a list of all bids of the selected job (models.job.bids)

ALTERNATIVE SCENARIO:
2.1 The employer does not have enough money and the application displays a notification about it

C. ACCESS RIGHTS

We can divide users into unregistered users, registered users and administrators. An overview of access rights for each use case is shown in Table 1. The table also lists the names of the controllers and the methods in which this functionality is implemented. In this way, the reader can easily orient himself during code analysis.

Table 1 - Access rights for each use case

Group	No	Use case name	Controller	Method	ADMIN	REGISTERED USER	UNREGISTERED USER
Manage account	1.1	Register	Register	register	●	●	●
	1.2	Login	Login	login	●	●	●
	1.3	Logout	Logout	logout	●	●	●
Manage skills	2.1	All skills	SkillController	index	●	●	●
	2.2	User skills	SkillController	user	●	●	●
	2.3	Create skill	SkillController	store	●	●	●
	2.4	Edit skill	SkillController	update	●	●	●
Manage jobs	3.1	All jobs	JobController	index	●	●	●
	3.2	Jobs by user	JobController	user	●	●	●
	3.3	Jobs by skill	JobController	skill	●	●	●
	3.4	Job bids	JobController	bids	●	●	●
	3.5	Create job	JobController	store	●	●	●
	3.6	Edit job	JobController	update	●	●	●
	3.7	Delete job	JobController	destroy	●	●	●
Manage bids	4.1	Bids by user	BidController	index	●	●	●
	4.2	Create bid	BidController	store	●	●	●
	4.3	Edit bid	BidController	update	●	●	●
	4.4	Delete bid	BidController	destroy	●	●	●
	4.5	Choose bid	BidController	select	●	●	●
	4.6	Deliver bid	BidController	deliver	●	●	●
	4.7	Accept bid	BidController	accept	●	●	●
Manage users	5.1	All users	UserController	index	●	●	●
	5.2	Create user	UserController	store	●	●	●
	5.3	Edit user	UserController	update	●	●	●
Manage transactions	6.1	Deposit	DepositController	store	●	●	●
	6.2	Withdraw	WithdrawController	store	●	●	●
	6.3	My transactions	TransactionController	user	●	●	●
	6.4	All transactions	TransactionController	index	●	●	●

D. SISTEM OPERATIONS

The analysis of usage case scenarios showed that most system operations can be reduced to trivial CRUD operations [4] or SQL queries. To implement CRUD operations, it is enough to call the methods that the Laravel framework already offers for all created models. Sequence diagrams [1] were used to describe the mechanism of non-trivial system operations that are crucial for business logic. Fig. 7 shows a sequence diagram for the system operation “Bid Selection”.

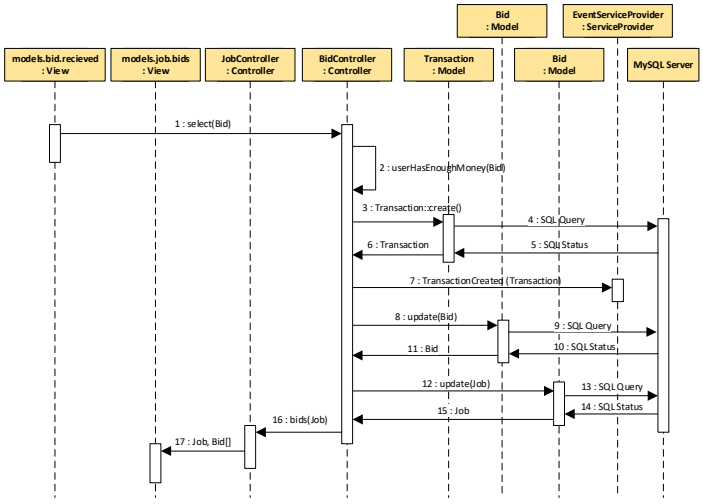

Figure 4 - Sequence diagram for sistem operation "Bid selection"

E. USER INTERFACE DESIGN

The user interface is sketched for each use case using the free app Pencil. In Fig. 5 is an example of a sketch for use case 3.4 “Job bids” which shows the app layout as well as the design of the main menu.

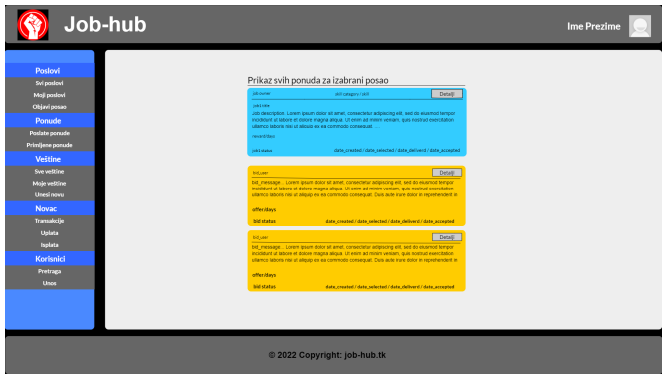

Figure 5 - Sketch of the user interface for SK 3.4

IV. CODING

A. MIGRATION AND MODEL CREATION

All models were created through a migration technique [5] that incrementally modifies the database schema. The migration class defines the primary and foreign keys, model attributes as well as the method by which the migration effect can be undone. The Laravel command "php artisan make: model Job -all" will create a class, controller, migration and factory for the Job model.

B. FILLING THE DATABASE WITH TEST DATA

Laravel comes with the PHP library "Faker" which contains methods that generate various test data (name, surname, email, address, country, etc.). A Factory class has been created for each model, which defines how to create a test instance of that model. The Seeder class [2] methods are then called, which use the Factory class method to fill the database with the desired amount of data.

C. GRAPHICAL INTERFACE CODING

Based on the sketches, the framework of the entire application layout was made first. A CSS grid [6] layout was used, which was set to switch between the standard layout and the layout adapted to smaller screens at a given resolution. The next step is to create the main menu. Alpine.js framework was used for animations within it. The menu options are displayed in accordance with the logged in user access rights table. The last step is to create forms for all system operations. All created forms are placed within the app layout. Fig. 6 shows the layout, main menu and bid entry form.

Figure 6 - Application layout

D. CODING CONTROLLER METHODS

In the first phase, controller methods were created for all CRUD operations. After that, all other more complex business logic methods were created.

V. DEPLOYMENT

The free domain job-hub.tk has been registered for publishing the application through the Freenom domain provider. For hosting is used IaaS provider Digital Ocean and its latest service which enables direct publishing of the Laravel application by connecting to its GitHub repository. The DigitalOcean system monitors the master branch of the GitHub repository, and if a change is detected on it, a new version is automatically deployed. So, it is enough to execute a git push command from the development environment on the master branch and the changes will be published in a few minutes.

VI. SOURCE CODE

Source code is publicly available on GitHub:
github.com/mariomilunovic/job-hub

VII. CONCLUSION

The Laravel framework has proven to be a very effective tool that deservedly holds the position of the best PHP framework. Laravel provides great comfort to web developers because it offers ready-made solutions for many functionalities that a typical web application needs. It has great support and a solution for almost every problem can be found on the Internet.
TALL is a complete solution for creating full-stack web applications and its main advantage is that it gives the developer the opportunity to stay focused on PHP language and produce a result that would otherwise require additional skills from another frontend JavaScript framework.

VIII. REFERENCES

[1] Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd ed., Prentice Hall, 2005.

[2] Pitt, Christopher. Pro PHP 8 MVC: Model View Controller Architecture-Driven Application Development. 2nd ed., APress, 2021.

[3] I, Anikina Yelena. Entity Relationship Approach to Database Design. LAP Lambert Academic Publishing, 2015.

[4] Sharma, Manu. MongoDB Complete Guide: Develop Strong Understanding of Administering MongoDB, CRUD Operations, MongoDB Commands. BPB Publications, 2021.

[5] Pecoraro, Christopher John. Mastering Laravel. Packt Publishing, 2015.

[6] Meyer, Eric. Grid Layout in CSS: Interface Layout for the Web. O'Reilly Media, 2015.