Volume 61, Number 2, 2021

ACTA TECHNICA NAPOCENSIS
Electronics and Telecommunications

AUTO DAKAR: A WEB APPLICATION FOR THE MANAGEMENT OF AN AUTOMOBILE REPAIR SHOP

David GRUIAN, Iustin-Alexandru IVANCIU
Communications Department, Technical University of Cluj-Napoca, Romania
Corresponding author: Iustin-Alexandru Ivanciu (e-mail: Iustin.Ivanciu@com.utcluj.ro)

Abstract: **When it comes to an automobile service, it is very important to ensure not only good communication with the customers but also a strong online presence. This paper presents AUTO DAKAR, a highly customizable and easy to use web application designed to provide online visibility for the company and communication methods between customers and the automobile service. The application was developed with Laravel using the Voyager package for backend management, a Heroku server for testing and a DigitalOcean server for production. The experimental results highlight the main advantages of AUTO DAKAR: flexibility, easy maintenance and cost efficiency.**

Keywords: *automobile service, DigitalOcean, Heroku, Laravel, MySQL, PHP, web application*

I. INTRODUCTION

In recent years, there has been a switch from the static websites to a more interactive approach – web applications. These web applications can be found in many of our daily activities: from socializing to online shopping and even public communication. The main advantage is the availability on most devices without any installation required. Moreover, as more efficient methods are implemented, web applications are capable of providing more features with less performance issues.

In 2020, Romania registered a number of 14,102 economic agents with the object of activity included in the CAEN code 4520 (Maintenance and repair of motor vehicles) [1]. The number of economic agents has increased in recent years. The analysis carried out by KeysFin shows that in 2010 there were 9320 economic agents in Romania and 5 years later their number reached 10,031 [2]. These values indicate a significant increase in the number of car services in recent years and imply a development of the competitive market in this field.

Many car services rely on person-to-person recommendations and sketchy websites or platforms that display outdated data about their company alongside ads or unwanted content. The problem is that customers are not able to learn more about the company and the quality of services they provide. Users want to see reviews, to communicate with the car service or ask for an estimate without visiting the service. Without these features businesses tend to lose trust and ultimately customers.

Specialized studies do not present any report or analysis on the quality of web pages of car services in Romania, so in order to form a more accurate and relevant image, the competitive area will be chosen for the service where the application, named AUTO DAKAR, will be used. Analyzing separately the companies with the CAEN code 4520 from the Alba County and from the city of Alba Iulia, sorted in descending order of turnover, the following results were obtained: out of 40 competing companies from the county (without the city of Alba Iulia) analyzed, only 6 of these have a website, and in the city of Alba Iulia, out of 15 companies analyzed, only 4 have a website.

A web application is a reliable way to achieve online presence and interact with the customer. To get an application running, 3 main components are required: 1) Frontend, 2) Backend, and 3) Server hosting. Frontend mainly refers to what the user sees and is the most relevant from a business standpoint. The backend is required to manage the functionality processes and send the data that will be displayed to the user in frontend; the server is the place where the application will be stored.

When we talk about web applications, hosting is a very important element. There are multiple methods to get the application on the Internet and one of them is self-hosting. It requires hardware acquisition, system administration knowledge and usually a high cost to maintain. On the other hand, cloud services provide a more cost-efficient and hustle free method, in terms of configuration and maintenance of hardware.

The goal of this paper is to implement a web application for managing the family business, an automobile repair shop located in Alba Iulia. This application uses Laravel as a backend framework, Voyager package as an admin panel and on the client side, HTML, JavaScript and CSS organized on Laravel blade structure. The development was done using a local Windows machine, the testing environment using a Platform as a Service (PaaS) Heroku server and the production environment using an Infrastructure as a Service (IaaS) DigitalOcean server for cost reduction purposes.

This paper is organized as follows: Section 2 presents related work, while Section 3 discusses the general aspects regarding web development tools and how they are working together. Section 4 describes the proposed web application and includes the setup of development environments. Experimental results are presented in Section 5, and the paper ends with conclusions and future work.

II. RELATED WORK

In paper [3], a comparison of PHP frameworks and their

Manuscript received July 30, 2021; revised November 29, 2021

performance is analyzed. Laravel was chosen because it provides its own ecosystem of built-in features and is compatible with many open-source modules. Considering that the application requires a client side and an administration dashboard, a flexible framework was needed.

An administration dashboard is needed for content management. Such a page can be created from scratch or using a Laravel package. Laravel recommends its own product: Laravel Nova. This package works with a monthly subscription and because of costs optimization, a free, open-source package was chosen: Voyager. An administration package is supposed to provide create, read, update and delete (CRUD) methods alongside a graphical interface for better usability.

When it comes to frontend and user interface, there are multiple JavaScript frameworks that use a Model-View-Controller (MVC) structure as Laravel does [4]. Using such frameworks will require an API point to get the data that will be displayed to the user. Laravel is a backend-oriented framework but it provides its own methods of creating views. By integrating *blade*, the interface can be split into components that display data provided by Laravel controllers.

Considering that the web application needs to be available online, a web hosting service is required. The focus is on finding a cost-efficient method so we've chosen a cloud service to satisfy our needs. Our web application uses a private testing environment and a public production environment. In paper [5], a comparison between the main providers and the quality of their services is made. Each provider was evaluated by availability, reliability, performance, cost and security. Based on those findings, a better choice can be made regarding the type of server wanted and its performance. The types of servers taken into account are Platform as a Service and Infrastructure as a Service. In paper [6], an in-depth research of the providers for each type of servers is made. Considering the traffic needs of the web application, the size will be set accordingly. Also, based on budget and preferences, there are options for monthly fixed payment or pay per consumption. In most cases, PaaS services are more expensive and offer less performance than IaaS, but require less knowledge to setup and maintain.

III. WEB DEVELOPMENT TOOLS

Laravel [7] is a free and open-source PHP framework used to develop web applications. The framework provides its own ecosystem of built-in features and is compatible with many open-source modules developed by the community. It was created in 2011 by Taylor Otwel, using the MVC architecture, based on the Symfony framework. Laravel provides a variety of tools for faster and better development: an Object Relational Mapper (ORM) for database interaction, blade for frontend development through reusable components, artisan commands for specific file generation or a full authentication system generated through a command and many more.

Voyager is an open-source Laravel package developed by The Control Group that offers a full administration system built with blade using jQuery, Bootstrap and many other small JavaScript libraries. Voyager offers features like media manager – used for storage administration, a menu builder for easy menu management, a database manager which allows the manipulation of the database table structure and a create, read, update, delete builder for any table thus simplifying data manipulation.

Hosting the project on the Internet requires a domain name and a host. Heroku is a company that specializes in PaaS cloud services. They provide a free tier where an app can be used with limited resources, ideal for testing. DigitalOcean [8] is a company that offers IaaS cloud services, perfect for a project like the one presented herein.

Ubuntu is one of the images that can be installed on a server. Ubuntu is a Linux distribution composed mostly of open-source software. Features like user privileges, network and firewall control and the file structure make it a very powerful and secure tool for hosting web applications in different cloud environments.

IV. IMPLEMENTED WEB APPLICATION

The implementation process is composed of 2 stages: the development stage and the deployment stage. The main components and the tools used in each stage are illustrated in Figure 1.

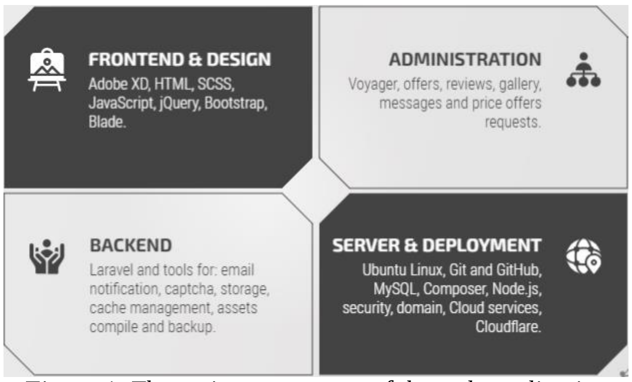

FRONTEND & DESIGN
Adobe XD, HTML, SCSS, JavaScript, jQuery, Bootstrap, Blade.
ADMINISTRATION
Voyager, offers, reviews, gallery, messages and price offers requests.
BACKEND
Laravel and tools for: email notification, captcha, storage, cache management, assets compile and backup.
SERVER & DEPLOYMENT
Ubuntu Linux, Git and GitHub, MySQL, Composer, Node.js, security, domain, Cloud services, Cloudflare.

Figure 1. The main components of the web application

Frontend & Design refers to what the end user or customer will see and it integrates most of the visual tools. The Administration or the Admin panel refers to the main methods that were implemented for the automobile repair shop and what the administrator can do using Voyager. The backend is the component that makes the binding between frontend and administration, and it consists of the Laravel framework and multiple tools that add value to the application. Server & Deployment represents the tools required for the application to be available within the Internet.

During the development stage the following components were integrated: **User interface**, consisting of static pages with information, contact forms, pages with dynamic content (manageable) and analysis tools. **The admin dashboard** is composed of an administration panel accessible with email and password, from where one can manage the dynamic content on the site and see the messages sent by customers. **The backend** enables the communication between interfaces and is built with Laravel and other packages such as: Voyager, mail, captcha, Amazon Web Services (AWS) S3, cache-response, Laravel mix, backup, etc. The development process is highly dependent on the environment configuration so the next section will focus on the steps taken on the local environment to create such an application.

A. Local environment

The framework was installed using the Composer command line interface (CLI). Once the download finished, the environment variables, specific to the new project and located in the *.env* file, were configured accordingly. Note that before actually starting to write code, some steps need to be taken to ensure the correct functionality of the platform. To run a PHP app locally a virtual host is required and that was achieved using XAMPP because it provides a MySQL database alongside its Apache server.

To setup and bind localhost to the *index.php* file the following code should be added:*127.0.0.1 service.local* inside *hosts* file located in C:\Windows\System32\drivers\etc. A virtual host should be added to XAMPP *httpd-vhosts.conf* file. Note that a restart of the Apache server is required for the changes to take effect.

The database was created using the XAMPP MySQL phpMyAdmin page: http://localhost/phpmyadmin/. After creation, the credentials were added to the *.env* file. Git also needed to be initialized so the project could be stored safely into a GitHub repository. Using the steps provided by GitHub, the project was added under an initial commit and stored on the *main* branch.

B. Heroku server

The next stage of the project focuses on how to use the Platform as a Service services offered by Heroku and how to configure the Laravel application previously configured on the local server. The first step is to create an account and a new app on the platform. For better usability and manually deployment, the Heroku CLI was installed. Note that because we're going to run a Laravel app, a Heroku specific file is required to specify the location of the *index.php* from where the application boots. The new file will be named *Procfile* and will contain the following code: *web: vendor/bin/heroku-php-apache2 public/.*

The next step is to login to Heroku through CLI and upload the code to a Heroku repository. The command *heroku login* will start the authentication process. Once that process is finished, the command *git push heroku master* will upload the code and create a new app that can be found on a subdomain provided by Heroku. Note that the *.env* file configured locally will not be uploaded for security reasons and it needs to be regenerated using Heroku's own *Config Variables.* Heroku offers an automatic build and deploy method by binding a GitHub repository to itself and when a change is detected, a new version of the app will be built without the need for manually pushing to Heroku using git commands.

One important factor to be considered when employing this type of server (PaaS) is that for every build, all the temporary files, cached data or media files that are untracked by the versioning system will be erased. This is a big disadvantage because third party suppliers need to be integrated and most of the time that adds more expenses. The main features that needed change were the media storage and database. We need to note that we're looking for sustainable resources at the lowest cost. Regarding database storage, Heroku offers some add-ons that can be integrated. The issue is that after a certain amount of database storage (5MB) the cost per month will no longer be zero. Considering this and the fact that the database version and backup are not under our control, this type of server was better suited as a testing ground and not a production one.

JawsDB was the add-on chosen and after adding the config variables to Heroku, MySQL Workbench was used to connect remotely through TCP/IP using the credentials provided by the add-on. A new database was created and added to the *config variables*. The media storage however needed cloud support as the app will grow and the storage is limited no matter the type of server that is going to be used. AWS S3 [9] was the perfect choice for us, as it offered free services under a certain amount of usage and once the app will surpass that threshold, the prices are low and based on usage. To set it up, a new account was created and the S3 service was selected. Within it, a new bucket was generated that keeps all the media files. To access it remotely through our app some special credentials needed to be generated. Identity Access Management (IAM) is a service provided by AWS for this purpose. A new user was created with the role *AmazonS3FullAccess* and a new access key was generated. Those keys were also added to the local *.env* file and Heroku's *Config Variables.* To swap our Laravel app from using default media storage to AWS S3 the *league/flysystem-aws-s3-v3* package was installed using Composer. The *FILESYSTEM_DRIVER=s3* environment variable was added and the accessibility of future to be stored files was set from private to public. This change is important as the images won't be visible when displayed within the app or when accessed by link.

C. DigitalOcean server

To avoid costs related to database and server hosting as demand and traffic rises, DigitalOcean has been chosen for its IaaS type servers called droplets. DigitalOcean offers 1GB RAM and one vCPU for 5$ monthly while Heroku offers the same performance starting at 50$ on their PaaS servers. Heroku offers backups, fast deployment and security by default but because that cost is significantly higher than what is considered a cost-efficient application, we tried to create those features from scratch on a droplet using a Ubuntu Linux image.

Once the droplet is available, we will use MobaXterm to connect through SSH using the credentials provided by DigitalOcean. The connection is through port 22 as root user and a password. The most important security changes that can be made refer to changing the default connection port and closing all unused ports. After we've done that, a new user was created and it was given sudo rights. Once that was done, the root user was erased altogether to prevent unwanted vulnerabilities. The last change was made to the way the user is able to connect to the server. A public-private login key was uploaded and all types of logins by password were disabled. These changes do not guarantee perfect security but they definitely make it harder for unwanted access to occur. One other step that can be taken in this direction is to use the proxy services provided by Cloudflare, most of which are free. Cloudflare offers a content delivery network (CDN) alongside multiple layers of security and we used it to bind the domain name to our DigitalOcean server.

Once the security settings are in place, we can configure the newly installed Ubuntu server. We decided to install the LAMP (Linux+Apache+MySQL+PHP) type of packages which is suitable for our Laravel app. After configuring the virtual host once again on the droplet, we've updated PHP extensions located in *php.ini* file and created a new database using MySQL CLI commands. The next step is to download the project code stored inside the GitHub repository. Using git commands, the project was stored in the directory specified in the virtual hosts' procedure above. Once the download finished, *composer install* and *php artisan migrate* commands were run to fully setup the project.

The admin dashboard has been implemented using a Laravel package named Voyager. It was downloaded using composer and installed by running the command: *php artisan voyager:install.* The Voyager admin routes were added to the *routes/web.php* file so the */admin* route will work. The next step is to generate the database schema that will map the users and the Voyager components. By running the command: *php artisan db:seed --class= VoyagerDatabaseSeeder* the default dashboard elements will be generated. To generate a new admin user the following command will be run: *php artisan voyager:admin admin@email.com –create.*

We wanted to create a panel from where the administrator will manage the content displayed to the frontend and a method to manage the messages sent via contact forms. To implement such functionality a migration was created containing data about the price offer request: name, email, phone, car make, mode, year, vin, details. A model to map the database structure was also added. Using Voyager, we added some CRUD methods on this table. Email notifications were also integrated such that when a new request is submitted, an email with a link to that certain request is sent to the administrator. By using composer to install the guzzlehttp/guzzle package and configuring the environment variables on all servers, our application was capable of sending emails with email templates of our own choosing.

The client side is constructed with Laravel blade components that implement code written in HTML, compiled SASS, JavaScript and jQuery. Forms were implemented to gather the customer requests. Each form has frontend and backend validation of fields. Beside the inputs, we've implemented a captcha field that requires the user to solve a simple math problem before submitting the form to ensure the senders are real people. A controller that validates the answers and accordingly responds with notifications was created.

Because the application will be used in a real automobile service, Google Analytics was integrated to track the type of users and their habits. This was done by creating an account, generating a tag manager key and then adding the JavaScript code, provided in the documentation, inside the *main.blade.php* file.

During the development stage, several difficulties were encountered in the process of configuring the servers and publishing the applications. Even if the documentation is well structured, the elements used in the project involve modifications and packages that are not included in a standard configuration. The implementation of Google Analytics has also been cumbersome due to the lack of clear documentation and the security changes made by Google in recent months that have made components unusable. Another unforeseen variable in the development process was the media files. After integrating the Heroku server, it was noticed that the uploaded images were lost, being deleted at each start of the service. Thus, the integration of AWS S3 services was needed.

V. EXPERIMENTAL RESULTS

The following experimental results will be presented: (A) User point of view, (B) Administrator point of view and (C) Application comparison with competitors.

A. User point of view

The application has a graphical interface through which the user can find out information about the services and availability of the company. This interface also provides methods of communication and feedback for users. There are 4 sections that make up the main page: 1) Introductory image with the company's motto (Figure 2), 2) Short description with a button to the contact form, 3) The service panel that at the push of a button expands and provides details, and 4) A testimonial slider that changes automatically every few seconds or via buttons (Figure 3). These sections present the most relevant information for the customer and present multiple buttons that send the user to the contact pages in order to make as many conversions as possible.

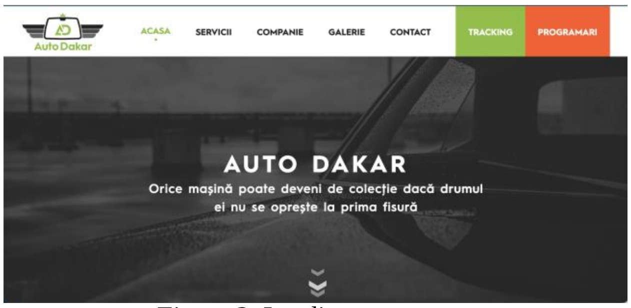
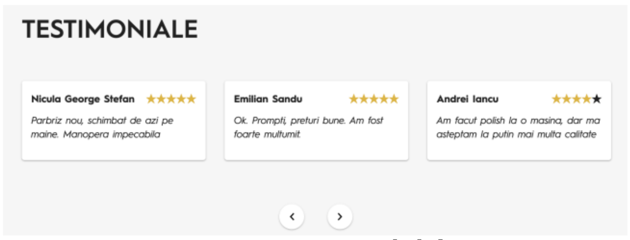

Auto Dakar
ACASA
SERVICII
COMPANIE
GALERIE
CONTACT
TRACKING
PROGRAMARI
AUTO DAKAR
Orice mașină poate deveni de colecție dacă drumul ei nu se oprește la prima fisură

Figure 2. Landing page motto


TESTIMONIALE

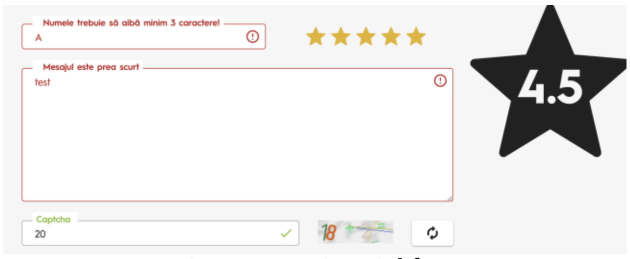
Nicula George Stefan ★★★★★
Parbriz nou, schimbat de azi pe maine. Manopera impecabila

Emilian Sandu ★★★★★
Ok. Prompti, preturi bune. Am fost foarte multumit.

Andrei Iancu ★★★★★
Am facut polish la o masina, dar ma asteptam la putin mai multa calitate

Figure 3. Testimonial slider

Figure 4 shows one of the forms implemented in the application. The testimonial form contains, in addition to the personal data and message fields, 5 stars that can be selected by users to evaluate the quality of services. On the right side of the form, the average of the points received is displayed.


Numele trebuie să aibă minim 3 caractere!
A

Mesajul este prea scurt
test

Captcha
20

Figure 4. Testimonial form

Within the contact form, depicted in Figure 5, personal data is validated for the existence of at least one contact method, either email or telephone number, of the applicant.

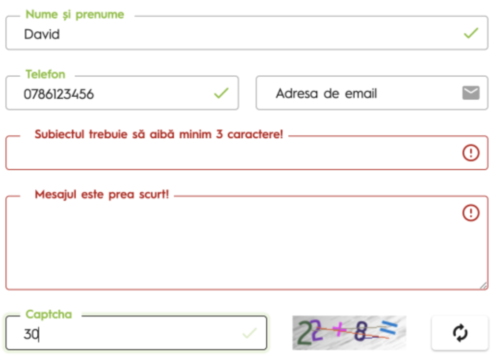
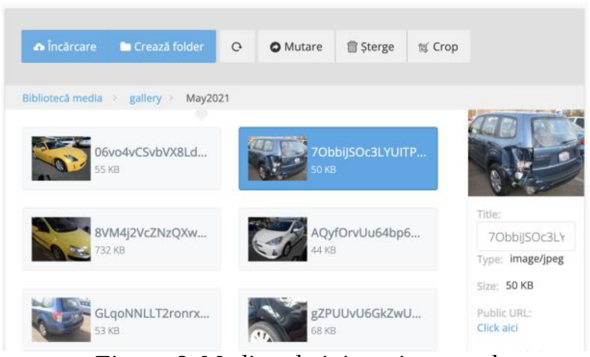

Nume și prenume: David
Telefon: 0786123456
Adresa de email
Subiectul trebuie să aibă minim 3 caractere!
Mesajul este prea scurt!
Captcha: 30

Figure 5. *Contact form*

B. Administrator point of view

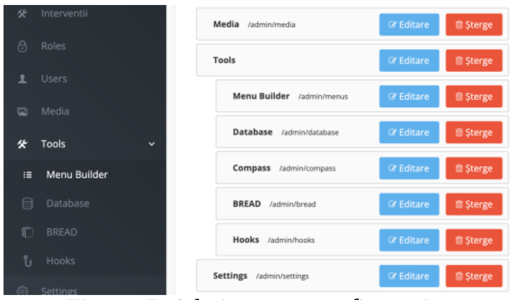
The administration panel offers accessibility, flexibility, and allows for content customization. The images and fields in the different forms are intuitive in order to facilitate the learning and accommodation process. Accessing the administration panel requires authentication. Each field is validated and the *remember me* option allows for quick authentication. Media file management is facilitated through a dedicated panel connected directly to the AWS S3. This panel, presented in Figure 6, depicts the images from the gallery. By selecting an image, its characteristics can be identified (size, type, and link).



Title: 7ObbIjSOc3LY
Type: image/jpeg
Size: 50 KB
Public URL: Click aici

Figure 6. *Media administration panel*

Figure 7 shows the menu editing page. Items that are indented are child items of menu items visible in the menu on the left. The menu can also access the pages that are used in the process of managing content within the user interface, but also the pages where the customer messages are displayed. Content management can be done from the following pages: Galleries, Offers and Partners. Messages and requests for quotation are referred to as messages and appointments.



Media /admin/media Editare Șterge
Tools Editare Șterge
Menu Builder /admin/menus Editare Șterge
Database /admin/database Editare Șterge
Compass /admin/compass Editare Șterge
BREAD /admin/bread Editare Șterge
Hooks /admin/hooks Editare Șterge
Settings /admin/settings Editare Șterge

Figure 7. *Admin menu configuration*

Gallery management can be performed by accessing the gallery editing form presented in Figure 8. This form includes the title, a field for selecting the date, and two fields through which images can be uploaded.

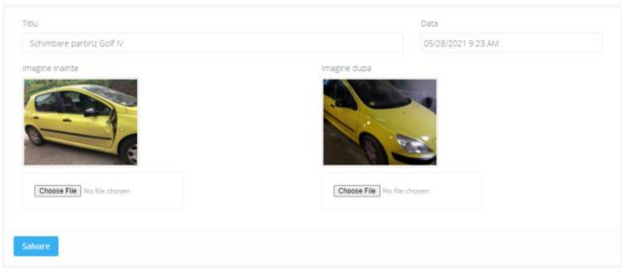
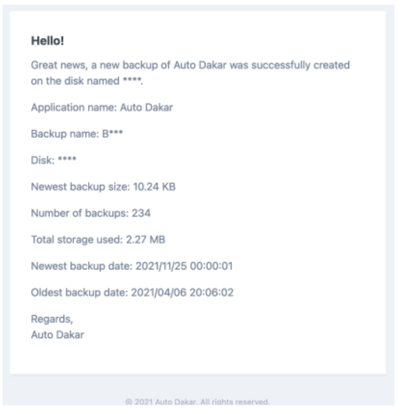

Titlu: Schimbare parbriz Golf IV
Data: 05/28/2021 9:23 AM
Imagine inainte
Imagine dupa

Figure 8. *Gallery editing form*

The administrator will receive email notifications regarding multiple events such as: new message, request for a quote, new testimonial or report on the status of the backup process. The email announcing the backup contains information about the date, file size, and storage method as seen in Figure 9. Unlike the other types of emails, which contain user-sent data and buttons that lead to the administration page, this one is informative and does not require any additional action.


Hello!

Great news, a new backup of Auto Dakar was successfully created on the disk named ****.

Application name: Auto Dakar

Backup name: B***

Disk: ****

Newest backup size: 10.24 KB

Number of backups: 234

Total storage used: 2.27 MB

Newest backup date: 2021/11/25 00:00:01

Oldest backup date: 2021/04/06 20:06:02

Regards,
Auto Dakar

© 2021 Auto Dakar. All rights reserved.

Figure 9. *Backup email notification*

User activity can be tracked in Google Analytics. Figure 10 illustrates the report from the last 7 days in which there was an increase of 200%, the figure reaching 6 users of which 4 are here for the first time. Also, the average time spent on the site is 9 minutes and 4 seconds and, on the right, it can be observed that no one has visited the application in the last 30 minutes. By applying filters and generating multiple types of tables, users' habits and preferences can be identified.

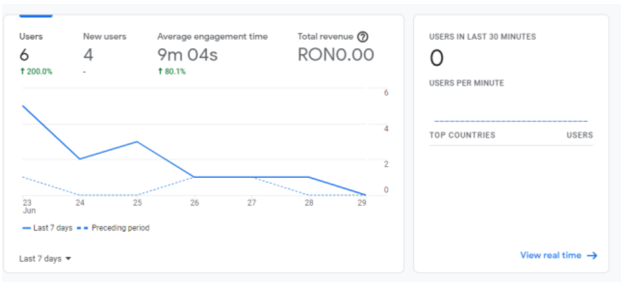

Users 6 ↑ 200.0%
New users 4 -
Average engagement time 9m 04s ↑ 80.1%
Total revenue RON0.00
USERS IN LAST 30 MINUTES 0

Figure 10. *Google Analytics report of last 7 days*

C. Comparison with the direct competitors

The results obtained from the analysis of web applications reflect a very good positioning of the AUTO DAKAR application on the target market. Figure 11 presents the scores obtained by the competitors organized by color: yellow for the companies located in Alba Iulia, orange for the companies from Alba County, green for the top companies with online presence and purple for our project. AUTO DAKAR achieved above average results and a considerable difference from the competition. It obtained the first place in each of the 4 categories analyzed: *content*, *design*, *organization* and *functionality*. Each category had multiple questions regarding the topic. The companies were rewarded with 10 points if the answer was true or 0 if it was false. For the *performance* and *indexing* metrics, the points awarded were from 0 to 10 based on the results provided by multiple testing websites used for this purpose. The cumulative results show an average of 286.55 points out of a total of 500 possible, which indicates a level of satisfaction slightly above the mid value of this analysis but unsatisfactory for users. The important goals of a site are to represent the company online, produce conversions and increase trust in the products and services offered. These cannot be achieved if the page cannot be found by customers. Following the analysis, it seems there is a low interest in search engine optimization (SEO) and indexing methods. At the same time, companies provide details about services, location and offline contact methods to the detriment of online communication and user education. The lack of interaction with the user and the lack of a method to find out additional information on the website can cause a decrease in trust in the services offered or even the loss of the potential customer.

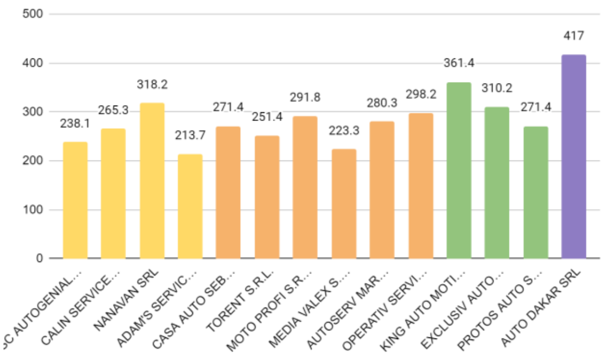


Figure 11. Web pages analysis results

However, it should be noted that the number of companies that choose this form of promotion is very small in the car services market in Alba County. The market is very competitive and any advantage counts. According to the analysis, only one company in the target group in the city of Alba Iulia has a site with above average results, and in the study only one company exceeded the threshold of 350 points. These results indicate an opportunity for development and orientation towards online consumer conversion and the proposed application can meet this need.

VI. CONCLUSIONS AND FUTURE WORK

The situation of the Romanian car market determines many entrepreneurs to start businesses in the field of vehicle maintenance, thus causing fierce competition between companies. A common strategy for companies to increase market share is promotion through recommendations. This however may prove to be a long process with high risks given the evolution of the market. Other companies choose to create an online presence through a website, but few understand its role and how it should be configured so that its impact is as large as possible.

This paper presents a method of building a web application with minimal costs and multiple functionalities in order to have an impact on the online environment. The web application, called AUTO DAKAR, is based on the Laravel framework, built with PHP, and combines the following elements: an Admin package, called Voyager, a MySQL database, an Apache server, DNS and Cloudflare proxy, domain name, AWS S3 storage, and Google Analytics. The application was built on a virtual local server, tested on a PaaS server and made available online on an IaaS server. The structure of the application not only allows for easy addition of new features but it also means that it can easily be updated and maintained for a long time at minimum monthly costs.

Future work involves the integration of Jenkins, for automating the code update process, and Docker containers and load-balancing methods for better resource management. In the administration panel, a digital document management engine will be added, alongside an intervention tracking system, which will help clients get the real-time status of their car.

REFERENCES

[1] TopFirme, „Top firme cod caen 4520 - Intretinerea si repararea autovehiculelor, Romania," TopFirme, 2021. [Interactive]. Available: https://www.topfirme.com/caen/4520/ [Accessed: June 24, 2021].

[2] A. Negrescu, „Analiză KeysFin. Tu unde îți repari mașina? Cum a ajuns România țara service-urilor auto," KeysFin, 17 10 2016. [Interactive]. Available: https://www.keysfin.com/EN/#!/Pages/News/NewsDetails&title=analiza-keysfin-tu-unde-iti-reparimasina-cum-a-ajuns-romania-tara-service-urilor-auto [Accessed: June 24, 2021].

[3] N. Prokofyeva and V. Boltunova, "Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems," Procedia Computer Science, vol. 104, pp. 51–56, 2017, doi: 10.1016/j.procs.2017.01.059.

[4] S. Delcev and D. Draskovic, "Modern JavaScript frameworks: A Survey Study," 2018 Zooming Innovation in Consumer Technologies Conference (ZINC), May 2018, doi: 10.1109/zinc.2018.8448444.

[5] S. S. Wagle, M. Guzek, P. Bouvry, and R. Bisdorff, "An Evaluation Model for Selecting Cloud Services from Commercially Available Cloud Providers," 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), Nov. 2015, doi: 10.1109/cloudcom.2015.94.

[6] M. Saraswat and R. C. Tripathi, "Cloud Computing: Analysis of Top 5 CSPs in SaaS, PaaS and IaaS Platforms," 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), 2020.

[7] „Laravel - The PHP Framework For Web Artisans," Laravel.com, 2011. [Interactive]. Available: https://laravel.com/ [Accessed: April 14, 2021].

[8] „DigitalOcean – The developer cloud," Digitalocean.com, 2013. [Interactive]. Available: https://www.digitalocean.com/ [Accessed: April 14, 2021].

[9] „Amazon Web Services (AWS) - Cloud Computing Services," Amazon Web Services, Inc., 2012. [Interactive]. Available: https://aws.amazon.com/ [Accessed: April 14, 20211].