Development of a prototype of a web tool for sharing and accessing data

Deniss Akmaikin

2021 Laurea

Laurea University of Applied Sciences

Development of a prototype of a web tool for sharing and accessing data

Deniss Akmaikin
Business Administration
Thesis
May, 2021

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Technology
Bachelor of Business Administration

Deniss Akmaikin
2021
Year 2021 Number of pages 43

The objective of this thesis project was the development of a fully functional web application, named “Project Instance”, which will help the client to optimize time and resource spending while transferring data from the start point to the end-user. The client is BeGo, a company which provides smart solutions using sensors and powerful data analysis tools. The development task was to create a pilot version of such a web application from scratch using open-source libraries and classic web development tools. As a part of the thesis, research about development methodologies was carried out. RAD, Agile and Waterfall methods were compared and it was found out that Agile/Scrum was the most suitable method for the development process in this thesis project. Development was done using a web stack of HTML/CSS, JS, PHP, SQL, Bootstrap, JSON languages. As a result, a fully functional pilot version of a web application was created with user-end and administrator-end functions. The piloting version was sent to the customer as a deliverable with accompanying documentation. The pilot version is ready for the next development stages and has a place for improvement. As a part of this thesis, research about development methodologies was carried out, RAD, Agile, and Waterfall methods were compared in terms of this thesis development and it was found out that Agile/Scrum is most suitable for this development process.

Keywords: development, web application, open-source libraries, scrum

Contents

1 Introduction

1.1 Background and thesis content

My client is a Finnish startup BeGo, which provides smart solutions for various business systems - from smart offices and homes to manufacturing equipment. Their smart solutions consist of several services: selection and installation of sensors, maintenance of sensors, and subsequent processing and analysis of data obtained with the help of sensors.
BeGo already has several customers in its history, for whom unique smart solutions have been developed using various sensors - from those that detect the frequencies of sound and vibration to those that detect the presence of people in the room. The analytical solutions of my client are advanced functions - they provide their solutions for clients on many parameters, for example, setting the correlation between air pollution and brain activity.

My client's services are in two areas - hardware and software. In the first case, my client offers analytical solutions that include not only data analysis but also their visual wrapper, which in the first case, in turn, assumes the presence of a dashboard, visual elements. In addition, in both cases, my client provides their customers with support and service for smart solutions, which in turn is one of the key reasons why they need an order management system.

This thesis covers the basic concepts of developing a small web application using open source libraries for data integration. The project was developed alone by a student of Business Information Technology based on the knowledge gained during the training. The main idea of the thesis was designed and set by both student and client based on the number of potential resources (time and money).

The final product of this thesis is a pilot version of the "BegoSolutions Project Instance" product - a web version based on classic web hosting and built using a standard web stack for web development. The application has a set of functions for this type of development, for example, editing and accessing data, sharing and viewing the project.

The thesis can be roughly divided into three parts. First: the theoretical part, which discusses the current services that provide services similar to those that were asked by my client, this part talks about existing services, their advantages, and disadvantages, as well as the reasons why they are not suitable for my client. It also tells about the reasons why the client decided to contact me with the task. In addition, one of the key elements of this part

is a discussion of popular development methodologies and a discussion of personal experience using development as an example. At the end of the first part, we discuss the potential data that will have to work with, and how it can be used and implemented in the project, as well as how to test the project.

The second part of the thesis is more practical - it discusses development issues: which web stack was chosen and why, what were the nuances when choosing this stack, and what tasks are planned to be implemented using a particular programming language. In addition, in the second part, the process of development and version control of the product is considered in detail. Also, in the second part, ideas about post-production and possible project improvements are presented. In the conclusion of the second part, difficult situations that arose during development, as well as the methods and results of the applied testing (including feedback from clients) are discussed.

The third part is the final one - it discusses the development results, summarizes the results (based on the tasks set: what happened, what could be improved or done differently), it also discusses the client's feedback.

Many modern smart solutions (not only in the niche area where my client works) implement visual support elements - mobile applications or websites. So, for example, banks, which are transferring more and more visual hints online with responsive applications (for example Nordea Wallet, displays spendings in real-time), because they allow you to personalize their applications according to the needs of clients, methods and helping the client to understand. intricacies of services and prompting him at the right time. Such solutions are often used by large companies that install the element of popular services, but my clients are needed (specific data and methods of their transfer), which do not allow using the services of any services.

1.2 Purpose of the development and scope

The reason why my client approached me with a development proposal was that the existing systems did not offer services that can adequately meet the requirements of the client. The reasons were different (more on the services and reasons in the following chapters) - from the financial side to an insufficiently wide range of possible implementations and flexibility.

From the very beginning of negotiations with my client, we agreed that the final product of development will be only a pilot version of a full-fledged application, many of the functions will have limited functionality or lack of flexibility, but the developed base should

be sufficient so that later client can independently (or with the help of third-party developers) continue to develop the application and existing functionality (as well as be able to supplement it). Several fundamental functions have been identified as «key features» during development (these functions will be discussed in the next chapter).

Before starting the development, it was necessary to understand whether it was possible to use any existing services within the framework of this project: for this, a sample of large services was made and an analysis of their strengths and weaknesses was carried out, after which decisions were made if usage of service is possible or not.

The main goal of the development is a system called “Bego Solutions Project Instance”, which is based on the existing website of the company. The web application was conceived as a system that would allow both my client (BeGo) and those to whom they sell their services to use it for their convenience. So, on the part of the user, it was necessary to show the available data in real-time, make it possible to connect other users to the current project, and display all relevant information. From the point of view of the administrator, it was necessary to provide the ability to connect data integrations and provide the ability to edit current orders - for example, add information about the current installation progress or add documents with instructions.

System flexibility is one of the important elements to consider when developing. Since my client works in different industries and uses different devices when creating their smart solutions, the sources of data acquisition (as an intermediary point between devices and application) may be different, in this regard, it is necessary to consider a flexible system for transferring data from the indexing point to the point of display.

Thus, an application like this should help my client maximize the speed of data transfer from the indexing point to showing to the end-user. The detailed data model will be discussed later.

Ultimately, the project can be presented as a development based solely on backend technologies, the final product of the thesis (pilot version of “BegoSolutions Project Instance”) does not imply the final (product/release) version of the frontend-development part (UX/UI), the presence of bugs or technical faults are admissible. It is also allowed to cut the minimum functionality within the framework of the agreements with a client (BeGo).

As an additional task, there is the need to create technical and user documentation for the further use and development of a web application. It is assumed that after the delivery of the project, its further development will be carried out directly by the client (Bego) or by a

third-party developer, in this regard, it is necessary to create technical documentation that will cover the main nuances of the system functions. In addition, it is expected that user documentation will be created to allow them to quickly become familiar with the system and start using it.

Additional attention in the development will also be devoted to the integration of third-party services (which will be discussed later), which are used for data transfer, this is one of the key points in development, which is used in most of the functions of a web application.

As additional research within this thesis, a comparative analysis of some of the popular application development methodologies will be carried out. Because the development of this application will take some time, an experiment will be set up to apply those methodologies for teams with only one developer.

2 Theoretical parts

2.1 Benchmarking of existing services

At the beginning of the practical part of this thesis, a comparative analysis of several existing services that exist for project management purposes will be carried out. A CRM system for project management and development will also be considered. All services will be considered from the point of view of the possible implementation of all the necessary details that were requested by the client (BeGo), the financial side, and from the point of view of the unique functionality (if any). This benchmarking is done to confirm or deny the need for a complete system development process from scratch.

Benchmarking is the analysis of products (in this case: online services) based on certain parameters (in this case, based on customer requests) to find the best solution without resorting to practical application. The bonus of benchmarking is that it can be done before the development cycle begins, thereby saving time for theoretical preparation.

2.2 Comparison of development methodologies

Starting any development, you need to understand how the project development process will take place - how often you will have to have feedback from the client, how often tasks will be updated, you also need to understand the general systematic approach to development. When you work in a team, it’s all much easier, but when you develop something large alone, it’s a little more difficult to develop a systematic approach, because

it’s impossible to distribute tasks within the team, to shift deadlines — you have to rely solely on yourself. In this project, it was understood that the systematics of development and tasks would be periodically updated, some difficulties would be added, and versions would be iterated, which in turn would add both problems and positive features in development (quick feedback is a useful tool for correcting technical deficiencies or making any changes if necessary). Even though there was only one specialist in the development, the team wanted to have a certain system or methodology that would allow us to work effectively and communicate with a client, and at the same time maintain technical documentation, that is, to be multitasking in a tight time frame. For this, we had to choose one of the currently popular methodologies. Below briefly described each of them and told how acceptable it turned out to be for us within this project.

2.2.1 Agile

Agile is probably one of the most popular methodologies among IT companies in the world today. We believe that many teams adhere to the basic principles that were described in the Agile Manifesto (2001) - the primacy of personality and iteration, emphasis on software and development, customer interaction, and response to emerging problems. This approach allows teams (or me, in this case) to quickly respond to minor issues that arise during development. Thanks to the existing sprints (short development times spent on creating a version/function) and a large number of iterations, the developer and the client can be in constant communication and look for the best solutions to implement their ideas. In development, it is important to pay attention to small details that may initially seem insignificant. So, a mistake made during implementation at the beginning of a project can remain in the release version and bring with it many problems, but the Agile methodology allows developers to avoid such problems due to constant iterations. In addition, due to this approach, the development process becomes very flexible, due to which the client can easily introduce changes to the terms of reference and/or requirements, and the development team can easily adapt them (if the function has not already been done). According to Sacolick (2020), it is precisely this flexibility and possible adaptive approach to development that lies the main advantage of Agile development. For many teams, Agile is the only right choice because of the principles of its work: the team is forced to interact with each other (which also improves the team atmosphere) to quickly produce a quality product, in which case the tasks are distributed more responsibly based on the skills of each individual team member. But in this project, it was the only developer who participated in the development - therefore, the distribution of tasks was not based on skills, but on the potentially spent time. We could expect that some of the tasks would be completed faster than others - so we gave them a higher priority (the faster they are completed, the more time we will have for other

tasks, in the ways of implementing which we are less confident). In addition, we realized that the sooner we submit tasks and versions of the project, the faster we can send them for review to my client, get feedback and, if necessary, fix something. Thus, the list of priorities and urgent tasks was constantly updated, and my client could see the results of our work and, in general, remain satisfied with the development progress. According to a study by Abrahamsson, Salo, Ronkainen, Warsta (2002), Agile methodology distinguishes several development methods, among them, for example, extreme, scrum, crystal family, and others. In general, all of these methodologies fit the principles of Agile and have minor differences - mainly it concerns the division of projects into phases. Extreme programming splits the project into five phases, scrum - into 3. In my case, there were 3 phases - like in Scrum. In the first phase, the necessary task is established - whether it is the development of a function, the correction of a shortcoming, or a large amount of work on the little things. After drawing up the plan, in the first phase, the initial (theoretical) architecture of the product is built, and the development backlog begins. In the second phase, the direct development already begins - the sprint development cycle is launched, which includes several tasks: direct development, testing, design, and delivery of the new version to the client. In the final phase of Scrum development, the product undergoes final testing by the client, and there are two outcomes: either the product is sent for revision (then the backlog and the task are updated) and the process is restarted, or the product is approved and goes into a release, and then additional documentation is created for it.

To summarize: Agile methodology in development is the most flexible system that allows both clients and developers to perform their functions quickly and adaptively, while not losing the quality of the product.

2.2.2 Rapid

The Rapid methodology, like Agile, was at one time created to replace the Waterfall methodology since it is outdated. Rapid first appeared in 1991 when James Martin proposed using RAD (Rapid Application Development) instead of the outdated Waterfall model. RAD tried to fix bugs and speed up the development process and give developers the ability to do more iteration. In some elements, Rapid can be called Agile-like, or even said that Rapid is the predecessor of Agile methodologies. But even now, 30 years after its development, RAD is changing and remains popular in some quarters. Singh writes (2019) that there are five main phases in the RAD methodology: analysis and design, development, testing, implementation. Each of the phases assumes close collaboration between the client and the development team, and the ability to iterate in projects is also very important in RAD. In general, RAD is often viewed as a methodology that exists for teams (or even several teams involved in one project) and which need to develop a full-fledged application soon, Agile, for example, is considered in other cases - when it is necessary to develop not a whole

application, and part of it. Thus, RAD here stands out against the background of Agile precisely by its globality and, one might say, even by the necessary haste. Some studies mention that RAD has a strict hierarchical model (An introduction to RAD, 2009), which is not found in Agile methodologies. So, each stage in RAD is divided into tasks, tasks are assigned a responsible and strict plan (this RAD took over from Waterfall). The responsible team for the task breaks it down into subtasks to speed up the development of individual functions. In addition to direct development, RAD is sometimes referred to as the prototyping stage, for example, in a study by Jones, Richey (2000), there is an important advantage of using this approach - rapid prototyping allows you to apply product iteration at an early stage, thus you can shorten the development cycle and spend more useful time to develop the final product and not its intermediate version. One of the shortcomings of the RAD methodology is poor documentation work. In many projects, the development phase with RAD does not imply post-work with the project and the creation of any detailed documentation, in connection with which problems often arise, for example in the study of Beynon-Davies, Mackay (1999), which emphasized that when working with projects that were created with RAD create very little documentation, which interferes with research. However, it is worth emphasizing that from the point of view of the developer, such a model takes place - in this way the developer can be focused on his tasks. Even though this development model is still very hierarchical for teams (everyone does their own thing), it allows you to speed up the development process and allows you to introduce iteration into projects. And although it has some drawbacks, it is still worth emphasizing its advantages: quick introduction of the project into the development phase, the stage of rapid prototyping, which allows you to shorten the development cycle and quickly start creating the final version of the product, the ability to conduct iterations and a development backlog. In addition, this approach allows all decision-makers to take part in the final decisions because the hierarchical model of the methodology allows each team member to participate in the development sector where he is most useful. Despite all the similarities with Agile, the RAD methodology still remains separate and has several striking differences.

Summing up: RAD, despite the fact that it was introduced a long time ago, is still relevant for companies that are forced to carry out their development in a short time. RAD focuses on more global tasks than Agile - this is their main difference.

2.2.3 Waterfall

The waterfall model is one of the oldest development methodologies. Its name and basic principles were defined in the article by Royce (1970). The basic principles of this methodology can be defined as follows: each stage must go from start to finish, it must not be skipped and every step must be documented. A clear hierarchy of tasks in the project must be completed in full and only then, at the end of everything, the product can be shown to the

client. If something needs to be changed, corrected, or the client has sent a request/demand for changes, then it is necessary to wait until the end of the development cycle and then, having revised the terms of reference, the entire development cycle must be started anew. Also, according to Lotz (2018), several clear stages can be distinguished in the methodology - design, during which the team collects and prepares the necessary documents (some teams also conduct analytics during this stage). Then comes the development stage - the teams create a function or product, according to the terms of reference, the Waterfall methodology does not imply the possibility of changes during the project (even for the sake of optimizing the product or reducing the resources spent in the form of time or money). Then comes the testing and client approval stage, followed by minor fixes. As McCormick (2012) writes, the Waterfall methodology is linear and this is expressed in the approach not only to development but also to business management. So, for example, during Agile development, one individual developer may decide to implement a different way of solving the problem, if this way it will be possible to shorten the development cycle, such decisions cannot be made in the Waterfall model, otherwise, the documentation will be violated. In addition, iteration in such a development model cannot exist, since iteration implies the ability to return to a specific development layer and instantly make any changes, this is almost impossible in the Waterfall model. Therefore, the main difference of this model from others (such as Agile) is the lack of flexibility. Moreover, the lack of flexibility is thought out in advance. Of course, Waterfall is a model for developing new products, probably a risky business - because when developing something new, additional difficulties and tasks always arise, and Waterfall does not imply the ability to quickly resolve new issues, because problems by the end of development can force the team to spend more time to solve them than for a new development, which is why the development cycle can increase indefinitely. The Waterfall model is suitable for those teams that have a clear plan, a clear technical assignment, and are confident in their toolbox, then everyone knows what, how, and when he should do - in this regard, the Waterfall model can be perfect.

To summarize: The Waterfall model has absolute differences from Agile or RAD - it is not flexible, does not allow iteration or instant use of client feedback, but it has its advantages and some projects can use it as the main methodology.

We believe that for us on this project, given that we have the only developer, we need flexibility and the ability to iterate during development. Therefore, the use of Agile / Scrum seems to us the most logical choice, in favor of which the excellent capabilities and flexibility of these methodologies speak.

2.3 Data service implementation methods

After the client and developer outlined a clear list of potentially possible data in the “project instance” system, we began to select potential methods in order to implement the data into the system. We needed to take this step before actual development because planning resources and choosing the right libraries could affect the final development process.

First of all, it was necessary to correctly integrate into the PowerBI Dashboard system, there are several reasons for this: firstly, this is the most important detail of the entire project, for which, in fact, it is created, and secondly, this is the most difficult function in terms of integration and optimizations - even though all calculations are done on the Azure side, this is still a very “heavy” element for hosting, which needs to be constantly loaded, updated and even hashed. Moreover, in theory, this element will be quite difficult to make responsive because its graphical component is "drawn" not on the site side, but on the Azure side, and we, as a service that integrates and uses this function on a third-party server, have only small possibilities for visual modifications. There are several variations for the Dashboard implementation that we, as developers, could have resorted to: firstly, Azure itself provides possible integration to third-party sites using the classic CSS + HTML + JS set. In this case, we may face some optimization problems (then most of the load will fall on our server, and Azure will act as a data store and send it on request, that is, at the moment when the client enters the service). At the same time, we will have a great opportunity when setting up and modifying the appearance, which will allow us to cut off some heavy elements and help with optimization, moreover, with this method it will be easier to adjust the responsiveness of an element to be displayed on different screens. The second method, which, just like the first one provided by Azure, will use integration - although in this case Azure does not provide libraries, but provides an embedded element that can be integrated into the site. In this case, the optimization issue is removed - after all, everything that happens on the site is a simple request-response algorithm, where as a response the site receives a visualization of a graphic element and not the amount of data for visualization. But then we lose the opportunity to optimize the appearance of the element, and to act on it exclusively using the tools available in Azure. In any case, as integration for pilot testing, the most important element is optimization and sustainability, the UX / UI for a pilot product in our case is a secondary element that is not worth losing stability. In this regard, for me, as a developer, the second method with an embedded element seems to be the most logical way out for integrating powerBI into a project.

Integration of other types of data does not seem to be a problem, since these types of data are quite classic for developing web applications. Text and graphic elements, like documents, are used everywhere, so it was not difficult to find ways to integrate them. For

text data, the obvious and only correct solution from the point of view of usability, impact, and editing, as well as readability from the server-side, is to integrate such data into a local database. Due to the fact that the database and the site are on the same hosting and server, access and reading of data will occur instantly without delays, which will allow the site to load quickly and display data for the client. Moreover, textual data is very easy to visualize - thanks to this, it will be possible to customize the skeleton of the visual structure of the site and optimize it for different devices. Working with textual data sets can cause only one potential problem - variations with encodings. The site plans to use UTF-8, which is standard for many web applications, but the local database runs on UTF-16, which is a non-standard solution, so you will have to find a way out to work with different encodings. Another interesting problem in the integration of text data is the issue of “layers” of texts. For example, when building a project instance of a system, we want some data to exist inside the system, but not be shown to the user and be hidden both from him and from the server (so that they cannot be accessed through the browser console, for example, F12 in Google Chrome). Previously, a schema with text data looks like this:

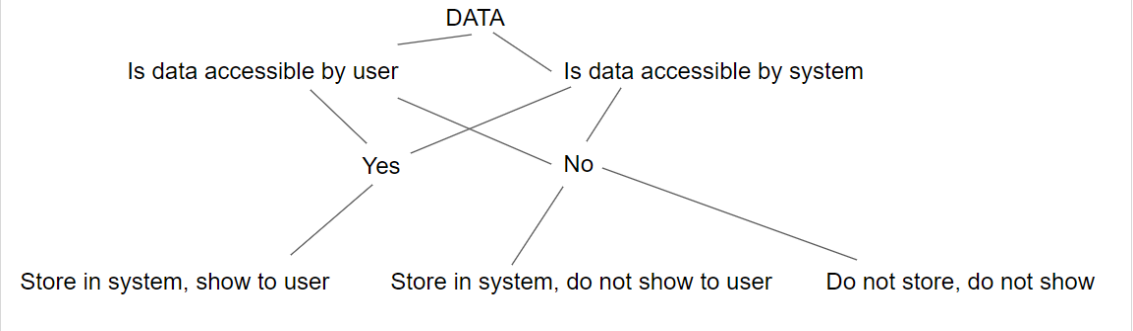

Figure 1. Text Data Layers

Thus, we see that in theory, it is possible to split the data and their integration into 3 layers in our case: in the first case, if the data is available to both the system and the user, we store it in the user's internal session and display it. In the second case, if the user is forbidden to see information for some reason, we save it inside the session, but turn off its display. In the third case, when the user and the system cannot access it for certain reasons, the data is not saved in the session and is not displayed by the user. Such a layer system is needed to personalize the application and make it more flexible to work with all types of data.

The last type of data is documents. There are certain difficulties with documents, but the general meaning and algorithm of their access are very simple: my client uploads the document> it is sent to the local storage to the site's hosting> its address is uploaded to the database> the generated link is sent to the end-user and displayed to download the document. But in the case of documents, the main problem is not the problem

with access, but the problem with their loading: hosting services often impose regulations on data loading - for example, restrictions on weight or file extension. In the case of a project instance, these restrictions will have to be taken into account - the system can potentially accommodate different extensions - .docx, .csv, .txt, .pdf - and their weight can range from very small (for example, documents in .pdf) to large (data in .csv can sometimes be up to several gigabytes in size).

As a result: in theory, the integration of all types of data that could potentially be needed when creating a project can be carried out directly through internal hosting services and using a database. In addition, it is expected that there will be additional opportunities to expand possible integrations should such requirements arise. A more detailed integration process and the final selection of models and integration methods will be described below in the chapters with a detailed description of the development. The problems encountered and possible solutions will also be described. However, data integration and its methods are considered in this project solely as a way to include them in a pilot project, and not as the final final version of the release version of the product.

2.4 Piloting opportunities

One of the important elements in the development of this project was the possibility of immediate testing. Such testing helps to quickly identify and correct found technical problems and deficiencies. For testing and piloting, we had two options.

The first possibility is internal. I, together with the BeGo team (3 people worked with me) tested ready-made functions. But the second opportunity turned out to be the most important - from the point of view of the preparation of the functionality and the operability of the project.

Pilot testing in any application development project is an important part, it allows not only collecting feedback from potential users but also conducting additional analyzes during testing. Piloting often occurs at the final stages of development, and during testing, not only the functionality is checked, but also the rest of the applications - UI, UX.

From the very beginning of development, my client notified me that in mid-April they will have a pilot user - a large Finnish company, ready to participate in pilot testing of the project. My client supplies smart solutions for factories in Finland. In addition to installing and maintaining the sensors, my client (BeGo) in this collaboration with the pilot acts as a

service that collects and analyzes data. The Project instance, which was developed within the framework of this project, carries out the final phase of working with the data - displays it for the pilot user.

During preparation for pilot testing, we divided the testing process into two conditional stages: preparation and implementation. Preparation included the following:

1. Testing goals - we planned to find out how effective our system is for the user in terms of tracking information in real-time
2. Testing tools - only “BegoSolutions Project Instance” and a standard feedback form are used as tools
3. Participants' tasks are to make the most of the platform's functionality to control data
4. Participant profile - the company that agreed to participate in the pilot testing

In addition, during the planning of the pilot testing, we wanted to use clear criteria that would allow us to avoid misunderstandings or unwanted errors caused not by system errors, but by human ones, so we adhered to three criteria:

1. We have prepared detailed instructions with explanatory tips on how the services of our system function and how to use them correctly.
2. We clarified that some of the functions are in test mode and asked to use them a little more often in order to identify possible errors or find ways to improve.
3. The time frame for pilot testing was agreed in advance - 2 weeks.

We planned that the pilot testing would help improve the conclusions and find new ideas for improving and developing the project, in addition, the significant results of third-party people (without an IT background) from the client's point of view could bring new comments on our UX solutions. However, the problem with this testing was that the iterative approach is not applicable here - due to the fact that we have only one pilot client and we are quite constrained in time frames, so the approach in preparation had to be detailed and scrupulous.

The results of pilot testing should show whether the project instance is ready for full-fledged deployment and work with other clients, or whether large-scale studies, changes, and decision changes are required.

3 Development part

3.1 Analysis and comparison of existing products and services

There are several large tools on the market that serve as product and project management, which my client could probably turn to, but for some reason, he could not. There is no need to consider all the largest projects, so those products will be considered that seemed to be the first choice, but due to serious circumstances, they were not selected.

- BaseCamp
  - Advantages: BaseCamp is one of the most common tools for small to mid-sized businesses and covers most industries of this size business. BaseCamp, as a service, provides a fairly wide range for internal use - distribution, task scheduling, grouping, and categorization of users, projects, and tasks. This service provides several options for dashboards that can be customized and personalized for a particular project. Moreover, the service assumes the ability to create separate dashboards for clients, which can display individual personalized information.
  - Disadvantages: One of the main disadvantages of BaseCamp for the “BegoSolutions Project Instance” is the lack of flexibility: in BaseCamp it is quite difficult (or even impossible, in certain situations) to integrate data from third parties, the service itself offers a rather limited set of integrations - for example, Gantt Chart, but without advanced editing capabilities, there is no need to talk about data integration with PowerBI - it simply does not exist in the service.
  - Summary: BaseCamp was one of the obvious services that could be used by my client, however, the lack of flexibility and integration capabilities made it impossible to implement this project using this service.
- Monday
  - Advantages: Monday is a service that allows teams to quickly implement the simplest applications with a minimum set of functions: scheduling, planning projects, breaking them down into smaller tasks, distributing projects and tasks, grouping users, and more. Monday offers one of the best solutions on the market (at the moment) for the implementation of the transfer of data and statuses in real-time

so that all team members working on a particular project can receive current tasks and notifications in time if necessary.

  - Disadvantages: The main disadvantages of the service in the case of BeGo were the lack of the ability to integrate third-party data sources (SQL / PowerBI) or implement visual solutions through third-party libraries (Frappe Gantt). In addition, the service lacks the ability to implement a third-party window for inviting clients to it. Personalization of data is not provided by this service.
  - Summary: Monday is a good service for those who need to organize internal instances, however, in the case of BeGo, the lack of the ability to implement a third-party window for the client was one of the key drawbacks, because of which it was decided to abandon the services of this service.

- Asana
  - Advantages: Asana is a small (relative to other services on the list) project that offers its services for the creation of project management tools. This service offers minimal sufficient functionality for project management: Gantt Charts are replaced here with an analog of project timelines and milestones, which functions in a similar way, with only minor differences in the form of tooltips. In addition, the service offers convenient functionality for working with files and documents - they can be easily distributed within a team.
  - Disadvantages: Asana, like other services, does not offer users the open opportunity to integrate other data streams that my client needs.
  - Summary: Asana is probably the perfect choice for a small project and a small team like BeGo, which offers its services for little money ($ 13 per month), but the significant disadvantages - the lack of possible data integration, third-party libraries - turned out to be more significant for my client than the benefits of the service.
- Jira, Trello
  - Advantages: Both of these services are functional and global project management systems that have a wide range of functionality. The main feature of these services is a wide range of possible solutions for publishing tasks, solutions, and projects. Jira is used in many ways as a product for version control because of its flexible systems of social construction (for example, inside Jira, you can recreate teams and departments, assigning tasks not only personally to people, but also

departments). The ability to create collaborations is another advantage of these services.

- Disadvantages: Jira and Trello have limited functionality in terms of integrating third libraries or solutions, however, in Jira, you can use a code snippet to create custom activities/objects, however, this code snippet has a significant set of limitations, due to which, for example, you cannot use an iframe or other tags for data implementation.
- Summary: These services have similar functionality and could be a good solution for my client, but the functionality of the services is not wide enough to implement a project instance of my client's project.
-

- Deltek
  - Advantages: Deltek is a large service that offers its solutions for project and resource management, as well as CRM and a wide range of financial functions. These solutions and functionality really help build large algorithmic relationships that allow you to control and integrate many projects, tasks, and people into the system.
  - Disadvantages: If we consider Deltek as a service for the implementation of my client's project, then the first drawback is the “corporate” focus of the service - it offers its solutions only for large projects (30+ people), otherwise the project is offered the use of truncated functionality, which would not enough for the BeGo project. In addition, even the premium version of Deltek would not be able to implement all the required data integrations.
  - Summary: Deltek could be a good solution for my client, if not for the focus on large projects and the lack of important features.

To summarize: there are many projects on the market now that offer various functionality for project/process management, CRM systems. In many of the services from the list above, variations of collaborations are presented for working within a team and with the invitation of third parties, however, after a detailed examination together with my client (BeGo), we came to the conclusion that none of the services is able to meet his requirements. The most important drawback in these systems was the lack of the ability to integrate all the necessary data sources with which my client works.

3.2 Development milestones

After the list of project milestones was agreed upon with the client, a detailed plan which consisted of project development, version control, and testing phases was created. First of all, the possibilities were taken into account based on the parameters of current skills and resources (time and finances). The plan was drawn up in such a way that, first of all, the bases or functions that are critical for the system were prioritized, afterwards prioritization was built based on additional functions or possible implementations. It is important to emphasize that the original plan changed somewhat during development, but subtle changes were made to it.

In total, four pre-release and one release versions of the product were initially assumed, excluding intermediate versions, which served to fix technical flaws, critical bugs, or make minor changes (or small functionality). Each of the four versions implied significant changes and additions to the functionality:

- Test Version 0.1
    - The first test version of the project implied the creation of the main functions associated with the creation of User Management systems, which were created as the basis for the future distribution of users and projects. In addition, the first administrative (moderator) functions were incorporated in this test version. Thus, the largest User Management functions were created already in version 0.1, namely: the ability to register a client (by filling out a special form "RFI", subsequent authorization.
    - From the user's point of view, in this test version, it was necessary to create an opportunity to log in - for this purpose, the first version of the test “personal account” was created, which at this stage of development was exclusively test in nature, which did not provide any informative functions.

Figure 2. RFI form

- From the point of view of the administration, it was necessary to create a separate authorization system and, accordingly, a separate “personal account”, which was planned as an opportunity to see and interact with orders.
- Technologically speaking, the test version assumes a customized database that my client did not have. In this regard, it became necessary to build a service architecture and the subsequent implementation of the architecture in life. For these purposes, a standard database on the current Siteground hosting is used. The databases were built and populated further in the terms of development using SQL, and access to them is carried out through the standard PhpMyAdmin web interface.

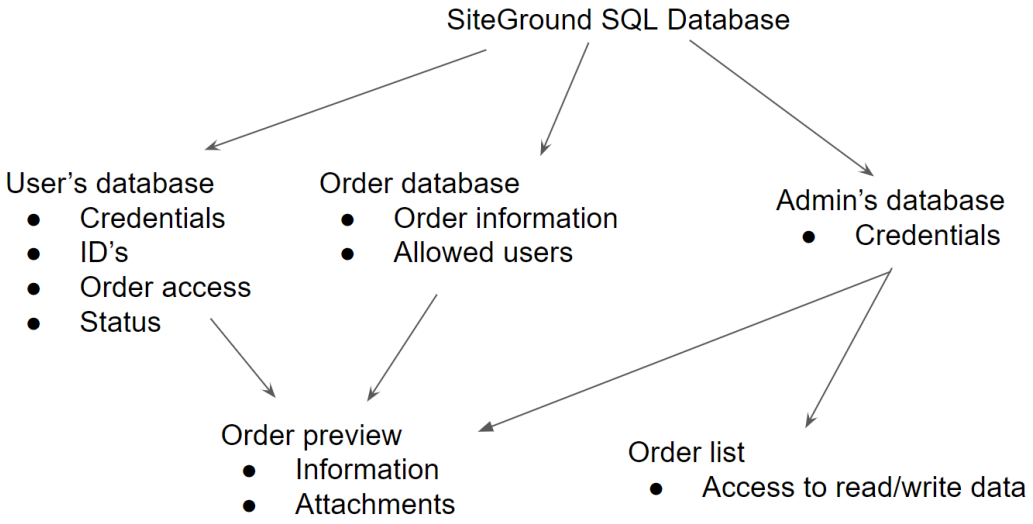

Figure 3. Database structure

- Test Version 0.2
  - Test version 0.2 involved a large amount of work with data that needed to be displayed in the user's and administrator's «personal account”. In this regard, at this stage of development, it was necessary to build functional pages that could interact with the database, in addition, the pages should be able not only to read and display data but also to send it.
  - This version was supposed to create a working version of the user's «personal account” with minimal functionality. At this point, it was not intended to use integrations to display data from third sources, but only to display information from our databases. Thus, the «cabinet» had to contain standard text values from the database, which were displayed to the user gradually (depending on the assigned status by the project administrator). In addition, in this version, from the user's point of view, additional and important functionality appeared - first of all, it was necessary to connect the function

to send email messages, as well as add the user the ability to invite third-party users on their own.

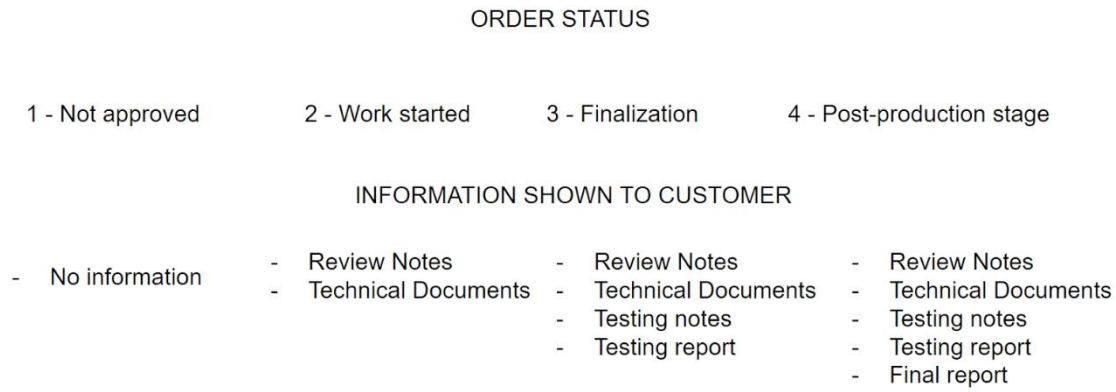

Figure 4. Order status and information accessible by client correlation

- The «cabinet» and administration functionality in this version has been changed - the ability to view and edit current orders has been added, in addition, an important function has appeared for adding files to order. All this is carried out directly through the administrative account page.

- Test Version 0.3
  - The third test version included cosmetic changes and fixes for current known issues, as well as additional minor changes to the web application. In this version, it was planned to continue working on data display.
  - To display the timeline and pipeline for working with the project, it was planned to use third-party libraries with open source code that help build graphs. For the "BegoSolutions Project Instance" development, the Frappe library was chosen, which creates Gantt Charts using the simplest scripts.
- Test version 0.4
  - In the fourth test version, it was planned to integrate a third-party service for displaying data in real-time, which informed the end-user about the status (and health) of the connected sensors, as well as a page with data analysis. This required the connection of a third-party Azure service, on which the data from the sensors was stored (and processed), as well as the connection of PowerBI (a child service of Azure), which allows you to integrate data with Azure and broadcast it using embedded connections to any platforms.
  - The data in this project was handled by the client - BeGo - they were engaged in the formation and analysis of data packets on Azure using built-in functions (such as Data Storage or Blob Storage) and visualizing using PowerBI. This

approach made it possible to use a simple technology for integrating the service into a web application.

- Test version 0.5
    - In the last, fifth, test version of the project, it was planned to make the last major changes, but it was not planned to add new functionality. So, the main goal of the fifth version was the idea to create a skeleton (structure) of a web application and create the possibility to add responsiveness in such a way that the web application was optimized and worked on various mobile devices. In addition, in this version, it was planned to introduce documentation and additional elements for the administration.
- Release version 1.0 [Pilot]
    - The release version did not imply the implementation of major changes or global functions. In this development project, the release version of the web application brought the product to the final form and prepared it for sending for pilot testing to the end-user. Based on this, it was planned that final changes will be made to the release version: such as transferring the project from the test platform to release one, changing some graphic elements and texts.

Thus, the original project development plan served as the basis for future testing phases and helped in the implementation of version control. The detailed terms of reference drawn up by me and my client helped to avoid major changes in the course of the project and to carry out the planned developments on time. Of course, it was not possible to completely avoid minor edits, but this did not prevent the development and the final product turned out to be almost completely consistent with the original plan. A more detailed description of the development progress and the introduction of intermediate versions of the project will be described below.

3.3 Data collection for the development

Before starting a full-fledged development, we needed to understand what kind of data will be used in the project and how we can get it for implementation in the project. Unfortunately, BeGo does not have a technical person who could describe in detail the whole picture of the data and prepare it for integration into the project, so we had to resort to a little research and make a certain list of data that can later be used in the project instance system.

First of all, we talked to the members of the BeGo team and figured out how their sensor system works, how they collect data, and how they output it. They provided us with a complete sensor data architecture and explained the data flow and its transformation

methods. For a general understanding, it should be noted that all processes occur inside Azure services.

The initial stage of data “Data Source Connections” serves to combine several data streams into one - in different circumstances and projects, data streams can vary in type, quality, and quantity of data. In the case of pilot testing, we assumed that the largest data source would be sensors, which generate millions of rows of data every hour. These are BeGo sensors and therefore we understood that their conclusion would not be difficult to access, as well as the subsequent analytics. Other sources of data in this project were APIs - they translated relevant data necessary for the correct operation of analytical algorithms. The API, as a source, is not controlled by us and therefore we could not ensure the stability of the data and guarantee its state (for example, so that the data came in the correct format, and not broken, or, for example, with distortion). Another source of data was data from our pilot user, it was a small and static amount of data that allowed my client to do some correlation with data from sensors for future calculations.

The data is then sent to Azure cloud storage, which helps connect the data to analytic algorithms. Sensors are connected to the Azure IoT Hub to track each sensor separately - monitor the status, health, output, and generation of data, as well as have access to the log of actions of each individual taken sensor. Moreover, this approach ensures the integrity of the system even if some (or even most) of the sensors fail - the stream will still be broadcast. In addition, connecting sensors to the Azure IoT Hub helps to ensure protection - the service has built-in threat recognition systems and the ability to connect secondary services to maintain operation in the event of a primary server outage. A similar approach to sensors is directly related to my development of a project instance system - I could be sure that in the end, the client will be able to see the dashboard he needs. Data obtained from secondary sources (API, client data) is not sent to the IoT Hub, but to the Data Factory, which serves as a place where the integration of third-party data takes place. The advantage of Data Factory is that it allows you to integrate, then transform, and then load the data into Azure for full use (ETL = Extract, Transform, Load approach). Since the project had third-party data sources (API, SQL, .CSV), my client had to resort to using such a service.

After the system has made sure that all the data is complete and not damaged, which means that you can work with them, it puts them in a joint storage - Blob Storage. Blob Storage - in the case of pilot testing, this is storage for unstructured data - all the data that is collected in the first stage goes here. In this project, Blob Storage serves for private storage of data, here the data has not yet been analyzed and categorized, which means that it cannot be shown to the client yet. In addition, Blob Storage serves not only as storage but also as an

optimizer. This storage works in such a way that it sends data to further Azure services (in our case, analytical centers) gradually, without causing loads on the servers and without losing data packets. Considering that we have a constant flow of data from sensors, this is an important element that allows us to maintain the stability of the project instance system. But Blob Storage is not suitable for data analysis in our case, because our data is very variable - only sensors are divided into several types, so in our case, it is necessary to categorize the data.

Data categorization and structuring happen in Azure Databricks. This technology allows you to do simple manipulations with a large pool of data - not only static but also dynamic (recall that our project uses both types of data). In our case, categorization is necessary to be able to personalize the data on the dashboard for the client and show separate data from the general arrays. Of course, Databricks allow us to do both analytics and categorization in one place, but my client decided that it would be too expensive - then Databricks were used only as a manual tool for categorizing incoming data, and categorized data is analyzed in the following stages. In theory, it is likely that with a large-scale deployment of Project Instance, my client will decide to implement machine learning in DataBricks.

Data analytics in our case happens on the Azure side, but using third-party tools. In this case, Azure is used exclusively as a data warehouse, further data analysis was carried out using manually created scripts, for this we used Python (in particular, the NumPy library was actively used), Excel toolkit (mainly computational capabilities). While the data volume for our pilot client is large, it is not large enough to resort to analytics automation using Azure tools. Data analytics in this case helps to categorize the data more quickly and make its conclusion simpler: for example, display those sensors that now have a dubious status (not working / unstable), or display data from a certain type of sensors. Moreover, due to the nuances of the manufactory in which it is installed, it is necessary to carry out a constant correlation with vibrations (for example, the data comes in a distorted format due to an internal failure, or due to external physical influences).

After all the manipulations with the data described above, the dates fall into Azure Data Lake. which acts as the final repository for structured data. Then my client builds a dashboard using PowerBI, a real-time data visualization toolkit. It is a large and powerful toolkit for plotting charts, displaying geolocation labels, tables, indexes, and other things into one single system that can be tabulated and personalized for an individual client. In our case, this process (with data passing through the Azure service chain and data output through PowerBI greatly simplified the task in developing Project Instance).

The general architecture looks like this:

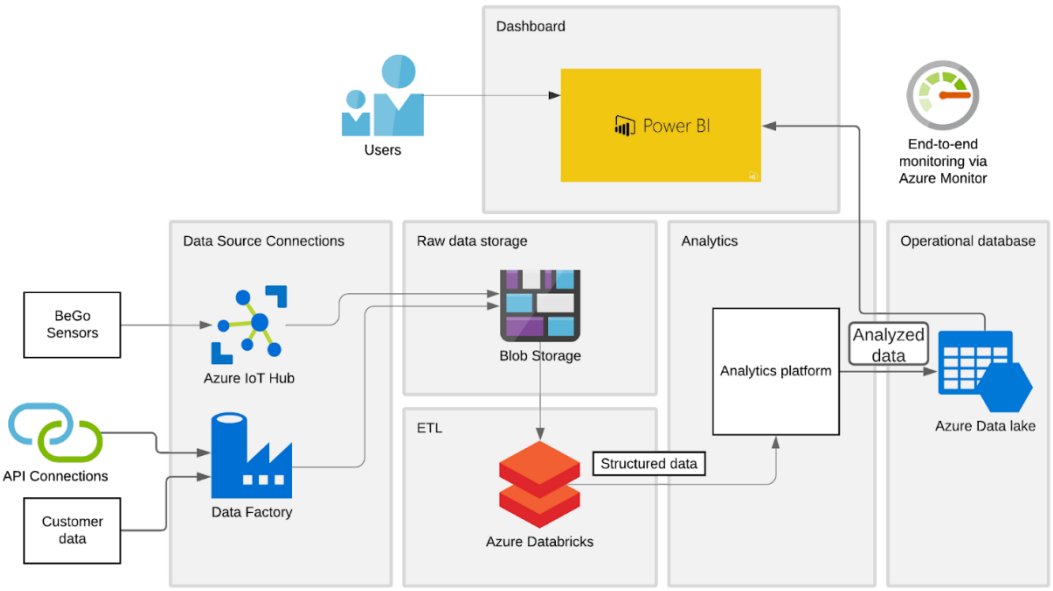

Figure 5. Data architecture (BeGo, 2020)

In addition, after talking with the BeGo team, we checked the logs of their reports and came to the conclusion that in addition to outputting technical data from Azure, we will have to output the following types of data:

- Text data: some information is conveyed to the client utilizing text values. For example, to display information about test notes. In our case, the text data also needs to be personalized, because pilot testing includes several fields that are important for clarification in the system that needs to be shown to the client.
- Visual data: for building specific charts (in our case, this is a timeline roadmap). Visual data is a small part of the data that we use in the project. First of all, this is a timeline roadmap, which we need to place in each project so that the user can track the current status of work on his order and see what further steps will be taken. In addition, Azure Dashboard, the principle of which was described above, is also in a sense visual data, since it is in the form of graphic elements that the end-user sees his data.
- Documents: An important part of this project is the ability to share documents between my client (BeGo) and his user. Documents can contain various information: from instructions for using the application (project instance) to the final report on sensors, data operation, and how to correctly interpret this data for subsequent assessment of their performance.

Thus, with the help of several face-to-face sessions with the client, I managed to build a clear picture and hierarchy of how to build the system during further development.

3.4 Choosing the web stack (coding languages, libraries, database type, etc)

To develop such a web application, it is quite easy to choose technical equipment in the form of the necessary programming languages, such as a database, external libraries, if you know and has a technical task. In the case of the project described in this document, by the beginning of development, the final goals and functions of the application were clear, so the following technologies were selected to implement the tasks:

3.4.1 Hosting

The choice of hosting in this project was based on several issues: security, optimization, and availability of third-party integrations (some hosting still prohibits installing third-party libraries like Yarn). Due to the fact that the application developed within the framework of this thesis will not assume a large and constant flow of customers and visitors, but it will still experience loads (when dashboards are loaded by the client or user). The chosen hosting provided several options to choose from: using shared hosting, virtual server (VPS), dedicated server.

For the implementation of this project, virtual hosting was chosen, since it fully met the expected loads and met the financial expectations of the client. Moreover, from a technical point of view, the chosen hosting also met the requests: it supported the implementation of PHP (and third-party PHP libraries), a SQL database. In addition, it implements MemCached technology, which will help in creating functions that are updated in real-time.

There were doubts about the database at the beginning of the project - it was necessary to choose the correct database and the type of data processing/transfer, several options were considered: a classic MySQL database, a NoSQL database, and alternative types like MongoDB or Postgre. When choosing a database, it was necessary to take into account several important factors: firstly, the amount of data that will be stored and used in it (change, access, request-response, and other operations are included), in the case of the project described in this document, the volume data is insignificant - we are talking about several gigabytes in total (taking into account all documents and integrations), secondly, this project will require various integrations - MySQL, for example, has a lot of integrations and most technical projects involve data integration into SQL databases. Due to the fact that the data in this project is structured and easily partitioned into tables, the choice of MySQL

seemed obvious, but in theory, some problems can be expected, such as mediocre performance. Even cluster data structuring in MySQL will not be able to provide high speed of working with data - it is limited by the technical nuances of MySQL databases and server technical components. Otherwise, the choice of MySQL looks in this case the most justified and correct.

3.4.2 Code

Creating a web application that will include both integration with Azure and the inclusion of different types of data from the MySQL library requires the use of several programming languages. First, there are several tasks - it is necessary to build not only the functionality but also its structure for future design. Secondly, it is necessary to ensure the optimization and correct operation of the application. For development purposes, the team chose a classic stack for development:

- The front-end, that is, the client-side of the site, is developed using the HTML / CSS stack, visualization, and control over events and triggers are done using JavaScript, the responsiveness of the site is done using the Bootstrap library.
- The back-end, that is, the server-side of the site, is implemented with PHP (mostly), which helps to connect the user and the server. With its help, such basic functions as data output, registration, login form, and others will be implemented. In addition, PHP helps to implement dynamic pages.

Thus, to create an application, it is planned to use a standard set of programming languages and additional libraries for them to implement individual functions.

3.4.3 Libraries

For the technical implementation of the site, it is necessary to use several libraries that will simplify the integration of some visual elements:

- Yarn is a library for managing other libraries. Basically, Yarn is a package manager that helps you work with other plugins and maintain documentation.
- Frappe is a Gant Chartt visualization library that allows you to build charts using JSON code and JS integration.

3.4.4 Integrations

To simplify the work and implement the necessary functions, several integrations will also be installed in the project:

- PowerBI is a necessary toolkit for connecting the created Dashboard and broadcasting it to the user
- PhPMyAdmin - for database management without third-party applications inside the hosting
- Dynamic Cache is an integrated library for managing and controlling the cache of a web page. With the help of it, you can avoid creating additional sessions and unnecessary queries to the database.

Also, the service will use internal systems to track events and collect anonymous data on application usage (such as session duration).

3.5 Version control

As already described in the chapters above, the project contains several development milestones, due to which the project has a clear version structure, however, for developing a large project it is easier and more correct to maintain not only a development backlog but also to use version control.

Version control avoids creating dozens of local copies of the source code and helps automate the process of creating a backup version of the code and, in case of an error, simplifies the process of rolling back the version to the working version. Moreover, version control greatly simplifies manual error catching (in the case of web development, automated error catching may not be entirely correct).

In this project, version control was done using private Git.

3.6 Step by step development

The actual development of the site began with the creation of simple skeletal functions using HTML and CSS markup. An important step was to understand and come up with the future structure of the web application and come up with a clear UX so that the pilot user does not get lost on the site because the first test builds had little functionality and a lot of tips. During the development, it was necessary to change the existing functions several times so that they would work correctly after the innovations or changes, there was a one-time need to change the SQL tables. More about the development stages:

3.6.1 Version 0.1

The first version of the application included the fundamental foundations of the application, which is aimed at working with the end-user - these were the forms of registration, authorization, and the user's “personal account”. To do this, it was necessary to set up a database, create tables and establish communication between the server and the client-side of the site. To begin with, the primary skeletons of future registration and authorization forms were created, which did not have a design, but already had full

functionality for an unprotected entrance (at this stage, the security issue was not considered, because the passwords were sent to the base in unencrypted form and it was assumed that at this stage there will be only testing of the main functionality was carried out. The most important thing at this stage was to build the future architecture of the tables in the database in such a way that at the final stages of development it was not necessary to recreate the tables anew, thereby not changing hundreds of lines of code.

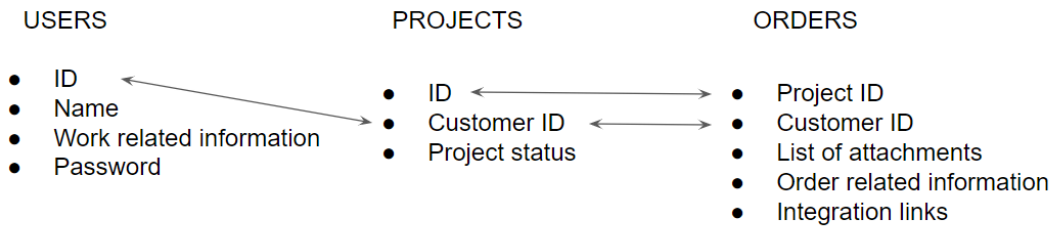

Figure 6. Database architecture

In the context of this application, it is necessary to understand that the dependence and correlation of data (especially in the case of pilot testing) are large - the user creating the query must have access to the data, therefore the table with the client's data is directly related to the table that stores the primary data about the project. The table with the primary data must have a link with the table in which more detailed data is stored. This is done for two things: security and optimization. In both cases, the system works in such a way that it shows and requests only that part of the data that is needed at a particular moment. For example, if a user (client) request is sent to display the current status of an order, then the table with detailed data will not be loaded, which will shorten the execution

time of the request.

Figure 7. Design template

In addition, at the first stage, it was necessary to enable the user to invite third-party people to his project, bypassing full-fledged registration, for this, it was necessary to make appropriate exceptions in the data tables and in the authorization systems in order to avoid possible conflicts. Separately, it should be mentioned that it was necessary to create a single authorization system for users (both original and affiliated), and a separate one for the administration. Thus, the results of the development of the first version were as follows:

- Complete creation of authorization and registration systems for the end-user
- Creation of a “personal account” for the user (and a structure for future functions in it)
- Creation of a similar office for the administration
- Setting up a primary database and creating a data table architecture for storing user information and order data.

3.6.2 Version 0.2

The development of the second test version was carried out in order to fill the “personal account”s of the user and administrator with information about orders, as well as the purpose of the development was to add functionality with the first integrations.

First of all, it was necessary to improve the registration system for users and the possibilities of inviting users, in general, the system worked correctly, but during testing several shortcomings were found - for example, the system did not register secure passwords (with the different case of letters and special characters), as a result of which there were errors from the database - passwords came either broken (in the form of a garbled line) or in the “Null” format. The bug was fixed by a small database reconfiguration. Another error that occurred in the first step was causing the user to lose their session after reloading the page - in this case, the error was deeper than expected.

The systems of “personal account”s for the user and administration are built in the project on the principle of an HTTP session. Roughly speaking, every time the user clicks a button that displays some data on the screen, the client-side sends a request, which the server processes and then sends a response. In the case of a project that is being developed within the framework of this thesis, in the user's “personal account” there was a lot of functionality that required a page reload or page change, this was necessary in order to optimize the site and not load any of the parties (neither the device nor the server). If it was decided to abandon the sessions, then each time new information was loaded on the request, a password would have to be entered, but this, although safe, is too long and incorrect from the UX point of view, therefore HTTP sessions were recreated that store personal data and if the page is reloaded or another page is loaded, the HTTP session sends data on its own and then the user can connect to the required data without any problems. In the case of the first test version, an error occurred, as a result of which sessions could not be created due to the circulation of redirects (for some reason, the page loaded independently without interruption, thereby clogging the available cache). As a result of fixing this bug, it was decided to make a system of internal page changes using the dynamic capabilities of PHP instead of regular redirects. The pages were redesigned in such a way that their elements were replaced not by loading another page, but by unloading a PHP script that changed the content of the page after the user clicks a button.

In this version, a system for outputting text data that was stored in a database was implemented. In addition, in the same version, a system for entering data from an administrator's position was implemented. Data input and data output occur in the project according to one similar principle: in the case of data input, the administrator enters the necessary values into the text fields inside the admin panel, then the client-side sends this

data to the database, which then distributes them inside the table based on the entered values ... The check occurs at the input stage - the table is checked whether there is an order with a similar ID in the database (the ID is sent together with the data), if there is, the table is updated. To prevent situations when the table cannot be updated due to a non-existent order, a system was created that allows the administrator to see only those orders that are actually in the system. Information for the user is displayed in a similar way - when entering the system, the user ID is sent to the database, according to which the system searches for a match in the list of orders, if the user has an assigned order, it will display all the necessary information on the screen, if not, it will report this, but will still allow you to get into the system.

Another important innovation in this version is the implementation of the mail-sending functionality. When registering, the user did not previously receive information about his own order by mail, nor did he receive data on how to get to the site and “personal account”. In the second test version, this was fixed - PHP Mailer installed on the site processes the simplest requests and sends messages to users with correct data.

3.6.3 Versions 0.3, 0.4

In versions 0.3 and 0.4, it was planned to introduce external integrations for building timeline graphs and displaying data from sensors for the user.

To implement a graphical display of timelines, it was decided to use the open-source library Frappe, which uses a JS + JSON bundle to visually display data in the form of timelines. The library functions thanks to the pre-installed plugin manager Yarn. Using such a library simplifies the development process and saves time - otherwise, you would have to develop graphic elements yourself using jQuery. The Frappe library is integrated into the site by installing some files (graphic CSS, JS scripts, and a JSON file that contains the data to display). One of the additional reasons for finding such a solution was that in the future the site can be used by people who do not have much programming experience, so the search for the optimal solution required the inclusion of an easy way to edit such graphics.

As described earlier, sensor data is stored and analyzed on the side of my client, BeGo, using Azure platforms. The customer provides PowerBI Dashboard, which combines graphical capabilities to visualize data from Azure servers. The Dashboard integration posed several questions: how to integrate it correctly so that the page skeleton does not "break", how to display it correctly on different devices and how to load it so that the page load time does not increase. The issue of the correct display was solved with the help of the Bootstrap

library, which allows you to create responsive containers and adjust any elements in the containers to the screen size. Thus, having created a site from several containers, it was possible to easily operate with free space. In addition, not all of the information on the site was displayed - the site implemented a system of "tabs", which disables the existence of elements until they are enabled by pressing a button. In this way, the site load time is optimized - each element has an attribute “none” applied to the “display” tag, therefore, formally, it is not loaded when the site is first turned on - which significantly reduces its load time.

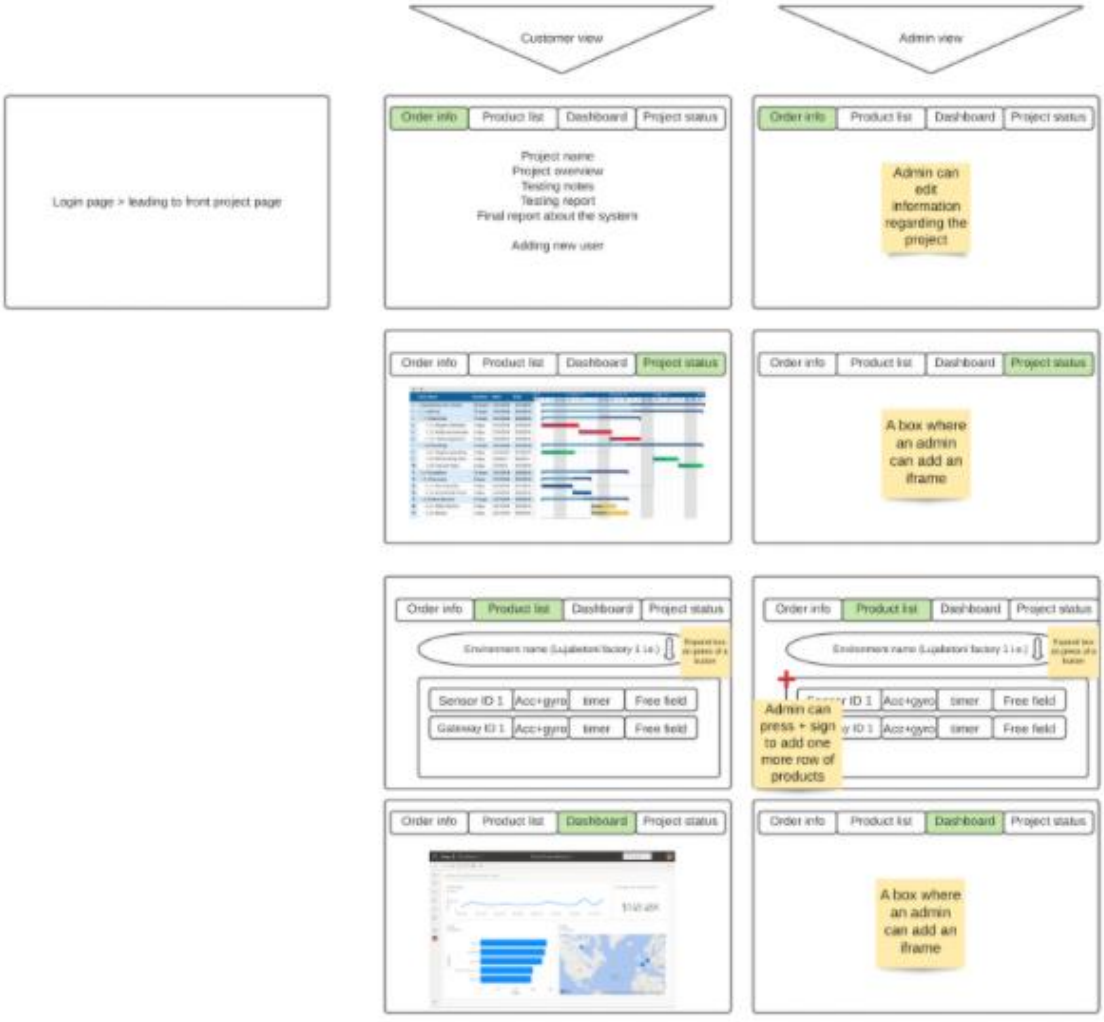

Figure 8. User and administrator panel concept

3.6.4 Version 0.5

At the stage of the fifth version of testing, it was planned to correct the technical shortcomings of the project - small ones were corrected during the development phase, and large ones were postponed to further stages. The fifth version became "pre-release" for pilot

testing - therefore, work began to fix bugs found during testing: first of all, bugs related to the database were fixed, there were several of these: incorrect display of data in the user system and incorrect loading of documentation from the panel administrator. Working with files, as mentioned above, is limited to hosting by weight and extension, therefore it was necessary to create artificial systems to bypass restrictions - the file was uploaded not to the database, but to the hosting's local storage, from where a temporary link with the file name was created, and then the link was added to a table with data, from which, in turn, they were then displayed for the user. This created another problem - for some reason, as a result of the conflict of different encodings, the user could sometimes see incorrect output - for example, hieroglyphs in the text. To solve this problem, when transferring data, an additional check for the encoding was created and, if it did not match, it was corrected for the correct one (the one that was present on the site).

In addition, at this stage, the structure and skeleton of the application were brought to the final form, navigation and menus were corrected, and the correct display of all functionality.

3.6.5 Version 1.0 [Pilot]

Version 1.0 is the version that was eventually sent for pilot testing to the end-user. When version 1.0 was created, it was not originally intended to introduce major changes or new features. The task of version 1.0 was, basically, the introduction of "cosmetic" changes - replacement of placeholders of pictures and texts, transfer of the application itself from the test site to the release site.

3.7 Challenges in the development

During development, there were situations that took longer than planned to fix or resolve. Optimizing the application loading speed became one of the key challenges during development. Initially, the user's information was displayed in one large list, which is why its loading took a long time. To avoid this and solve the problem with loading, it was decided to divide the information into tabs, each of which activates its own element containing information. The actual implementation of tabs is a simple script that hides elements from the site and prevents the user from accessing them. This was implemented by disabling the display attribute, but the problem was that some of our integrations use third-party graphic libraries (for example, Frappe has its own .CSS code, and the dashboard has graphic elements sent from Azure), and when the display is disabled, the element cannot receive them. In fact,

an element can get its graphic year only when the page is loaded; it is impossible to manually force it to load anything, so it was necessary to come up with a solution that would allow loading elements with a graphic component and then turning them off. The final solution to the problem was as follows: elements that require external graphics libraries were loaded along with the body of the application but had visibility of 0% (which is why they were not loaded for the user, but loaded for the application). Thanks to this manipulation, it became possible to use the tab system in the correct way, but at the same time, it was possible not to increase the application load time.

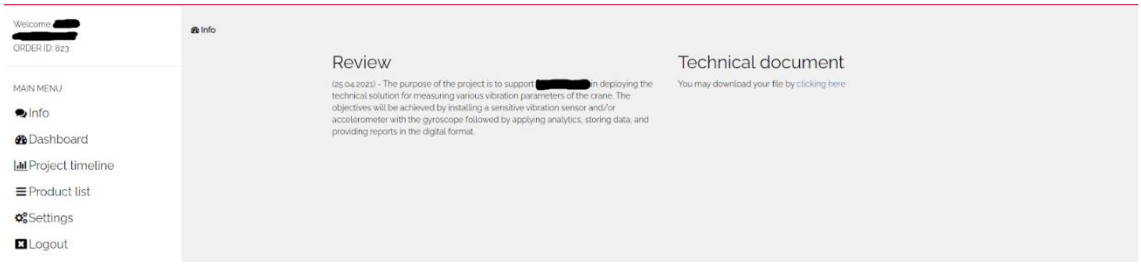

Figure 9. User panel

Another key issue during development was page caching. The hosting chosen for this project introduced serious restrictions on the size of the saved cache on the site, so some updates were applied from several hours to a day. Although this slowed down the development of the application, it was possible to work with it, but then it became a big problem when it turned out that such hashing on the hosting side interferes with the correct display of data from the dashboard, which updates data in real-time. It was built in such a way as to update the data approximately every 5-7 minutes, but due to hashing, the dashboard version in the application was updated every few hours, which caused the main function of the application to disappear. In this case, it was necessary to eliminate the limited use of the hash - but this could not be done either with the help of money or by disabling other functions. Then a different decision was made - to remove the page from under the hashing cycle. In fact, the hash was saved for the sake of the session - but after the information on the site began to be displayed after the introduction of the data tabs, the need to recheck the session to display the data disappeared, so it became possible to disable it and free up space for regular dashboard updates.

In the course of development, there were other, less significant bugs, but they were often fixed immediately after the release of the functions. This was made possible by using an iterative approach to the final product and by the fact that BeGo responded quickly to new builds and participated in testing. Of course, all the technical flaws have not yet been fixed (and, probably, some of them have not been found yet), but at the moment the pilot version of the product allows you to use all the functions as intended.

3.8 Piloting the project

After the development of version 1.0 was completed and all data was transferred to the production platform, the team began to form the necessary platform for conducting pilot testing.

First of all, the documentation for the pilot user was created, which included a description of all available functions, possible nuances of use, and a general description of the system. As planned, the pilot period lasted for two weeks, during which time the pilot was able to try out the system. It is important to clarify that already at the time of testing, BeGo and the pilot user had a signed contract, so the initial steps of using the system (like registration) were skipped because at the time of the start the sensors were already installed. The pilot testing itself consisted of monitoring the functionality of the functionality, how the server withstands the load and, of course, it was necessary to establish how well the UX elements work and collect feedback.

In a personal conversation (Dunkin, Sirtse. 2021. Personal communications.) with representatives of the pilot client, it was possible to find out that, in general, the application interface is intuitive, but there are some details that can be changed to improve the overall visual style. Technically, the pilot team was unable to find critical or recurring bugs, but minor technical flaws were found as a result of misinterpretation by the browser (Firefox, for example, did not display the login button correctly). The documentation generated for the pilot provided the test takers with sufficient contextual information to use the “out of the box” service, and separate prompts were placed in the application.

The results of the pilot were satisfactory to all parties: as a result of the feedback, BeGo was able to obtain data that will help determine the primary goals for subsequent developments, both in terms of functionality and in terms of visual aspects. Pilot users were satisfied with the possible functionality and appreciated the convenience of such an application.

3.9 Product release

As stated earlier, further development of the project and its possible prospects will be carried out by BeGo or by third-party developers, but already now, at the stage of pilot testing, it is clear that the application has room for development and improvement. First of all, you can add flexibility to this project: for example, allow one user to have several connected projects and the ability to switch between projects. This flexibility will help customers create unique accounts with access. In addition, it is necessary if the customer has two simultaneous orders. Another opportunity to improve the service will be the integration of service APIs available on Azure that broadcast individual states. Thus, using the API, for example, you can broadcast the values of individual sensors or, for example, show the user individual states upon request. Likewise, you can create even more personalized pages for

the client and display more detailed data, while maintaining the speed of the application and not overloading the server. In general, the functionality of the application can be improved: in the list of products, for example, you can add categorization for more convenient grouping and display of products, the work of the admin panel can also be facilitated by creating a more flexible system with dropdown menus. Another issue for the client to work on is the security of the service. Primary security was implemented during the past testing phase, but I think, given the data, BeGo will want to increase security, there are several options: for example, to enter authorization by token (for example, Google Auth and 30-second passwords), both for users and administrators - two-factor verification is a reliable way to protect the user. In addition, the client will need to work on the data encoding and the possibility of external influence on the database. As a result, there is certainly room for improvement, but the project is already ready at this stage to launch a minimum valuable product.

4 Conclusion

4.1 Deliverables

After completing the last phase of development, the client (BeGo) received:

- Source code of the entire project
- Technical documentation on the construction of the project and the possibilities for its further improvement (the document contains all the technical nuances of the code, as well as data on the libraries used and how they are integrated into the code)
    1. The document that covers used libraries and integrations
    2. The document that covers crucial parts of the code
    3. The document that covers user functionality (mainly for users)

4.2 Conclusions

The aim of this thesis was to build a pilot version of a fully functional web application capable of performing the functions of a project management system. After conducting pilot testing, we can say with confidence that in the course of development it was possible to create a system that uses a wide range of web technologies and implement all the conceived functions - at the moment the system works with data: you can interact with them (add, delete, edit), they are correctly displayed individually for each user. The application

correctly integrated and functions a dashboard created using Azure and PowerBI, in addition, the graphical part in the form of a Frappe Gantt Chart is correctly implemented. Using open-source libraries in the creation of this project was an important part because it allowed a shorter development cycle and faster start testing functions. Despite the fact that the project was developed alone, all the functionality turned out as expected at the beginning - it was pilot testing and the implementation of all functions was supposed to be minimal, which was achieved by the end of development. The project has development opportunities and, probably, the client will want to continue working on it.

In addition to the initially set tasks, some others were also implemented - for example, connecting the PHP Emailer service, which allows you to send letters, or, for example, creating a minimal design and UI (although initially it was planned to create only one skeleton). That is, in a sense, the task was over fulfilled.

Scheduling the development of a project and "breaking it down" into parts based on the implementation of the main functions proved to be a useful idea, which helped to quickly implement the functions and test them, which in turn contributed to the rapid detection and fixing of bugs. This phasing became possible thanks to a clear specification and close collaboration with the client. Of course, there were some changes made during development that influenced the course and the final development schedule. For example, the addition of third-party functions like sending mail, but the decisions made in the end contributed to the simplification or faster implementation of other tasks since these functions covered the originally planned ones. During development, of course, there were also problem situations when it was necessary to spend more time and resources on the implementation of the functionality, but ultimately the product was put into use on time.

An interesting topic of this development was the possibility of implementing development methodologies in a team of one developer. Despite the fact that most often methodologies are developed on a team basis, it was an interesting challenge to try to implement certain processes alone. The development cycle system was adopted from scrum and agile methodologies - when each version was separately discussed with the client, then it was launched into development and several iterations were released along the way, after which final decisions and improvements were made. The distribution of tasks, unfortunately, cannot be applied in a team of one developer, but they can be prioritized and categorized according to some statuses - for example, according to the speed of their implementation or complexity.

The client has already expressed his desire to continue the development of this project. This application has great development potential since the basic functions laid down during development can serve to deploy more global tasks. For example, the implementation

of the multi-project display function is possible based on current developments. Moreover, the implementation of further tasks will be simplified thanks to the created technical documentation and detailed comments on the final code.

References

Electronic

4. Abrahamsson, P., Salo, O., Rohkainen, J., Warsta, J. 2002. Agile software development methods: review and analysis. Accessed 02 May 2021. https://arxiv.org/ftp/arxiv/papers/1709/1709.08439.pdf
5. Beynon-Davies, P., Carne, C., Mackay, H., Tudhope, D. Rapid Application Development (RAD): An empirical review. 1999. Accessed 03 May 2021. https://www.researchgate.net/publication/31978101_Rapid_application_development_RAD_An_empirical_review
6. Jones, T.S., Richey, R.C. Rapid Prototyping Methodology in Action: A Development Study. 2000. Accessed 03 May 2021. http://www.uky.edu/~gmswan3/609/Jones_Richey_2000.pdf
7. Lotz, M. Waterfall VS Agile: Which is the Right Development Methodology for your project?. 2018. Accessed 03 May 2021. https://www.seguetech.com/waterfall-vs-agile-methodology/
8. McCormick, M. Waterfall vs Agile Methodology. 2012. Accessed 03 May 2021. http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf
9. Sacolick, I. 2020. What is agile methodology? Modern software development explained. Accessed 02 May 2021. https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html
10. Royce, W. Managing the development of large software systems. 1970. Accessed 03 May 2021. https://web.archive.org/web/20160318002949/http:/www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf
11. Singh, A. 2019. What Is Rapid Application Development (RAD)?. Posted 6 December. Accessed 03 May 2021. https://blog.capterra.com/what-is-rapid-application-development/#:~:text=Rapid%20Application%20Development%20(RAD)%20is,strict%20planning%20and%20requirements%20recording
12. An Introduction to Rapid Application Development (RAD). 2009. Accessed 03 May 2021. https://www.ogcio.gov.hk/en/our_work/infrastructure/methodology/system_development/past_documents/rad/doc/g47a_pub.pdf
13.
14. Manifesto for Agile Software Development. 2001. Accessed 02 May 2021. http://agilemanifesto.org/

Personal Communications

1. Dunkin, D. 2021. Head of Marketing. BeGo. Email to the author. 12 May 2021. Personal Communication
2. Sirtse, S. 2021. Documentation engineer. BeGo. Email to the author. 12 May 2021. Personal Communication.

Figures