МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

__________________Жуков І.А.

« ____ » ____________ 2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

випускника освітнього ступеня "МАГІСТР"
спеціальністі 123 «Комп'ютерна інженерія»
освітньо-професійної програми «Комп'ютерні системи та мережі»

Тема: Система моніторингу робочого часу працівників IT-компанії

Виконавець:_____________________________________ Руснак А.А.

Керівник:_______________________________________ Надточій В.І.

Нормоконтролер:_________________________________ Надточій В.І.

Київ 2020

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
Faculty of Cybersecurity, Computer and Software Engineering
Computer Systems and Networks Department

PERMISSION TO DEFEND GRANTED

The Head of the Department

_____________________ Zhukov I.A.

"_____"______________________2020

MASTER'S DEGREE THESIS

(EXPLANATORY NOTE)

Specialty: 123 Computer Engineering
Educational-Professional Program: Computer Systems and Networks

Topic: _____*IT company employees' working time monitoring system*_

Completed by: ______________________________________Rusnak A.A.

Supervisor: ____Nadtochiy V.I.

Standards Inspector___________________________________Nadtochiy V.I.

Kyiv 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра комп’ютерних систем та мереж

Освітній ступінь: «Магістр»

Спеціальність: 123 «Комп’ютерна інженерія»

Освітньо-професійна програма: «Комп’ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
____________Жуков І.А.
« ____ » ___________ 2020 р.

З А В Д А Н Н Я
на виконання дипломної роботи

Руснака Андрія Андрійовича
(прізвище, ім'я, по батькові)

1.Тема проекту (роботи): “Система моніторингу робочого часу працівників ІТ-компанії”
затверджена наказом ректора від " 25 " вересня 2020 року № 1793/ст.
2.Термін виконання проекту (роботи): з 01.*10*.2020 до 25.12.2020
3. Вхідні дані до роботи (проекту*): вимоги до системи*
4. Зміст пояснювальної записки:
аналіз предметної області.
вимоги до СМРЧ
структура СМРЧ
прототип СМРЧ

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:
ілюстративний матеріал в Power Point.

6. Календарний план-графік

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Узгодити технічне завдання з керівником дипломної роботи	1.10.20-8.10.20	
2	Виконати пошук та вивчення науково-технічної літератури за темою роботи	9.10.20-15.10.20	
3	Опрацювати теоретичний матеріал	16.10.20-18.10.20	
4	Проаналізувати відомі підходи та методи розв'язку проблеми	19.10.20-03.11.20	
5	Виконати аналіз результатів, розробити рекомендації та оформити пояснювальну записку.	04.11.20-15.12.20	
6	Оформити графічну частину записки та подати матеріали работи на антиплагіатну перевірку матеріалів	06.12.20-12.12.20	
7	Отримати рецензію та відгук керівника. Надати матеріали роботи на кафедру.	13.12.20-14.12.20	

7. Дата видачі завдання: “1” жовтня 2020 р.

Керівник дипломної роботи _________________________ Надточій В.І.

(підпис керівника)

Завдання прийняв до виконання _____________________ Руснак А.А.

(підпис випускника)

NATIONAL AVIATION UNIVERSITY

Faculty of Cybersecurity, Computer and Software Engineering

Department: Computer Systems and Networks

Educational Degree: “Master”

Specialty: 123 “Computer Engineering”

Educational-Professional Program: “Computer Systems and Networks”

APPROVED BY

The Head of the Department

____________________ Zhukov I.A.

“_____” ____________________2019

Graduate Student’s Degree Thesis Assignment

Rusnak Andriy Andriyovich

1. The Project topic: “*IT company employees' working time monitoring system*”

Approved by the Rector’s order of 25.09. 2020 № 1793/st.

2. The Thesis to be completed between 01.10.2020 and 25.12.2020

3. Initial data for the project (thesis):.*requriments to the system*

4. The content of the explanatory note (the list of problems to be considered):

subject area analysis

requriments for WHMS

structure of WHMS

prototype of WHMS

5. The list of mandatory graphic materials:

illustrative material in Power Point.

6. Timetable

#	**Completion Stages of Degree Project**	**Stage Completion Dates**	**Remarks**
1	Agree on the terms of reference with the thesis supervisor	1.10.20-8.10.20	
2	Perform search and study of scientific and technical literature on the topic of work	9.10.20-15.10.20	
3	Elaborate theoretical material	16.10.20-18.10.20	
4	Analyze known approaches and methods of problem solving	19.10.20-03.11.20	
5	Analyze the results, develop recommendations and issue an explanatory note.	04.11.20-15.12.20	
6	Make a graphic part of the note and submit the materials of the work for anti-plagiarism	06.12.20-12.12.20	
7	Get a review and feedback from the manager.	13.12.20-14.12.20	

7. Assignment issue date: 13.05.2019

Diploma Thesis Supervisor ______________________________ Nadtochiy V.I.
(Signature)

Assignment accepted for completion ______________________Rusnak A.A.
(Signature)

ABSTRACT

The Explanatory Note to the Bachelor's Degree Thesis "IT company employees' working time monitoring system.": 46 pages, 14 figures, 3 tables.

The Goal of the Bachelor's Degree Thesis – create IT company employees' working time monitoring system

The purpose of the thesis – creation of a system for monitoring work processes in the technical department of the IT company, which will help record the time of arrival at work, check the timely return of the employee after lunch break, presence at work throughout the day, determine the end of work, identify absences, divide paid working hours.

Development method - OOP - object-oriented approach.

Hardware and software - PC with Windows 8 or 10; IntellIJ Idea and Visual Code object-oriented programming environments.

Results is recommended to use in the technical departments of IT company.

Forecast assumptions about the development of development objects - the developed software can be expanded in the future by connecting new activities outside working hours, for example, while working on vacation.

CONTENT

LIST OF SYMBOLS, ABBREVEATIONS, TERMS

WHMS	working hour monitoring system
FR	functional requirements
NFR	non-functional requirements
IT	information technology
PC	personal computer
DB	database
OS	operating system

INTRODUCTION

To optimize the work process, increase work efficiency and meet deadlines, you can not do without clear planning and control of working time. A clear organization of working hours is equally necessary for both the manager and subordinates. Supervision by the employer or manager maintains work discipline in the team and guarantees fair pay.

Control as part of the monitoring of working hours allows to solve such tasks as fixing the time of arrival at work, checking the timely return of the employee after lunch break and presence at work throughout the day, determining the end time, identifying absences, distribution of paid working hours. , holidays, sick leave, downtime due to the employer, etc.

Manually controlling the work of the team is an impossible task when the staff of the enterprise has more than one hundred people who also work in branches or remotely, to cope with such a task by force only automation.

In this thesis project, I will develop a prototype of the monitoring system of the technical department of the Internet provider, which will help to avoid the described problems.

PART 1
SUBJECT AREA ANALYSIS

1.1. The concept of optimization of working time.

The term "working time" means the period during which the employee performs work duties. The length of the period depends on the terms of the employment contract, working hours and internal rules of the company. The law does not prohibit the use of employees to perform professional duties at other times.

To optimize the work process, increase work efficiency and meet deadlines, you can not do without clear planning and control of working time. A clear organization of working hours is equally necessary for both the manager and subordinates. Supervision by the employer or manager maintains work discipline in the team and guarantees fair pay.

Managers of large organizations with a large staff do not have time to personally monitor each subordinate. In addition to the control function, the manager, head of department or project manager performs other duties. During work, there are unforeseen difficulties or circumstances not provided for in the plan, and the work process can not be fully predicted and planned, even the leaders of time management.

Simultaneously with the increase in staff and the number of projects, the tasks multiply, so that personal control of working hours becomes impossible. One solution is proactive management, which helps reduce the number of difficult situations in the employee's work.

1.2. The concept and legal justification of working time accounting.

Monitoring of working hours is the monitoring of staff activities during the working day and control over compliance with labor regulations. Monitoring belongs to a group of preventive measures that help maintain discipline and prevent abuse of trust by the employer. This method is primarily aimed at identifying violators who solve personal rather than official issues.

Control and accounting of working time is conducted within the framework of the Labor Code. By law, employees have the right to breaks, weekends and holidays.

Why keep track of working hours?

Control as part of the monitoring of working time allows you to solve several problems:

• Record time of arrival at work to calculate systematically late and absent.

• Check the timely return of the employee after the lunch break and attendance at work throughout the day.

• Determine the end time.

• Detect truants.

• Divide paid periods of working time into actual hours worked, vacation, sick leave, simple through no fault of the employer, etc.

In practice, managers often face such a problem as non-compliance with deadlines. It is easy to calculate the terms: the norm of working hours under Article 91 of the Labor Code is 40 hours per week. The norm is compared to the qualification directory or functional responsibilities of the employee. The result is a rational scheme of distribution of staff time.

A significant disadvantage is that the scheme does not take into account a number of important factors.

Table 1.1.

It is not covered in standard systems during the working hour.

Factor	Examples
The human factor	Constant smoking breaks, hospital stays, delays, personal affairs at a paid time, etc.
Force majeure	Emergencies, natural disasters, accidents, delays by contractors, etc.

All this leads to the difference between the time spent and the work actually performed.

The introduction of methods of control of time indicators of work is required first of all by the employer. This will allow you to pay for the time actually worked, rather than for tardiness, smoking breaks and computer games in the workplace. Analysis of the collected data on the one hand, written reports and explanations of employees - on the other hand, allow the manager to make informed decisions about the rational use of the working period and accordingly accrue wages.

1.3. Monitoring employees' working time

The tale of wasted time - how not to make it come true? The classics of Soviet literature and cinema teaches children to value their own and other people's time. Such genres no longer affect the older ones. In adulthood, serious arguments are needed to spur on fruitful work.

Working time monitoring becomes a salvation for employers from idlers. For employees, this is a way out of a protracted immersion in the Internet.

Monitoring working time means observing how staff use it, finding problems, and optimizing processes.

How to control the working time of employees?

People have learned to control time. But more - someone else's. Especially colleagues and subordinates. Time sheets are kept in circulation. Everywhere, from factory gates to cultural and educational institutions, equipment has been installed that celebrates punctuality. Time tracking and control systems for employees' actions are being introduced into PCs. There are plenty of options for monitoring employment. It is equally important for the employer not to overdo it in this matter and not let the team go by itself.

Monitoring the use of working time

It is difficult to find a person who does not engage in personal affairs during working hours - not a minute for himself, all for the sake of the cause. This includes tea drinking, smoke breaks, games, reading social networks, talking on the phone. Everything is permissible. You have to relax, rest. But within reason. According to statistics, from 30 minutes to three hours a day are ineffectively used.

The spread is significant. Economic damage - accordingly. The issue of productivity remains relevant for companies of different levels, large and small, over ten thousand and up to 10 people. Regardless of their specialization, they are united by the desire to be successful, to make money. In employees, see interested colleagues involved in the processes.

1.4. Methods of working time control.

The employer is interested not only in the timely appearance of employees in the workplace, but also in the productivity of the team. Productivity directly depends on what the employee is doing in the workplace.

How to control the work of employees on the Internet and on computers through one interface? Learn.

Does the employee have time to perform duties if he spends time on social networks, correspondence with friends, watching the news, reading books or surfing the Internet. Methods of controlling the working hours of employees include fixing:

• tasks performed between turning the computer on and off;

• telephone conversations with clients, partners and contractors;

• screenshots of the desktop during the desktop for using specialized IT solutions;

• content of corporate correspondence, including e-mail correspondence;

• presence on site during working hours using video surveillance.

Personnel should be notified in advance before applying control methods.

To do this, the additional agreement specifies the methods of monitoring the use of working time, to change the conditions is allowed only with the consent of staff. Moreover, it is illegal to install surveillance cameras in offices without notifying employees.

Data on monitoring the use of working time form the basis for the analysis of staff performance. Performance metrics include:

• final - evaluation of the results of the completed project as a whole;

• intermediate - control of work performed for the specified period;

• periodic - summarizing after a certain period of time;

• election - irregular selective monitoring of individual employees at the discretion of management.

Any monitoring methods are aimed primarily at assessing how efficiently and rationally the employee allocates working time, and at the same time assess the level of professionalism.

Ways of monitoring, accounting and estimating staff working hours have evolved over time, and yet organizations, especially government agencies, follow traditional approaches. For example, appoint a person in charge, a watchman or a duty officer, to keep a log of working hours. The person in charge records the time of arrival and departure and systematically prepares reports for the head.

Another option is to introduce the position of administrator in the staff, which will be located in the same room with colleagues or in a separate office and monitor the continuity of the work process.

Another way is for employees to keep personal records, monitor and record their time. The method helps to evaluate the work done from the point of view of performers and develops independence.

A more common way is to install an access control and management system using passes or fingerprint scanners. Information on each employee is stored in a file and is available for viewing at any time.

A reliable but expensive method of monitoring and accounting of working time of the company's staff is a video surveillance system. It will be necessary to spend money not only on the purchase and installation of video cameras, but also on the salary of an individual employee, whose responsibilities include continuous monitoring of staff activities and recording violations of internal labor regulations. In addition, the introduction of a video surveillance system causes employees psychological discomfort from constant surveillance. Therefore, the installation of cameras is often a special measure that is applied to critical objects.

With punctuality, each employer applies to the extent of requirements. In production, on conveyor lines, delays are unacceptable. In creative directions, intellectual areas, solving problems depends largely on reflections, discussions, deliberation. And not only from the timely arrival to work or from specific movements at the PC, and from the time spent - to score data.

Everything is individual. From that they choose methods of control over the actions of subordinates:

• time sheets, Journals are kept in paper form, manually. The method is outdated, but remains in the practice of small organizations;

• turnstiles are a convenient accounting tool. Skips one at a time, clearly shows the net time, takes into account delays, early departures, etc. And leaves the temptation to jump over the fence or crawl under it;

• video surveillance complements devices for passing people, reduces the number of violators of discipline. It is difficult to hide from the all-seeing camera lens. The system is installed not only at the checkpoint, but also in offices, other rooms, for example, with an elective access system;

• terminals for time recording are necessary if work is paid upon arrival and departure. When the equipment identifies a person twice a day. Basically, identification occurs by biometric properties: face shape, fingerprint, iris of the eye, etc.;

• meetings, councils, where instructions for the next day are drawn up the day before, an action plan is thought out;

• automatic programs for accounting of working hours. They are installed in the open, the personnel are notified that their actions are being monitored. Or they introduce spy programs that read personal e-mail, take screenshots of the computer screen. Spies are prohibited by law. The employer risks using them for control.

1.5. Automated time tracking

A person follows a person. This method of discipline control remained the only one for a very long time. T-12 and T-13 time sheets were filled in manually. The factor of support for friendship and mutual assistance “I know you well, I will not mark you late” and other reasons when it is simply impossible to track everyone, reduce the effectiveness of this method of control.

In addition, it is important for the manager to know that the employee is not just in the office, but is directly engaged in what he is obliged to do. Modern IT solutions can confirm this. They are for those whose office life takes place behind a PC that takes into account everything and everyone first:

• fixes on / off minutes as the beginning and end of the day;

• marks mouse movements as active activity. The immobility of the device speaks of "non-computer" time or a halt in business;

• identifies employees who are overburdened with responsibilities, which is why they are regularly delayed at work. And those team members who are objectively little involved in labor processes;

• identifies employees who are looking for a new position because of dissatisfaction with the present.

The advantages of automatic programs are obvious. It is necessary to choose such that it compiles a complete picture of the staffing table, the range of responsibilities, the timing of assignments and the work plan, takes into account the productivity of personnel.

How many employees work and how many hours they spend on outside activities; whether they are overloaded, which ultimately leads to burnout - this is information that is important to the employer. More and more often they conclude: it is necessary to control personnel. Without this, it is difficult to move the business forward, it is impossible to develop and be competitive. IT solutions come to the rescue. Automated systems monitor and analyze the productivity of each employee. According to statistics, if they are available, the staff begins to work more efficiently by 30%.

It is important for a manager to monitor employees during their hours at the enterprise for many reasons. And enforce compliance with productive accounting systems. They are important in achieving goals:

• to record the workload of each specialist in accordance with the established schedule, i.e. his punctuality or absence after a break, unreasonably long pauses;

• to strengthen discipline within the team;

• make management decisions in personnel matters, according to the daily routine;

• plan the employment of subordinates faster, draw up schedules;

• reduce financial losses.

For the most part, automatic time attendance systems (ASURV) monitor the movement of employees during certain periods. And they do not give an understanding of what exactly they are doing with computers, how high their efficiency is. In this respect, software products with “bonuses” are optimal. That is, ASURV studies what the working time is spent on, the resources of a particular user, their productivity and unproductiveness, analyzes the ratio of employment and completed tasks.

1.6. Types of monitoring

Employee performance monitoring

Whose activity to control? Who should you pay attention to more often? What processes should you track? Where and how can this be done? Questions that require answers for the development of the enterprise. We'll also have to figure out what is better to check: process or result? So, what types of work monitoring are distinguished:

• checking previously completed tasks;

• management of stages;

• analysis of the final result;

• periodic tracking;

• selective control.

With the pros and cons of the indicated methods, you need to enlist the consent of subordinates. First, to comply with the law, to avoid delicate "espionage", negativity from subordinates. But, perhaps, the main thing that he wants to receive guidance from monitoring is the achievement of goals by an efficient team assembled together, in which they understand their specific tasks and the mission of the partnership.

Computer monitoring

Control of employee employment during working hours on an office computer complies with the law. But if you follow him already outside of working hours, this is considered an interference with personal life. Programs help maintain a balance.

PPs can be hidden. Or employees know about them, understand what the essence of such observation is. Firms in charge of classified information must monitor the PC. Hardly anyone wants important information to be "leaked" to competitors. Or the documents were deliberately deleted. Or they used the internal capacities and resources of the company for their own purposes.

Alas, more than one entrepreneur has come across these facts. Big business, medium or small - it doesn't matter. The priority is information security, intellectual property rights. Today it is already common for employees to install programs on their computers that allow remote administration.

1.7. Legality of monitoring

Employee performance monitoring: when is it legal? Computer monitoring of employees - is it generally legal?

It all depends on how you position this process yourself. There are concepts of personnel performance analysis, that is, loyal observation, “from the outside”. There is control, that is, surveillance in order to find “bad” employees and give a “hat”. And there is surveillance. And if the first two somehow get along with the laws of Ukraine, then the last option is definitely outlawed.

Legal Monitoring Rules

In fact, it is quite simple to comply with the rule of law in the process of monitoring employees. You need to be guided by only 2 basic rules.

#1. The employee must be aware of this.

You must obtain formal consent for any method of monitoring work. That is, discard the concept of covert surveillance at once - this is never a legal way. When hiring, the employee's contract must contain a clause about exactly how you intend to control the work. If you have implemented a control system after hiring, it is best to get the written consent of the entire team. This can be a collective consent, or from each employee separately.

Written form will help you in situations where an employee may apply for termination. This can happen if the dismissal is done incorrectly.

# 2. Control only working devices.

Installing a program to monitor the work of employees on a personal computer is not legal even with the consent of the computer owner. If you want to analyze working time on a personal day, let the employee turn off the program on his own. Then you can only control the working time without having access to the employee's personal data.

Confidentiality and personal data

By the way about that. For violation of personal space, surveillance, reading personal letters, viewing a person's personal documents, and so on, criminal responsibility follows. According to the Constitution of Ukraine and the State Civil Code, a person has the right to privacy and privacy. By violating these boundaries, a company can infringe on the privacy of not only its employee, but also other involved people (when it comes to personal correspondence, photos, videos, etc.).

That is why, with control, you need to be “you” and understand where and how to apply it. Avoid monitoring personal pages, instant messengers, video filming in non-working parts of the office. You can get confidential information by accident by overhearing a conversation or reading a message.

1.8. The importance of monitoring employees' working hours.

So what is the time of employees at work? The results of research on this topic are quite interesting.

Approximately 10-12% of the time is lost when checking e-mail, answering morning calls and starting the workflow. After that, about 10% go to conversations with employees about the current affairs of the company, discuss projects or tasks and their implementation. In addition, about 15% go to rest, relaxation and coffee breaks.

So how much time do you actually have left to do the job? In fact, not very much.

What threatens the irrational waste of time?

So why is there such a waste of working time? This is not only the cause of many problems, violations and errors, but also the first thing that reduces efficiency.

With this mode of operation, the tasks can be performed for a long time, in addition, there is no share of the required control over the process.

Delays in deadlines: Lack of observation of how tasks are performed, very often leads to the fact that their implementation is delayed or even disrupted.

Customer dissatisfaction: Another problem is customer dissatisfaction, as instead of responding to their inquiries, employees often respond to personal messages. Figure 1.1 shows one of the most common problems that arise due to deadlines and disorganization of performers.

Fig.1.1. Dissatisfied customer due to delays

The most common problems: they arise due to abstractions from the process, and lack of proper attention and control.

Why monitoring? It not only allows you to accurately monitor the entire work of the team, but also to quickly obtain information. When using it, as a rule, all data arrives very quickly and precisely that gives the chance to manage workflow effectively.

How to organize effective time monitoring?

The most effective way to organize and conduct monitoring is to use special systems designed for this purpose. They can not only provide the greatest amount of information, but also guarantee its accuracy and reliability. For example, the Yaware.TimeTracker system. What are its advantages?

The system allows you to accurately and continuously monitor the time of employees, determining exactly how they spend it. The time of the beginning and end of the working day, the time of breaks and coffee breaks are recorded, the actual time worked is displayed in the report.

Employee working time monitoring: why is it needed?

Monitoring working hours or evaluating the effectiveness of time use can help prevent a lot of problems.

If you notice in time that employees began to spend more time on forums, in social media. networks, you can quickly identify and eliminate the cause.

And there may be several reasons, for example:

• a “provocateur” has turned up in the team, who does not work himself and discourages colleagues;

• the staff is underloaded with work and is wasting time for personal purposes;

• instead of solving complex problems, which are "not clear" how to approach, employees choose social networks, etc.

• procrastination of managers;

• focus on secondary tasks.

The difficulty is that it is difficult to notice negative signals in time. The manager always has a lot of unresolved issues: suppliers, salaries, tax. It is not surprising that there is simply not enough energy for monitoring.

But there is a great circumstance: office workers spend most of the day at the computer. This means that you do not need to monitor the time yourself, an automatic system will do it for you!

Why is it important to constantly monitor employees' working hours?

Business leaders and managers agree that a competent time tracking system can positively affect labor productivity growth and bottom line performance indicators. Why is constant monitoring so necessary?

The manager has the right to make a choice: to remain in the dark about what employees or an entrepreneur are really doing - decisive steps towards understanding the real state of affairs. If employees devote a lot of time to solving personal issues or simply are not able to correctly use the provided working hours, then what kind of efficiency can we talk about.

Monitoring is a tool that can help you identify downtime and eliminate it to the fullest.

In fact, the loss of working time is great. If you count how many minutes one employee spends on the preparatory process, morning introductory calls, checking emails, communicating with other employees, and also resting in between, the percentage will be colossal. The third part of the day that could be spent on priority tasks is simply scattered in space.

Waste of time is a key cause of many problems and mistakes. It reduces efficiency. If you leave everything as it is, then the assigned tasks will be performed extremely slowly and constantly undergo alterations.

Monitoring is a real chance to optimize processes. Analysis and accounting of working time, and on a stable basis - the ability to use the full potential of the human resource.

Lack of oversight of the rational use of time by employees leads to destabilization, increased neglect of tasks, customer dissatisfaction and general business decline. If implemented, it will be possible to minimize errors and speed up the exchange of information, which will undoubtedly affect performance.

The choice in favor of systematizing time tracking is obvious for those managers who do not want to pay their employees money just like that.

1.9 Time tracking system: pros and cons

The main advantage of implementing such systems is that the manager has more information about the work of employees. He can see who is on time and who is late, who is spending time at work, and who is spending time talking with colleagues. It's easier to estimate how long a job takes and calculate averages. For example, the speed of response from technical support or the time to prepare a quarterly report from an accountant.

Employees, realizing that they are being watched, try to take a more responsible approach to work, use only work programs and not be distracted by social networks. In organizations with hourly wages, accounting systems make it easier to calculate earnings. The customer can be invoiced based on the data on the time spent on solving the problem by the personnel. If the need arises, at any time you can view the recordings of CCTV cameras, call history or reports from time trackers.

The time tracking system involves certain costs: the purchase of equipment for control, payment for software, hiring workers who will maintain the monitoring system and analyze the data.

Controlling employee time, like corporate dress code, is a factor that increases stress and decreases motivation. There are ways to circumvent: the employee who came to work first, turns on all computers; the cards are replaced, according to which the time of entry and exit is recorded; anonymizers are used to circumvent the ban on social networks; with the help of pendulum toys, movement of the mouse is imitated, and the most advanced ones write scripts that "work" for them. All of this greatly impairs the working atmosphere.

1.10. Existing IT solutions for working time control.

Manually control the work of a small team is a feasible task for the immediate supervisor or one assigned to specialists. But if the staff of the enterprise has more than one hundred people who also work in branches or remotely, to cope with such a task by force only automation.

In addition, monitoring the work of staff is required not only to record the effectiveness of the individual employee, but also to ensure the safety of the enterprise. Two tasks simultaneously perform IT solutions to monitor productivity during working hours. Automated systems record information, analyze and generate reports.

The cost of implementing an automatic program for monitoring working hours is covered by minimizing the costs of the employer associated with the lack of staff in the workplace: absenteeism, delays, long breaks. IT tools capture and analyze the data on the basis of which management encourages the "leaders" of the office, thereby motivating the whole team to work more productively and disciplined.

Table 1.2.

Problems that allow you to detect automated IT systems.

Problem	Description
Violators Disciplin	Employees who are systematically late or go home earlier than necessary, heavy smokers or coffee lovers, the duration of "five-minute" breaks in which exceeds the duration of work.
Employees engaged in other affairs	Employees who solve personal affairs during working hours, read news and social media feeds, have friendly, not business correspondence in messengers, play on the computer
Overloading employees	Employees who are so overwhelmed with work tasks and are forced to stay constantly at work, which threatens burnout and - in extreme cases - nervous breakdown
Dissatisfied employees	Employees who are dissatisfied with the position on certain issues and are busy looking for a new job.

Opportunities of "KIB SearchInform" are even wider. In the ProfileCenter module you can analyze the psychological characteristics of employees and create risk groups that require special control. WorktimeMonitor allows you to monitor all the actions of the employee at his workplace in real time, as well as to analyze data on staff activity in retrospect. Each module has its own "area of responsibility". Thus, the SearchInform ActivityMonitor module allows you to get comprehensive information about employee activity in certain applications during the day. Another component, SearchInform MonitorSniffer, allows you to take screenshots of a user with running applications. SearchInform Keylogger allows you to intercept all data entered by the user on the keyboard, as well as all data copied by the user to the clipboard. Figure 1.2 shows the interface of the WorktimeMonitor application from SearchInform.

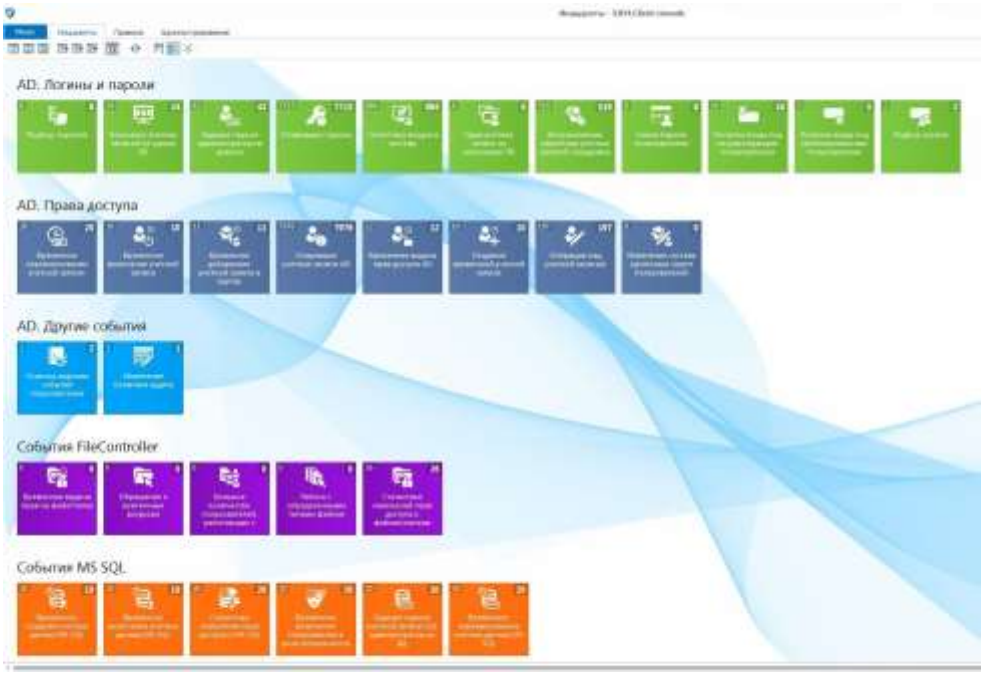


Figure 1.2. WorktimeMonitor interface

Automated monitoring of working hours is not limited to the installation of specialized software and also includes the installation of control equipment at the entrance / exit of the building; equipment of checkpoints; use of a system of personal identifiers for each employee; fixing the movement of personnel while working on the territory of the enterprise.

1.11. Let's summarize

Each owner wants to be aware of what his employees are doing, how effectively they use their working time. The oldest and simplest time tracking system is the maintenance of special timesheets, but they are of little information, they only reflect the presence of a person at the workplace.

There are many additional control methods, each with its own advantages and disadvantages. You need to choose the best method based on the specifics of the organization's work: there is enough pass system in production, in the IT sphere it is more important to keep track of the time spent on a specific task.

In addition to the obvious advantages: increased work efficiency, increased control, the ability to improve accounting; the introduction of a monitoring system has its drawbacks: implementation costs, increased stress levels among employees, and a deterioration in the psychological climate. All this must be taken into account in order to develop an accounting system that will help to control time in the best possible way and, based on its analysis, achieve maximum efficiency.

Conclusions

The first section of the diploma project described the need to implement a system of time control of employees due to the fact that it is almost impossible for a person to cope with this task alone. Such software is needed primarily to optimize the workflow, increase efficiency and meet deadlines. This requires clear planning and control of the use of working time.

Control as part of the monitoring of working hours allows to solve such tasks as fixing the time of arrival at work, checking the timely return of the employee after lunch and presence at work throughout the day, determining the end time, identifying absences, dividing paid working hours by actual hours , holidays, sick leave, downtime due to the employer, etc.

I have identified the main problems that can be identified using an automated IT system to monitor employees' working hours: violators of discipline, identifying employees engaged in other matters, overloading employees and identifying employees who are dissatisfied with the position on certain issues and are busy looking for a new job.

PART 2
REQUIREMENTS FOR WHMS

2.1. Functional requirements for WHMS.

Functional Requirements are requirements that define the functionality or behavior of a software system that developers must create to enable users to perform their immediate responsibilities within the business requirements that exist in the company.

Functional requirements for registration and authorization:

• The user can register in the system

• The user can log in to the system

• The user can choose the language of the system at will

• To register, the user must enter a name, email, and password

• The user must enter a mail and password to log in

• When registering a user, he can choose the language of the system at will

Functional requirements for the main page:

• The user should see a list of their activities for the day

• The user puts on each activity the amount of time spent

• The user uses the button to send data about the time spent, which is stored in the database

• Every day the data on the main page is reset

• The user must enter the time spent from the drop-down list

• The user should see what activity has already been performed

Functional requirements for the user page:

• The user should see the full list of activities he can add

• The user should see a list of their activities

• The user can send a request to add the activity to their list

• The user can request to remove the activity from their list

• When adding or deleting an activity, the list of common activities must be updated

• The user can send a request for statistics of time spent on activities on the specified date

Functional requirements for the admin page:

• The administrator should see all user requests to add or remove activities

• The administrator can accept or reject user requests

2.2. Non-functional requirements for WHMS.

Non-functional requirements are those requirements to the system that determine the properties that the system must demonstrate, or such restrictions that the system must comply with that are not relevant to the behavior of the created system. For example, NFV can include performance, ease of maintenance, scalability, reliability, operational factors.

User interface requirements**:**

• The home page should contain a table with activity fields and fields for entering the time spent. Next to each activity should be a button to send data to the database.

• When hovering over a table, it should be expanded with scripts. When sending activity data, a row from the table must be dynamically deleted without refreshing the page.

• The user's page should contain a dynamic table that is updated without reloading the page. When you click the add or remove activity request, the button should change the color and status of the query.

• The field for requesting user statistics must have a “data picker”.

• When you request statistics, the response should be displayed dynamically without reloading the page.

• The profile page should only contain static user data that was obtained during authorization

• The logout button should clear the session to redirect to the authorization page.

Software interfaces:

The web system must be connected to the Postgersql database. Any web browser must be installed for the system to work. The system must be supported on all types of browsers, be cross-platform.

Productivity. Up to 60 users can be authorized in the system at the same time, queries to the database must be in parallel streams.

Reliability:

• All data from the system is stored in the PostgreSQL database. Database backup must be present.

• Registration and authorization requests must be logged to files stored on the server.

Accessibility. All queries to the database should be received in no more than 4 seconds. The difficulty of the search should be linear.

Security. All user passwords must be stored in encrypted form in the range from 30 to 40 characters.

All user session attributes must be encrypted in the local repository.

All data from the queries must be in the body of the query, must not be displayed in the URL.

Escort. All classes must be implemented on the principle of single dependence. one entity performs only its tasks to further expand the functionality.

Transfer. The front-end software must be server-independent in order to be able to operate the system on a mobile platform or desktop in the future.

Conclusions

In the course of work on this section, functional and non-functional requirements to the system of monitoring the processes of the technical department of the Internet provider were formed.

The functional requirements block is divided into registration and authorization requirements, home page requirements, user page requirements, user profile page requirements, and administrator page requirements.

The block of non-functional requirements is divided into requirements for the user interface, requirements for software interfaces, requirements for performance. System attributes such as reliability, security, traceability, availability, and portability are also described.

PART 3
STRUCTURE OF WHMS

3.1. Architecture of WHMS.

The software product is created according to the MVC template (model, view, controller).

This template is a scheme for dividing the data of the user interface and control logic into three separate components: model, view and controller - so that the modification of each component can be done independently.

The Model provides data and responds to controller commands by changing its state. View is responsible for displaying model data to the user in response to model changes. The Controller interprets the user's actions, notifying the model of the need for change.

Figure 3.1 shows a graphical representation of the MVC template.

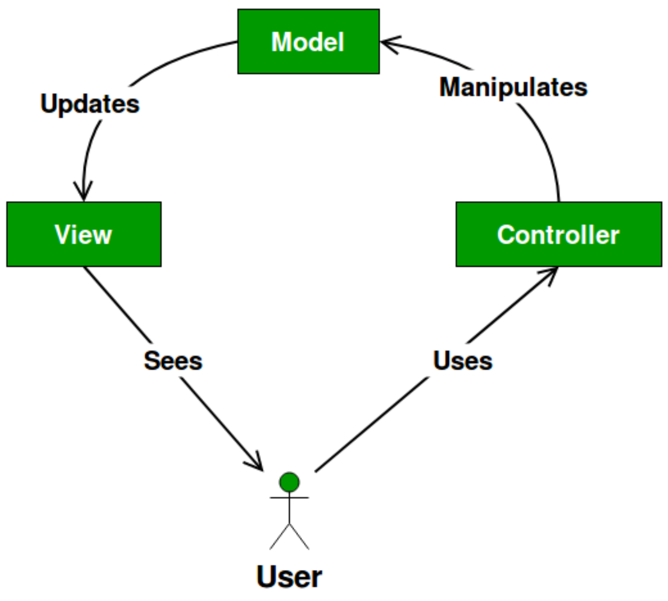


Fig.3.1. MVC template presentation

3.2 MVC Design pattern

MVC is a set of architectural ideas and principles for building complex information systems with a user interface;

MVC is an abbreviation that stands for Model-View-Controller.

Disclaimer: MVC is not a design pattern. MVC is precisely a set of architectural ideas and principles for building complex systems with a user interface. But for convenience, so as not to repeat every time: “A set of architectural ideas ...”, we will call the MVC pattern.

Let's start simple. What is behind the words Model-View-Controller?

When developing systems with a user interface, following the MVC pattern, you need to divide the system into three component parts. These, in turn, can be called modules or components. Say what you want, but divide by three. Each component will have its own purpose. Figure 3.2 shows the main functions of each of the parts.

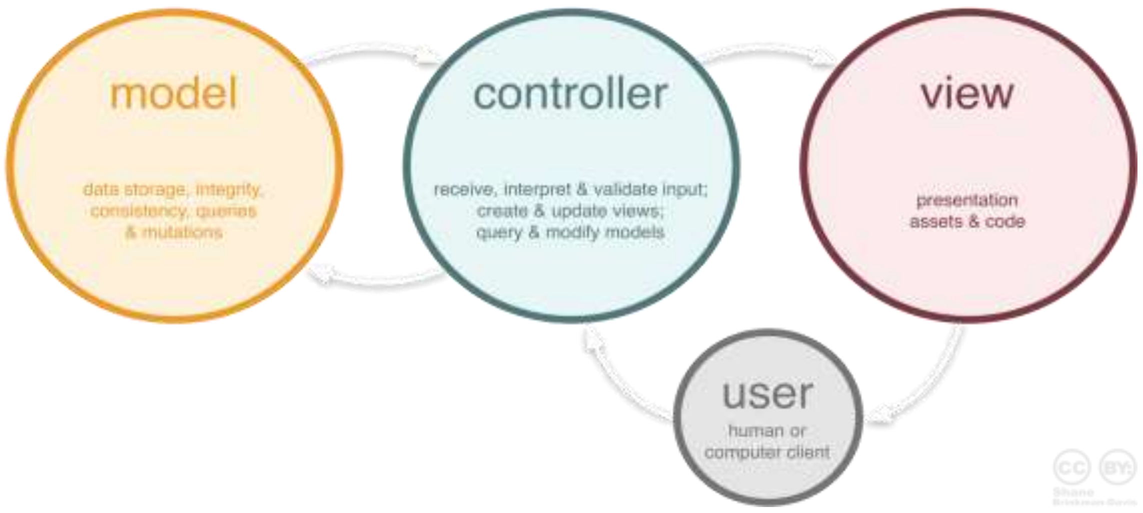

model
data storage, integrity, consistency, queries & mutations
controller
receive, interpret & validate input; create & update views; query & modify models
view
presentation assets & code
user
human or computer client

Fig 3.2. Programming in JAVA using the MVC Architecture

Model. The first component / module is the so-called model. It contains all the business logic of the application.

View. The second part of the system is the view. This module is responsible for displaying data to the user. Everything the user sees is generated by the view.

Controller. The third link in this chain is the controller. It stores the code that is responsible for processing user actions (any user action in the system is processed in the controller).

The model is the most independent part of the system. So independent that it doesn't have to know anything about the View and Controller modules. The model is so independent that its developers may know almost nothing about the View and the Controller.

The main purpose of the View is to provide information from the Model in a user-friendly format. The main limitation of View is that it shouldn't change the model in any way.

The main purpose of the Controller is to process user actions. It is through the Controller that the user makes changes to the model. More precisely, in the data that is stored in the model.

From all this, a completely logical conclusion can be drawn. A complex system must be divided into modules.

Let's briefly describe the steps to achieve this separation.

Step 1. Separate the business logic of the application from the user interface

The key idea of MVC is that any application with a user interface can be, as a first approximation, divided into 2 modules: a module responsible for implementing the application's business logic and a user interface.

The first module will implement the main functionality of the application. This module will be the core of the system, which implements the application domain model. In the MVC concept, this module will be our letter M, i.e. model.

The second module will implement the entire user interface, including displaying data to the user and the logic of user interaction with the application.

The main purpose of this separation is to ensure that the core of the system (Model in MVC terminology) can be independently developed and tested.

Step 2. Using the Observer pattern, achieve even greater model independence and synchronization of user interfaces

Here we are pursuing 2 goals:

Achieve even greater model independence.

Synchronize user interfaces.

The following example will help you understand what is meant by user interface synchronization.

Suppose we buy a movie ticket over the Internet and see the number of available seats in the movie theater. Someone else can buy a ticket to the cinema at the same time as us. If that someone buys a ticket before us, we would like to see that the number of available seats for our session has decreased.

Now let's think about how this can be implemented inside the program. Suppose we have a system core (our model) and an interface (a web page where we make a purchase).

On the site, 2 users simultaneously choose a place.

The first user bought a ticket. The second user needs to display this information on the page.

How is this supposed to happen?

If we update the interface from the core of the system, our core, our model, will be dependent on the interface. As you develop and test your model, you will have to keep in mind various ways to update the interface.

To achieve this, you need to implement the Observer pattern. With its help, the model sends notifications of changes to all subscribers. The interface, being such a subscriber, will receive a notification and update.

The Observer pattern allows the model, on the one hand, to inform the interface (view and controller) that changes have taken place in it, and on the other hand, it actually “knows nothing” about them, and thus remains independent. On the other hand, it will keep the user interfaces in sync.

Step 3. Splitting the interface into View and Controller

We continue to divide the application into modules, but at a lower level of the hierarchy. In this step, the user interface (which was separated into a separate module in step 1) is divided into a view and a controller.

It is difficult to draw a strict line between the view and the controller. If we talk about the fact that the view is what the user sees, and the controller is the mechanism through which the user can interact with the system, you can find some contradiction.

Controls, such as buttons on a web page or the virtual keyboard on a phone screen, are essentially part of a controller. But they are as visible to the user as any part of the view.

This is more about functional separation. The main task of the user interface is to provide user interaction with the system. This means that the interface has only 2 functions:

- display and conveniently display information about the system to the user;
- enter user data and commands into the system (transfer them to the system);

These functions determine how the interface should be divided into modules.

As a result, the system architecture looks like this in Fig 3.3. :

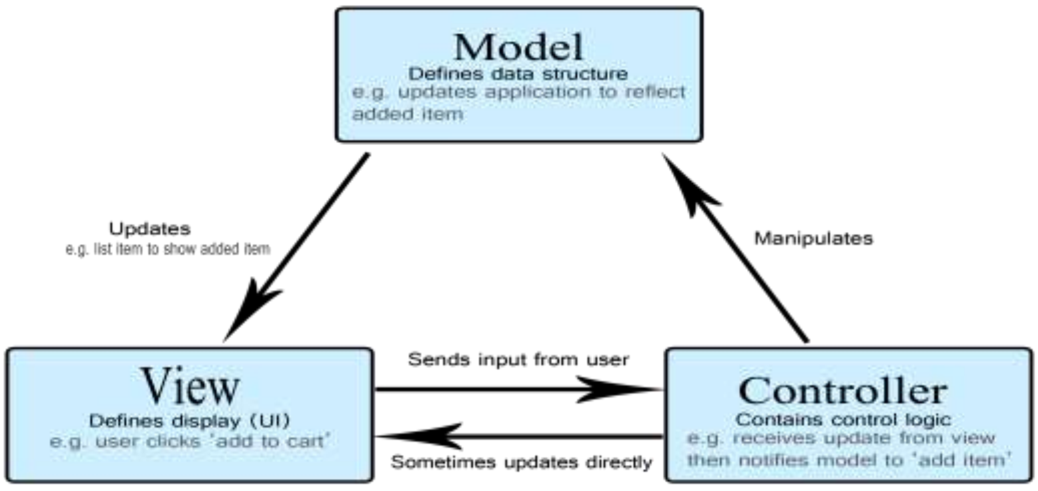

Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Fig 3.3. Representation of system architecture

So, we have an application of three modules called Model, View and Controller.

To summarize:

- Following the principles of MVC, the system should be divided into modules.
- The most important and independent module should be the model.
- The model is the core of the system. You need the ability to develop and test it independently of the interface.
- To do this, at the first step of segregating the system, you need to divide it into a model and an interface.

- Next, using the Observer pattern, we harden the model in its independence and get the synchronization of user interfaces.
- The third step is to divide the interface into a controller and a view.
- All that is needed to enter information from the user into the system is into the controller.
- All that is for the output of information from the system to the user is in the form.

A little about the relationship of the View and Controller with the Model

When the user enters information through the controller, they are making changes to the model. At least the user is making changes to the model data. When the user receives information through the interface elements (through the View), the user receives information about the model data. How does this happen? How do the View and Controller interact with the model? After all, it cannot be that the View classes directly use the methods of the Model classes to read / write data, otherwise there can be no question of any independence of the Model. The model is a closely related set of classes to which, in an amicable way, neither the View nor the Controller should have access. To link the Model with the View and Controller, you must implement the Facade design pattern. The facade of the model will be the very layer between the Model and the interface through which the View receives data in a convenient format, and the Controller changes the data by calling the necessary methods of the facade.

MVC – what’s the profit?

The main purpose of following the MVC principles is to separate the implementation of the business logic of the application (model) from its visualization (view).

This separation will increase the reusability of the code.

The benefits of using MVC are most obvious in cases where the user needs to provide the same data in different forms. For example, in the form of a table, graph or chart (using different views). At the same time, without affecting the implementation of views, you can change the reactions to user actions (clicking with the mouse on a button, entering data).

If you follow the principles of MVC, you can simplify the writing of programs, increase the readability of the code, and make it easier to extend and maintain the system in the future.

In the final article of the series "Introduction to Enterprise Development", we will look at the implementation of MVC using the example of Spring-MVC.

Advantages and disadvantages of MVC architecture

There are few advantages and disadvantages of MVC architecture as every architecture has. First, Let’s see it’s advantages.

Advantages of MVC architecture:

- Development of the application becomes fast.
- Easy for multiple developers to collaborate and work together.
- Easier to Update the application.
- Easier to Debug as we have multiple levels properly written in the application.

Disadvantages of MVC architecture:

- It is hard to understand the MVC architecture.
- Must have strict rules on methods.

There is not much in the disadvantages part of the architecture. And the disadvantages are not so huge and are very easy to ignore in comparison with all the benefits we get.

If you have any questions regarding MVC, please feel free to use the comment section given below. We are happy to help!

3.3. Spring Framework

The development of industrial applications on the J2EE platform has always been considered quite difficult. The statement, of course, is controversial, because questions will immediately begin - who was considered, when and what was then a simple task ... But let's take it on faith, because otherwise it is difficult to explain why it was necessary to come up with the Spring Framework. So, it was invented specifically in order to simplify the development of enterprise applications.

Spring is, one might say, an architectural framework, because it was invented not so much to perform some application task (as, say, Struts, which is needed to write web applications), but to provide better scalability, the possibility of easier testing and easier integration with other frameworks (for example, the already mentioned Struts or Hibernate). This makes it easier to write large applications — developers simply avoid some of the challenges of building enterprise applications instead of solving them.

But how does the Spring Framework enable developers to take advantage of these benefits? To do this, you need to look at the constituent parts of the framework and talk about each of them separately.

Spring Framework Components

Spring is a fairly large framework. No, what is really there - it's just huge. Because the creators of this framework have managed to cover almost every aspect of industrial Java application programming. Accordingly, the Spring Framework has a lot of components. So, here they are:

- IoC (Inversion of Control) container;
- AOP framework (including integration with AspectJ);
- Data Access framework;
- Transaction management;
- MVC framework;
- Remote Access framework;
- Batch processing;
- Authentication and Authorization Framework;
- Remote Management;
- Messaging framework;
- Testing framework.

In the illustration, you can see the framework diagram From this diagram, you can see that the first two components on our list are the most important - this is, in some way, the "heart" of the framework. Therefore, before talking further about each of the listed following these two components, let's take a closer look at what lies at its core - that is, with the Inversion of Control design pattern and aspect-oriented programming.

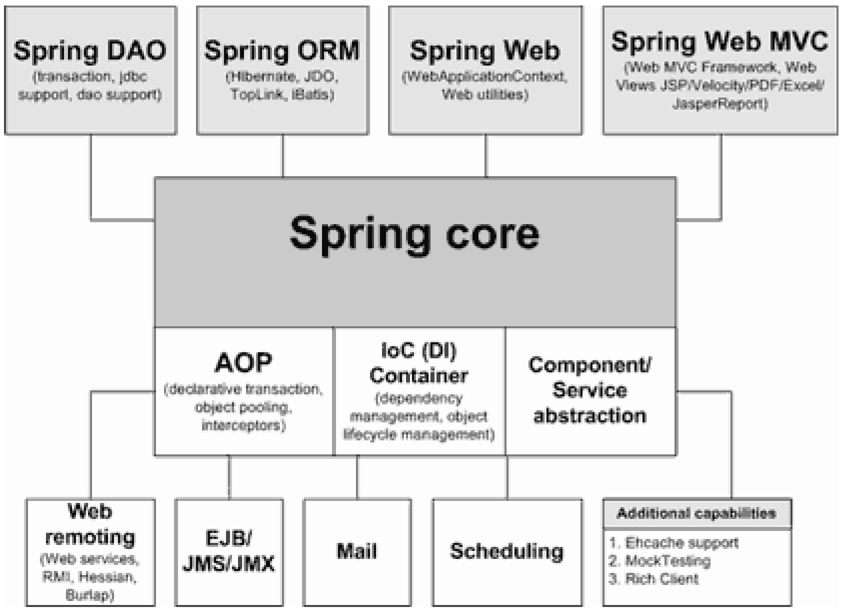

Spring DAO
(transaction, jdbc support, dao support)
Spring ORM
(Hibernate, JDO, TopLink, iBatis)
Spring Web
(WebApplicationContext, Web utilities)
Spring Web MVC
(Web MVC Framework, Web Views JSP/Velocity/PDF/Excel/ JasperReport)
Spring core
AOP
(declarative transaction, object pooling, interceptors)
IoC (DI) Container
(dependency management, object lifecycle management)
Component/ Service abstraction
Web remoting
(Web services, RMI, Hessian, Burlap)
EJB/ JMS/JMX
Mail
Scheduling
Additional capabilities
1. Ehcache support
2. MockTesting
3. Rich Client

Fig 3.4. Spring Framework Components

As you can see, the spring has a modular structure. This allows us to connect only those modules that we need for our application and not connect those that we obviously will not use. As far as I know, it was this approach that helped the spring to outflank its competitor at the time (EJB) and seize the lead. Because applications using EJB pulled a lot of dependencies, and in general they turned out to be slow and clumsy.

The image shows that the spring framework consists, as it were, of several modules:

- data access;
- web;
- core;
- and others.

Today we will get acquainted with some concepts of the main module, such as beans, context and others.

As you might have guessed, the data access module contains tools for working with data (mainly with databases), web - for working on the network (including for creating web applications, which will be discussed later).

In addition, there is also the so-called whole spring infrastructure: many other projects that are not officially included in the framework itself, but are seamlessly integrated into your spring project (for example, the same spring security for working with user authorization on the site).

3.4. Spring MVC Framework

The Spring MVC framework provides the architecture of the Model - View - Controller pattern using loosely coupled off-the-shelf components. The MVC pattern separates aspects of the application (input logic, business logic, and UI logic), while providing a loose connection between them, like it is shown in fig. 3.5.

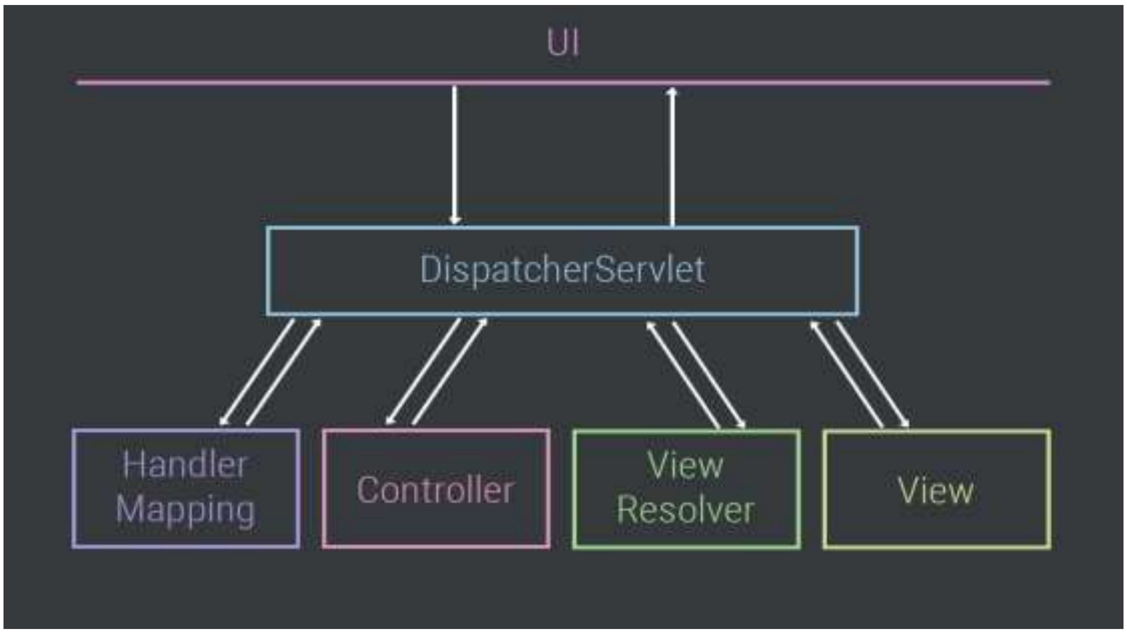


Fig 3.5. Spring MVC structure

Spring Web MVC, also known as Spring MVC, is a Spring web framework. This allows you to build anything web-related, from small websites to complex web services. It also supports frameworks like Spring Boot.

In the context of rendering HTML pages, say a user account page, this is what MVC looks like in Spring:

- Your Model contains the data that you want to display on the web page. However, the data is completely independent of your HTML, it's simple Java objects (like user objects) that make up your application.
- Your View will be an HTML template, which is the skeleton for your HTML page, written with a specific template library. These libraries allow you to include

placeholders in your templates that allow you to access model data, such as your username.

- Controller will be a @Controller annotated method that responds to the HTTP / account request and knows how to convert the HTTP request to Java objects, and your Java objects to HTML response.

Understanding the flow of Spring Web MVC

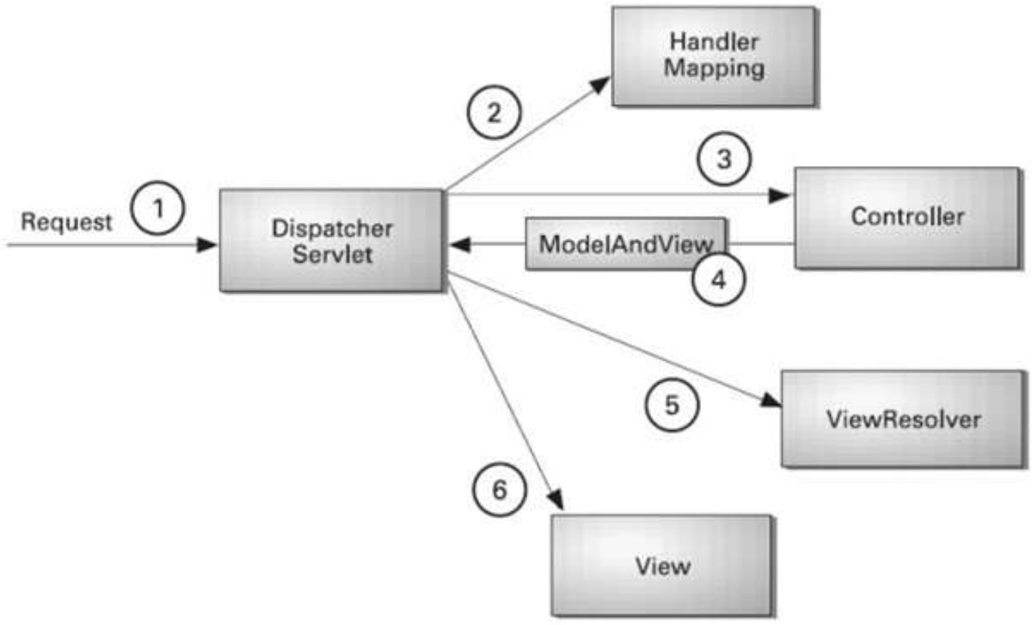


Fig 3.6. Incoming request is intercepted by the DispatcherServlet

As displayed in the figure, all the incoming request is intercepted by the DispatcherServlet that works as the front controller.The DispatcherServlet gets an entry of handler mapping from the XML file and forwards the request to the controller. The controller returns an object of ModelAndView.

The DispatcherServlet checks the entry of view resolver in the XML file and invokes the specified view component.

DispatcherServlet

The Spring Web Model-View-Controller (MVC) framework is built on top of the DispatcherServlet, which handles all HTTP requests and responses. The Spring Web MVC DispatcherServlet request processing workflow is shown in the following diagram:

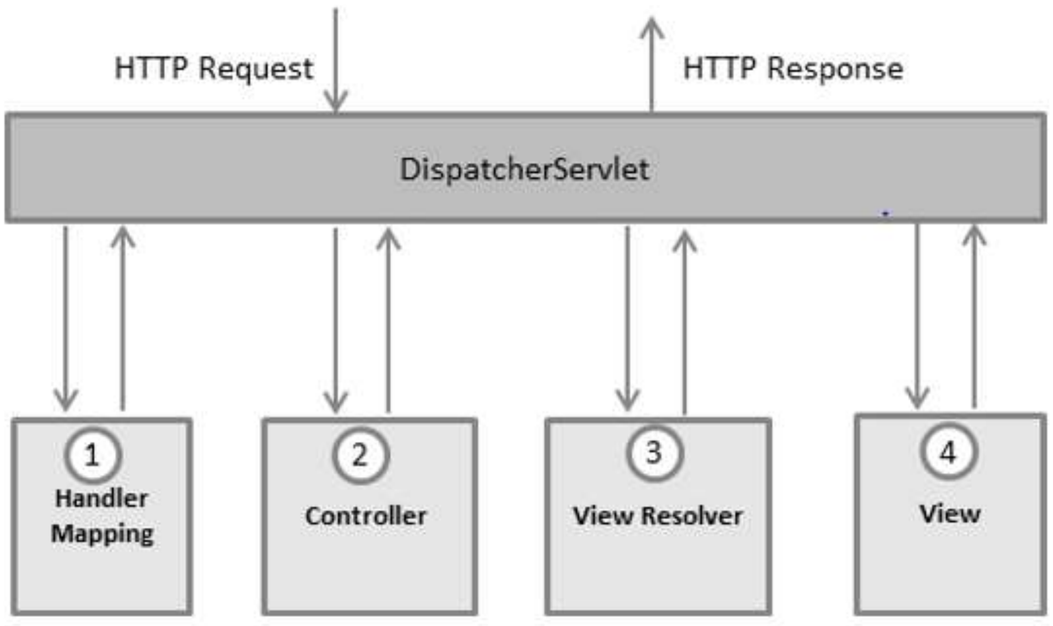


Fig 3.7. the sequence of events corresponding to the incoming HTTP request to the DispatcherServlet

The following is the sequence of events corresponding to the incoming HTTP request to the DispatcherServlet.

Upon receiving an HTTP request, the DispatcherServlet uses HandlerMapping to invoke the appropriate controller

The controller accepts the request and invokes the appropriate service methods based on the GET or POST method used. The service method sets the model data based on the defined business logic and returns the name of the view in the DispatcherServlet.

The DispatcherServlet will get help from the ViewResolver to get a specific view for the request.

Once the view is complete, the DispatcherServlet passes the model data to the view, which is ultimately displayed in the browser.

Upon receiving an HTTP request, the DispatcherServlet uses HandlerMapping to invoke the appropriate controller.

The controller accepts the request and calls the appropriate service methods based on the GET or POST method used. The service method sets the model data based on the defined business logic and returns the name of the view in the DispatcherServlet.

The DispatcherServlet will get help from the ViewResolver to get a specific view for the request

Once the view is complete, the DispatcherServlet passes the model data to the view, which is ultimately displayed in the browser.

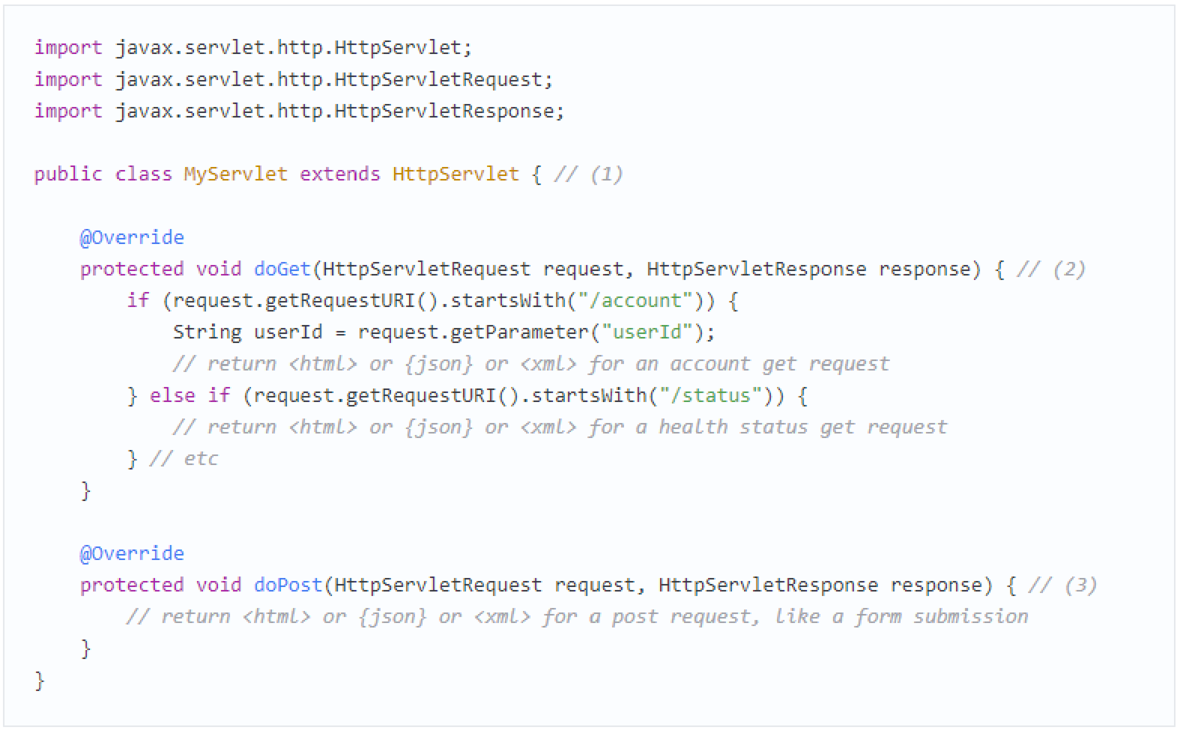
All of the above components, that is, HandlerMapping, Controller and ViewResolver, are part of the WebApplicationContext w, which is an extension of the simple ApplicationContext with some additional functionality required for web applications.

HttpServlet memo

Ignoring MVC for now: to write anything about HTTP with Java, you would use Servlets, or specifically HttpServlets (note for nitpicking: yes, there are other ways, thanks for the comment). Servlets can handle HTTP requests and can return an appropriate HTTP response to the browser or client

After writing your servlet, you must register it with a servlet container like Tomcat or Jetty. Servlet registration always includes a path to specify which URLs in your web application your servlet is responsible for. Let's assume the path is "/ *", so each incoming HTTP request to your application is handled by one servlet.

This is what this servlet might look like:



```java
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyServlet extends HttpServlet { // (1)

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) { // (2)
        if (request.getRequestURI().startsWith("/account")) {
            String userId = request.getParameter("userId");
            // return <html> or {json} or <xml> for an account get request
        } else if (request.getRequestURI().startsWith("/status")) {
            // return <html> or {json} or <xml> for a health status get request
        } // etc
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) { // (3)
        // return <html> or {json} or <xml> for a post request, like a form submission
    }
}
```

Your servlet should extend Java HttpServlet.

You can override the doGet () method to handle Http GET requests. With servlet mapping, "/" means for all GET requests. Thus, the request “/ status”, “/ info”, “/ account” will eventually be executed in the same doGet method.

You can override the doPost () method to handle Http POST requests. With servlet mapped "/" this means for all POST requests. Thus, submitting forms to "/ register", "/ submit-form", "/ password-recovery" will ultimately be done with the same doPost () method.

Handling every request to your application in just two ways is a bit cumbersome, and it takes quite a lot of work to get the MVC right

Also, your MyServlet (controller) needs to do quite a lot of manual HTTP specific connection, validating the request URI, looking at strings, converting requestParameters and responses, and so on.

It would be so much nicer if you didn't have to take care of all that plumbing and let Spring do it for you. This is where the DispatcherServlet comes in.

What does the DispatcherServlet do?

The Uber Controller in the Spring MVC framework is a servlet called DispatcherServlet.

It's called DispatcherServlet because it can literally handle any incoming HTTP request, parse its contents, and forward the data as cute little Java objects to the Controller class.

It's also smart enough to take the output from these controllers and convert it to HTML / JSON / XML, whichever comes up. The whole process looks like this (neglecting a lot of intermediate classes, because the DispatcherServlet doesn't do all the work itself.)

This is exactly what you want.

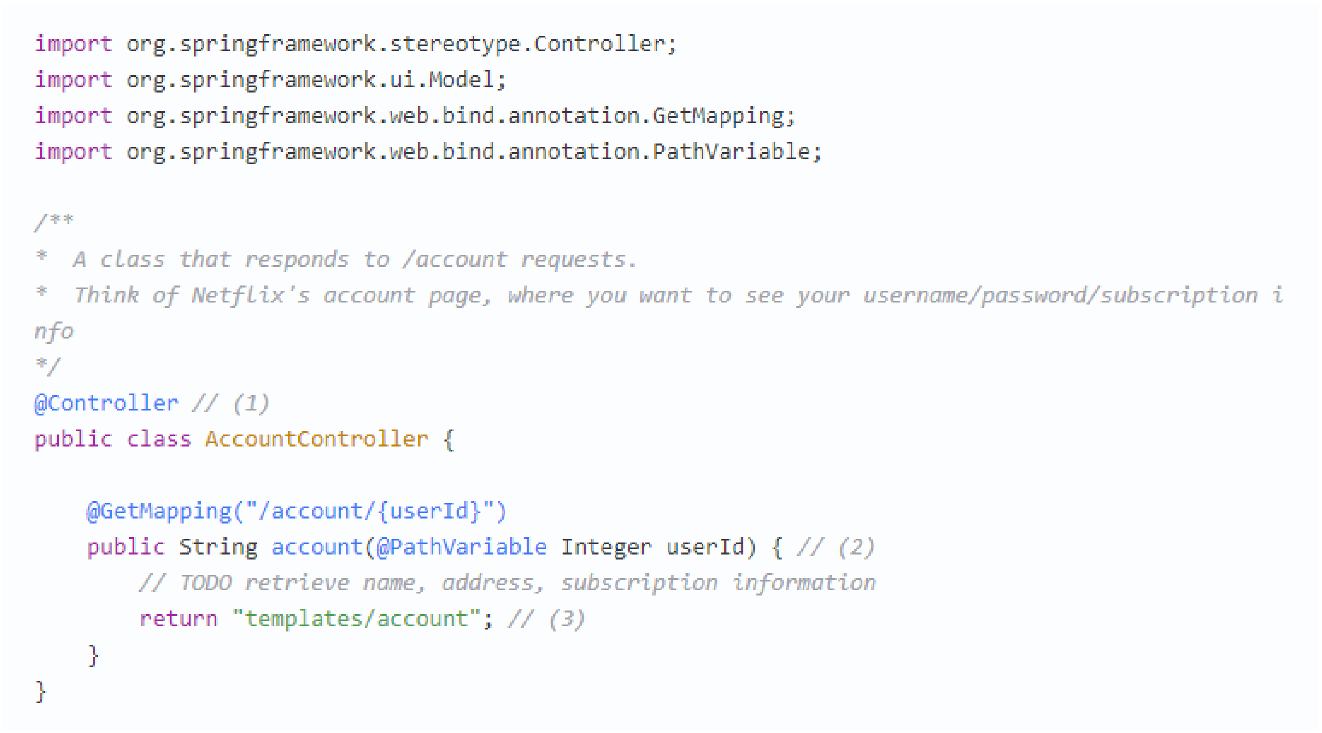
- Spring takes care of all the HTTP mechanics.
- You write your controllers.
- As well as views (templates) and models (your Java objects).

Let's take a look at these Controller classes in more detail.

How to write Controller classes

Finally, we can write our Controller class that handles / account requests

The / account page will be an HTML page with several dynamic variables like username, address, subscription information, etc.



```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

/**
*  A class that responds to /account requests.
*  Think of Netflix's account page, where you want to see your username/password/subscription i
nfo
*/
@Controller // (1)
public class AccountController {

    @GetMapping("/account/{userId}")
    public String account(@PathVariable Integer userId) { // (2)
        // TODO retrieve name, address, subscription information
        return "templates/account"; // (3)
    }
}
```

There's a LOT going on in those two lines, let's see what exactly.

You have an AccountController class that is annotated with @Controller. It tells Spring that this class wants to react to HTTP requests and responses so the DispatcherServlet is aware of it.

You have a simple Java method called account (). More interestingly, the method is annotated with @GetMapping. This tells the DispatcherServlet that all requests like / account / {userId} should be handled by this controller method. Moreover, dispatcherServlet will take {userId}, convert it to an integer and use it as a method parameter!

Our Java method returns a string named account. Actually it is not just a string, but a link to a view (HTML template). Let's take a look at these templates for a second.

How to generate HTML view with Spring Web MVC

Spring MVC assumes by default that you want to render some HTML. And of course you don't want to render HTML strings yourself using string concatenation, rather you want to use a templating framework like Velocity or Freemarker. Spring integrates with all of these technologies.

So, you could have the following view (pattern):

classpath: /templates/account.vm

```
<html>
  <body>
    Hello $user.name, this is your account!
    <!-- list subscriptions etc -->
  </body>
</html>
```

This is a very simple HTML page that contains one variable, $ user.name. How does this variable get into the template? So far, this was not in our account controller. Let's put this in.

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@Controller
public class AccountController {

    @GetMapping("/account/{userId}")
    public String account(Model model, @PathVariable Integer userId) { // (1)
        // TODO validate user id
        model.addAttribute("user", userDao.findById(userId)); // (2)
        return "templates/account"; // (3)
    }
}
```

1. Spring can automatically inject the "Model" parameter into your controller methods. As mentioned earlier, the model contains whatever data you would like to present in your view, i.e. your template.

2. Spring model behaves almost like a map, you just add all your data to it and then you can reference the display keys from your template.

3. Again, this is a link to your view, the account template we wrote above.

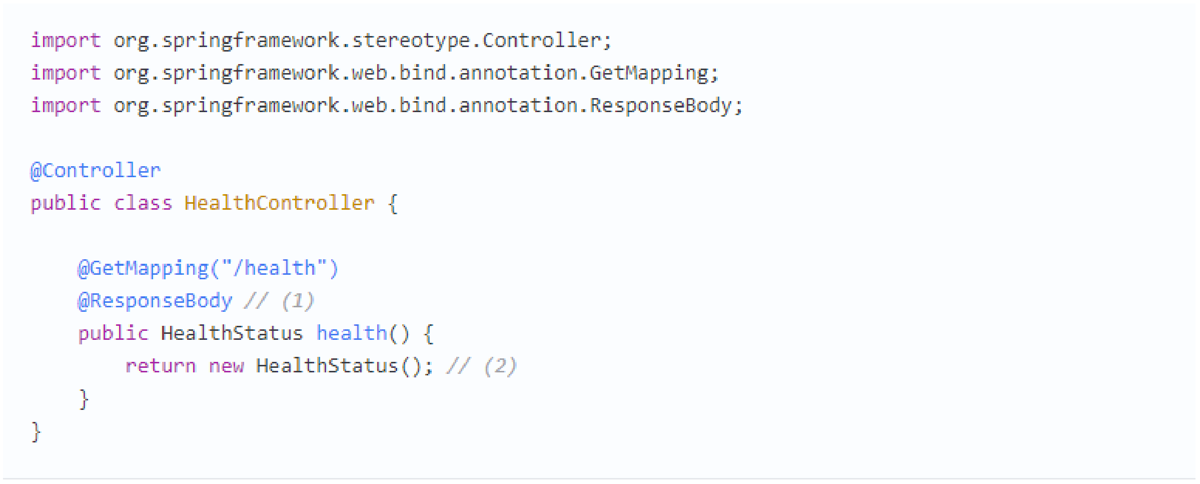
That's it, Spring MVC development is complete!

How to generate JSON / XML (views) with Spring Web MVC

With web services, you are not generating HTML, you are generating XML or JSON. It's pretty simple, with Spring MVC as well.

Of course you need an appropriate library like Jackson added to your project, but then you can just annotate your Controller with an additional annotation to signal

Spring: please convert my Java objects directly to XML / JSON instead of me giving you link to the view.



```java
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class HealthController {

    @GetMapping("/health")
    @ResponseBody // (1)
    public HealthStatus health() {
        return new HealthStatus(); // (2)
    }
}
```

- This time, you add an additional @ResponseBody annotation to the controller method that tells Spring that you want to write your Java HealthStatus object directly to the HttpResponse (for example, as XML or JSON).
- You are just returning a simple Java object inside your method, not a string reference to your view.

But how does Spring know if it should be returning XML, JSON, or whatever?

How Spring MVC Content Negotiation Works

There are many ways that you, as a client, can tell Spring MVC which response format you want when making a request to your Spring MVC application.

By specifying an Accept header such as "Accept: application / json" or "Accept: application / xml". This works out of the box, but requires certain libraries in your classpath to support XML or JSON sorting.

By adding url extension to your request path like /health.json or /health.xml. This requires some configuration on the Spring MVC side to work.

By adding a request parameter to the request path like / health? Format = json. This requires some configuration on the Spring MVC side to work.

Then Spring knows: you have annotated this method with @ResponseBody and the client wants "JSON". I have a classpath library like Jackson that can render JSON? If so, okay, let's convert HealthStatus to JSON. Otherwise, throw an exception.

If you want to know more about content negotiation, check out the official documentation.

What type of HTTP request input does Spring understand?

Spring MVC understands basically everything that HTTP offers - using third party libraries.

This means you can throw JSON, XML, or HTTP (Multipart) Fileupload request bodies into it, and Spring conveniently converts that input to Java objects.

What HTTP responses can Spring MVC write?

Spring MVC can write whatever you want to HttpServletResponse - using third party libraries.

Be it HTML, JSON, XML, or even WebSocket response bodies. What's more, it takes your Java objects and generates these response bodies for you.

What about other Spring MVC concepts?

The official Spring MVC documentation literally contains hundreds of pages describing how the web framework works.

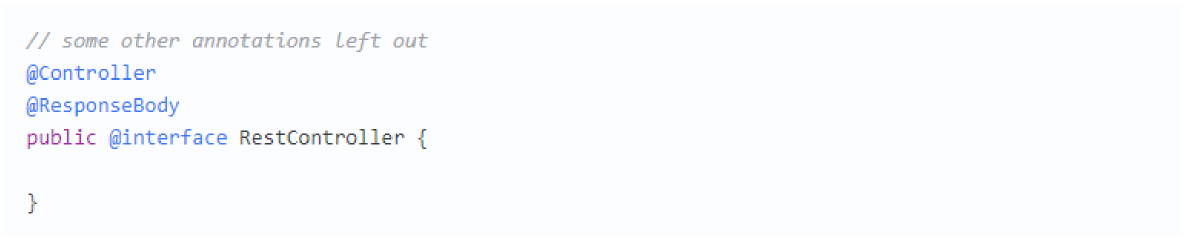
So if you want to know more about RequestParams, Models, Views, ViewHandlers, RootContexts, Filters, Caching, and Security, I invite you to check it out. It is simply beyond the scope of this guide to cover everything.

However, the FAQ section of this guide has answers to a few additional questions.

Optional: Briefly about Spring Boot

While this tutorial is not about Spring Boot, let's digress a bit by looking at the source of the @RestController annotation that you may have already come across in your Spring Boot project.

You might think that it is inherent in Spring Boot (as I mistakenly did for too long), but as ше шы correctly pointed out, it is part of a plain old Spring MVC. Here is its source:



```java
// some other annotations left out
@Controller
@ResponseBody
public @interface RestController {

}
```

That's right, Spring MVC @RestController is nothing more than Spring MVC @Controller combined with Spring MVC @ResponseBody annotation - although you might think it's something related to Spring Boot.

Advantages of Spring MVC Framework

Let's see some of the advantages of Spring MVC Framework:

- Separate roles - The Spring MVC separates each role, where the model object, controller, command object, view resolver, DispatcherServlet, validator, etc. can be fulfilled by a specialized object.
- Light-weight - It uses light-weight servlet container to develop and deploy your application.
- Powerful Configuration - It provides a robust configuration for both framework and application classes that includes easy referencing across contexts, such as from web controllers to business objects and validators.
- Rapid development - The Spring MVC facilitates fast and parallel development.
- Reusable business code - Instead of creating new objects, it allows us to use the existing business objects.
- Easy to test - In Spring, generally we create JavaBeans classes that enable you to inject test data using the setter methods.
- Flexible Mapping - It provides the specific annotations that easily redirect the page.

Summary: Spring MVC

Spring MVC is a good old MVC framework that makes it pretty easy to write HTML / JSON / XML websites or web services. It integrates nicely with the Spring dependency injection container, including all of its helper utilities.

In short: it allows you to focus on writing Java classes instead of dealing with slate servlet code, i.e. dealing with Http requests and responses, and gives you a good separation of concerns between your model and views.

3.5. Dependency Injection and Spring

Dependency Injection (DI), also often called Inversion of Control (IoC) (in the Russian-speaking world, the term dependency injection is also used), is one of the fundamental principles of modern software containers. This approach is described in detail in one of the articles by Martin Fowler, a recognized guru and mastermind of many modern software trends.

Currently, a large number of software products exist and continue to emerge, which are based on the ability to configure dependencies between the components of an applied software system not at the development stage, but either during its launch, or in general already in the process of functioning. As a result of this approach, ensuring the interconnection of various components of the system does not depend on a specific technology (for example, this is the case in EJB2) and, for example, at the testing stage it is not difficult to replace some objects that depend on the underlying layer with stubs.

Initially, the developers of open source software were involved in the implementation of this approach, such containers as Pico Container, Spring, a little later Guice in the Java world, Castle Unity, Autofac for the .NET platform, Copland for Python, The IOC Module for Perl and many others. This principle also left its mark on the creation of new technologies, for example, the experts of the Java Community Process could not help but use the pattern of decreasing dependencies between system components when creating the new standard JSR 220: Enterprise JavaBeansTM 3.0, and later JSR 330: Dependency Injection for Java, which should be part of the new J2SE 7.0 release. At the same time, for most software developers, especially from the java world, the mention of dependency injection is still primarily associated with the Spring Framework, since the use of Spring has actually become an unspoken standard in the development of J2EE applications (the numbers vary in various studies, usually no less than in about 75% of all newly created systems).

Dependency Injection Principle

This section will briefly introduce the basic concept of configuring dependencies, for a more detailed description, you can refer to the already mentioned article by Martin Fowler. The underlying approach can be summed up in the famous Hollywood principle: "Don't call me, I'll call myself." If in traditional systems a component receives a direct link to the location of objects or services necessary for operation, or refers to a service locator and requests a link to the implementation of a certain type of service, then in the case of a DI container, the environment is based on configuration data that can be provided to the system in the form xml file or annotations, itself knows about the necessary relationships between the components, and provides the necessary objects during initialization or execution. Using dependency injection introduces additional flexibility to the system, since it makes it easier to create alternative service implementations, and allows you to specify in the configuration which implementation should be used.

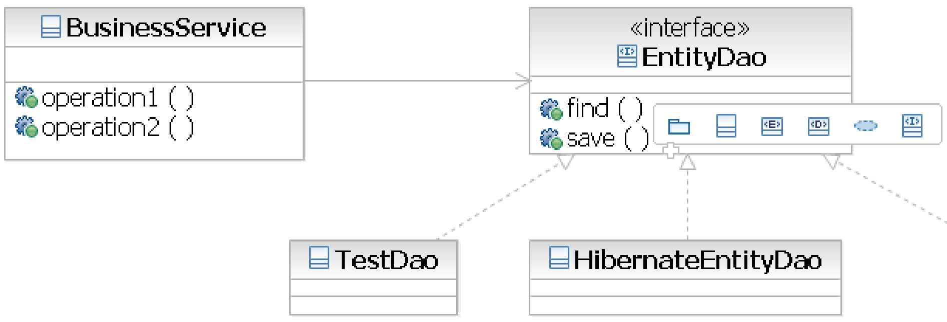


Figure 3.8. An example of using Dependency Injection

Figure 3.8. is an example of a class diagram that reflects the dependency that can be found in almost any real system. As it follows from it, the BusinessService service uses the EntityDao interface to work with the database, and there are several implementations of this interface: TestDao, HibernateDao, and JdbcDao. TestDao is used during service testing, and the choice between HibernateDao and JdbcDao should preferably be made by the system administrator based on his preferences or objective performance indicators. In the case when the BusinessService itself creates an instance of the class that implements the EntityDao interface, the development, and most importantly, the development and maintenance of such a system becomes much more complicated, since then for testing it is necessary to create an instance of TestDao, then after successful testing, change the code of the BusinessService service, and then recompile it, which itself on its own may introduce a new error. Using JNDI inside the BusinessService (also called Dependency Lookup) solves the issue of recompiling after testing the system. This was the mainstream in the days of EJB2 containers, which did not support the principle of configuring dependencies. The component itself needs to ask the container where the objects necessary for operation are located. However, this increases the amount of service code in the system (glue code), that is, code that was not developed to solve the business requirements of the system, but to link together various parts of the system, handle exceptions, etc. In addition, testing BusinessService with this approach is only possible inside an EJB2 container, which makes organizing unit tests not at all trivial. The ServiceLocator pattern, which is often used in such containers, which encapsulates all JNDI calls inside, also allows not to inflate the size of the auxiliary code, however, the corresponding ServiceLocator method must still be called explicitly within the BusinessService.

The most beautiful and effective solution is to use the dependency injection principle and a container that supports this principle. In this case, an instance of the class that implements the EntityDao interface is initialized inside the container based on the configuration specified by the administrator. After that, the container automatically "injects" a reference to the created EntityDao implementation into the corresponding field of the BusinessService class.

The main advantages of using a container that implements the principle of dependency injection:

- Centralized and external dependency management, in the case of using text configuration files that are not subject to compilation, and can be individually adapted to different stages of development and operation of the system, without requiring recompilation. In the traditional approach, the necessary components are either initialized as needed - within the classes using them, or class factories are used, which is also not an ideal way, since then the using class also receives a dependency on these factories.
- Lack of supporting glue code. When using a container that supports the DI principle, the size of the code is significantly reduced, since all necessary dependencies are resolved and the components are "glued" together automatically.
- Improving testability of individual system components. Using dependency injection allows you to easily replace dependent components, which is especially useful

for organizing a unit test of the system. Consider the above example shown in Figure 1. In order to test the implementation of the BusinessService, you need to verify that it is working correctly with the various data that the service receives using the EntityDao interface. To do this, you can use an instance of the TestDao object, which simulates working with the database and provides the service with the necessary test data. Checking the correctness of the implementation of JdbcDao or HibernateEntityDao can be done in another test.

- Using the best software development techniques. Building applications using a DI container pushes the developer to make heavy use of interfaces. Typically, all the main components are defined using interfaces, then concrete implementations of these interfaces are created, which are linked using DI. Actually, such applications in the Java platform were possible before dependency injection and DI containers appeared. But using Spring or a similar container allows for a proven implementation of the DI pattern, allowing developers to concentrate on creating business logic.

Spring implementation of Dependency Injection.

In the world of the Spring Framework, the concept of a bean is actively used, which denotes a managed component, that is, a bean is created and initialized by a container and exists inside it. Usually, a bean is a java class, the development of which adhered to the rules defined by the JavaBean specification, but this is not a prerequisite (for example, it will be demonstrated later that it is not necessary to have a constructor that has no arguments).

The main interface responsible for managing beans and their dependencies in the Spring Framework is the BeanFactory. At some point in time, the application must initialize the class that implements this interface and provide information about the system configuration. During configuration, each initialized bean is assigned at least one name, although there can be any number of names. This name can be used by the application to access this component.

Configuring dependencies in Spring Framework

The relationships between components can be defined programmatically, but the most commonly used configuration file or annotations.

Application configuration methods

There are three main ways of configuring an application (that is, telling the spring exactly which objects we need to work):

- using xml files / configs;
- using java-configs;
- automatic configuration.

Spring developers arrange them in this order of priority:

the most prioritized way to give preference is automatic configuration;

if using automatic configuration it is not possible to correctly configure all possible beans - use java configuration (creating objects using java code);

well, the lowest priority method is the old-fashioned way, using xml configs.

In addition, spring allows you to combine these methods. For example, everything that can be configured automatically - let the spring do it itself, where you need to specify some special parameters - do it using java-configs, and besides, you can connect some legacy configs in xml format. In general, all this can be done quite flexibly. But still, if everything can be done with automatic configuration, use it.

I will only cover auto-configuration and java configs; xml configs are already used in almost every spring example on the Internet, and having understood how the java configuration works, there should be no problems in order to "read" an xml file that does the same.

Automatic configuration is used when the objects we need for work are objects of the classes we have written.

If we need some very specific logic to create an object of our class, or if we do not have the ability to mark some class with the annotation we need, which would be picked up by automatic configuration, this can be done in java-configs.

3.6. Component diagram.

In Figure 3.9. shows a diagram of the components of the diploma project, based on the principle of MVC, where MVC stands for model-view-controller (from the English. Model-view-controller).

This is a way of organizing code, which involves the selection of blocks responsible for solving various problems. One unit is responsible for these applications, another is responsible for the appearance, and the third controls the operation of the application. This principle is described in more detail in the previous paragraph of this section.

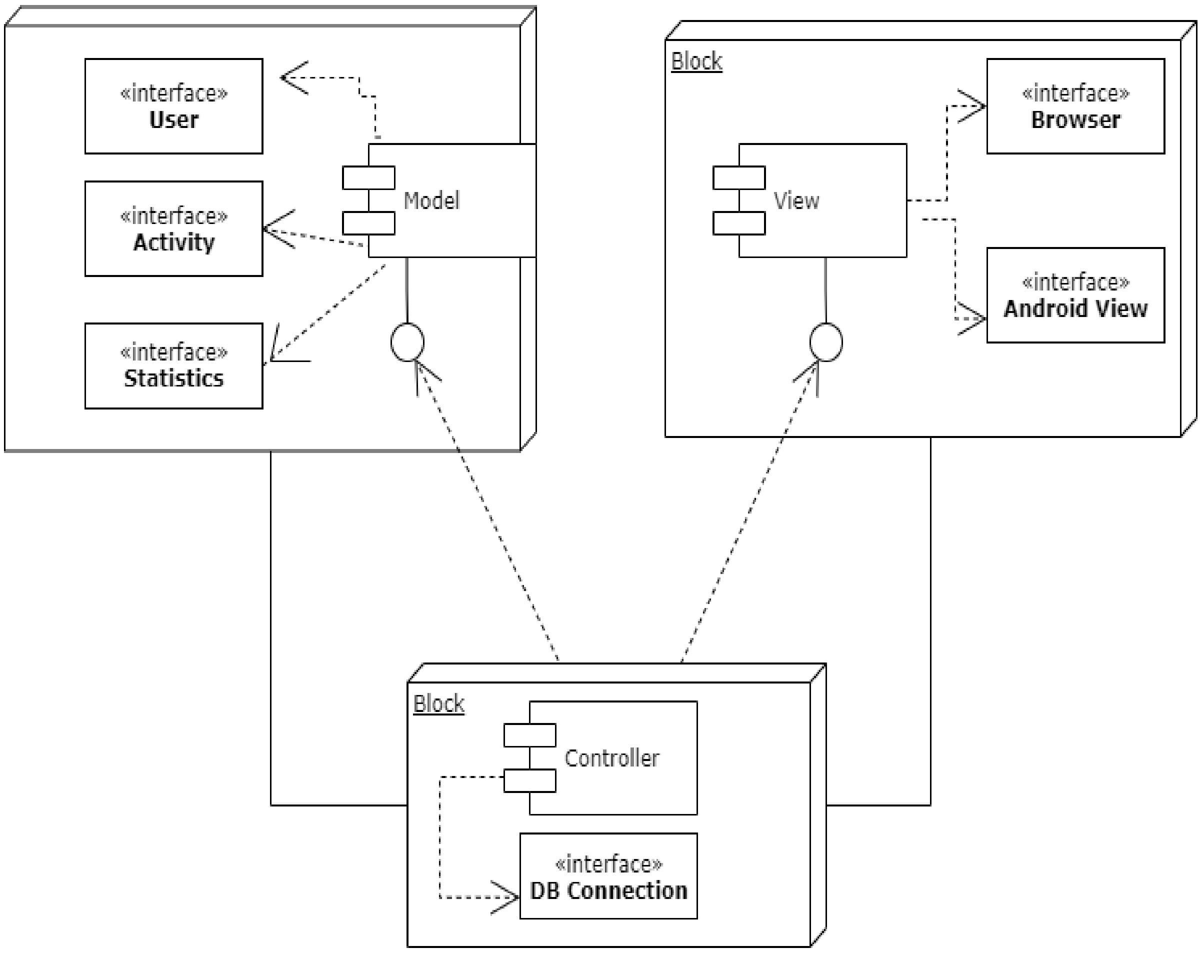


Fig.3.9. Diagram of diploma project components,based on the MVC principle

3.7. Class diagram.

In Figure 3.10. the diagram of classes of the diploma project is shown.

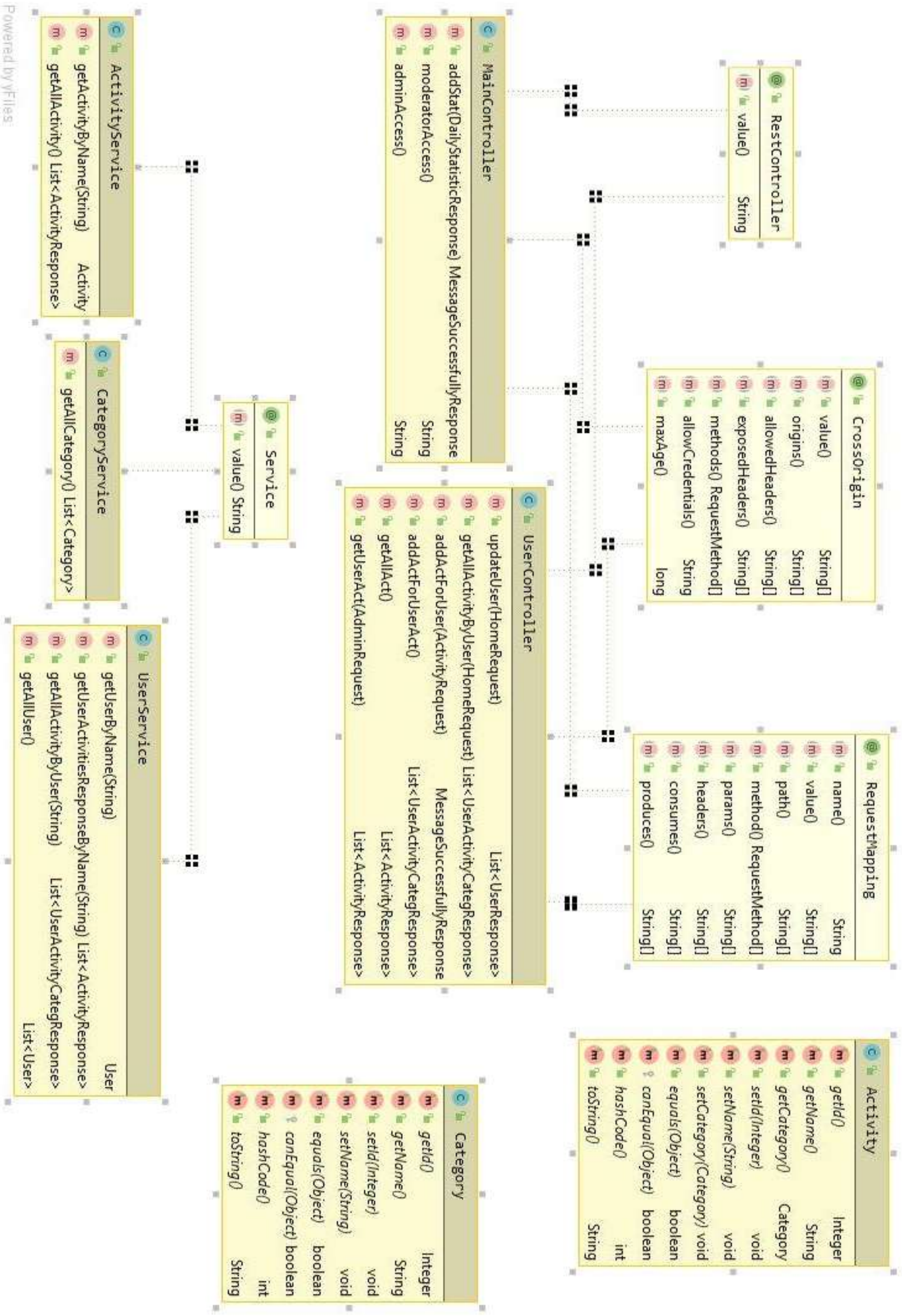


Fig.3.10. Diploma project class diagram

This diagram shows the structure of classes implemented on the principle of MVC. The main entities are User, Category, and Activity. Controller implementations are based on the RESTController interface for receiving and transmitting JSON objects. Relevant services have methods that delegate requests to the JPA repository, and it sends to the database itself.

Services implemented in a pattern - Singleton. All classes adhere to the principle of single dependence.

The Activity class stores the essence of the "Activity" which has a field with the name of the activity itself and a link to a specific category.

The Category class retains the essence of "Category". It has a field with the name of the category to which the activities refer.

The UserService class performs operations on objects with User user data: sorts by name, returns to the database or to the controller.

The ActivityService class receives objects of the "Activity" type from the controller, assigns them a certain category and writes them to the database. Also through this class data collection from a DB is carried out.

The CategoryService class receives a list of all categories from the database.

The MainController class is used to issue pages based on user roles. For the administrator - the administrator page, for the user - the user page.

The UserController class receives get- and post-requests from the client, collects the object, and sends it to the required service for its processing.

The RestController interface is used so that a regular controller can receive and send JSON objects.

CrossOrigin interface - is used to assemble objects of various formats.

The RequestMapping interface is used to retrieve the body of the query (body) and headers.

3.8. Web-system deployment diagram.

The components are implemented on the principle of independent servers. The user only needs a web browser to work on the system. After the browser, the request is sent to a dynamic node js server, which stores pages of information: HTML + Java Script. On the server there is a routing table on which requests are redirected to the Back end server implemented on Java + Spring Framework. The back end server connects to the database server and sends the response back to the front end server. Figure 3.4 shows a deployment diagram of a web system.

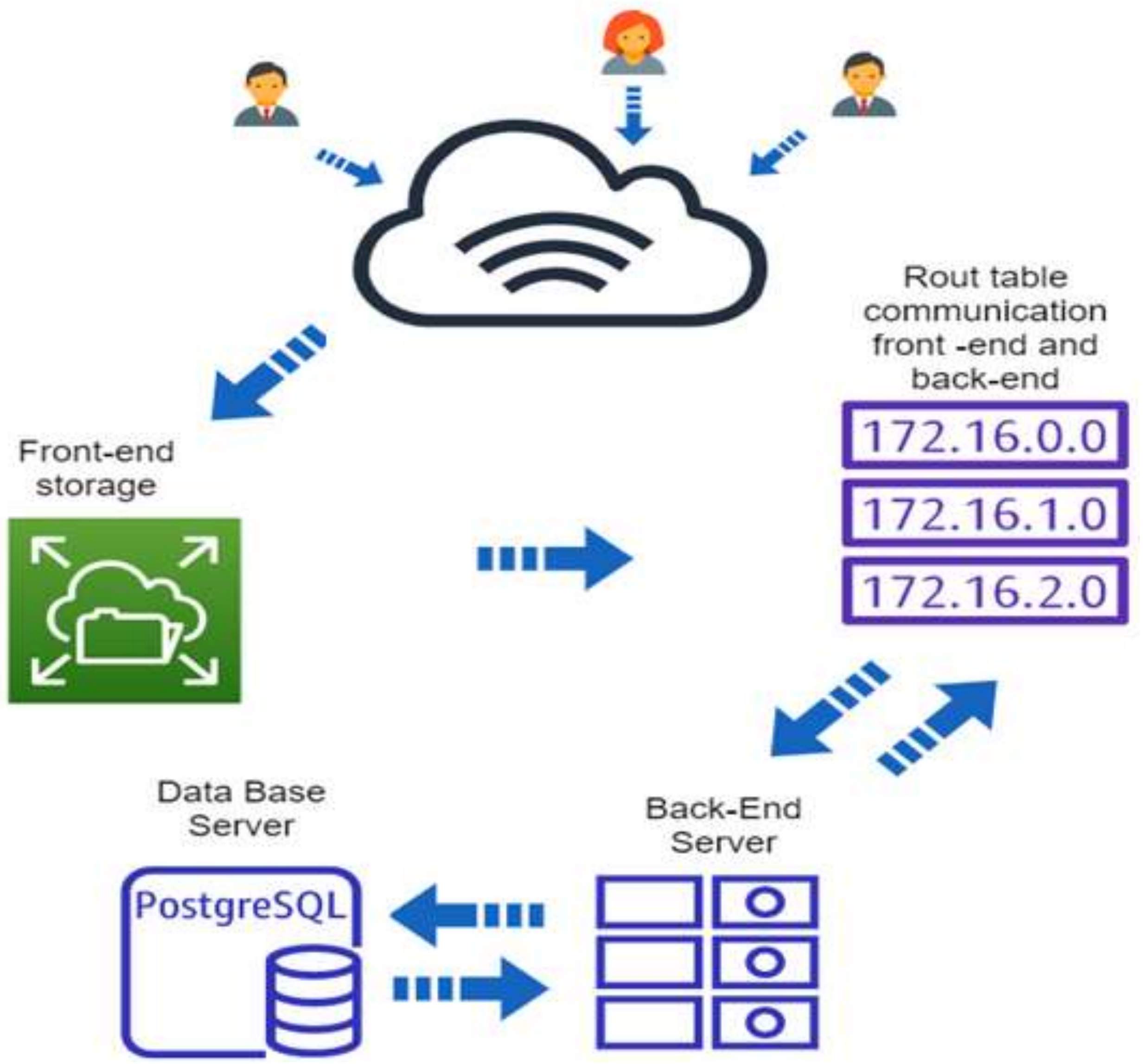


Fig.3.4. Web-system deployment diagram

Conclusions

In the third section of the diploma project it was determined that the system of monitoring the processes of the technical department of the Internet provider will be built using the architectural template MVC (model-representation-controller).

I also constructed a diagram of the components of a thesis project based on the MVC principle, which shows how the model provides data and responds to controller commands, view is responsible for displaying model data to the user, and the controller interprets user actions, notifying the model of changes.

To implement the future prototype of the system, a class diagram was constructed, which describes in detail the composition of the future system for monitoring the processes of the technical department of the Internet provider.

The deployment diagram showing the topology of the system and the distribution of its components by nodes, as well as the routes of information transfer between hardware nodes, was also presented and described in detail.

In the third section of the diploma project it was determined that the system of monitoring the processes of the technical department of the Internet provider will be built using the architectural template MVC (model-representation-controller).

I also constructed a diagram of the components of a thesis project based on the MVC principle, which shows how the model provides data and responds to controller commands, view is responsible for displaying model data to the user, and the controller interprets user actions, notifying the model of changes.

To implement the future prototype of the system, a class diagram was constructed, which describes in detail the composition of the future system for monitoring the processes of the technical department of the Internet provider.

The deployment diagram showing the topology of the system and the distribution of its components by nodes, as well as the routes of information transfer between hardware nodes, was also presented and described in detail.

PART 4
STRUCTURE OF WHMS

4.1. Development of a prototype WHMS.

To implement the prototype of the monitoring system of the technical department of the Internet provider, I took into account the functional and non-functional requirements created in the previous sections of the explanatory note, as well as UML diagrams: class diagram, web server deployment diagram and component diagram.

I implement the prototype SMRC in the Java programming language using the object-oriented programming environments IntellIJ Idea and Visual Code. As a result, I created seven classes and three interfaces. In table 4.1 I will give their names and description.

Table 4.1.

Classes and interfaces of the prototype WHMS.

Essence	**Description**
class Activity	Preserves the essence of "Activity" in which there is a field with the name of the activity and a link to a specific category.
class Category	Preserves the essence of "Category". It has a field with the name of the category to which the activities refer.
class UserService	Performs operations on objects with user data User: sorts by name, gives to the database or to the controller.
class ActivityService	Receives from the controller objects of type "Activity", gives them a certain category and writes in a DB. Also through this class data collection from a DB is carried out.
class CategoryService	Gets a list of all categories from the database.
class MainController	Used to issue pages based on user roles. For the administrator - the administrator page, for the user - the user page.
class UserController	Receives get- and post-requests from the client, collects the object, and gives it to the required service for its processing.
interface RestController	Serves so that the usual controller could receive and give JSON objects.
interface CrossOrigin	Used to assemble objects of various formats.
interface RequestMapping	Used to obtain the body of the request (body) and headers.

The PersistenseChoiceService class has a method for retrieving data from the database, namely user requests to add or remove activities, and generate a response in the form of a list for each user. This method performs 3 queries to the database at the same time, in order to retrieve all users who want to add or remove activity, the list of activities, save the entire response to one object, and give the controller to transfer the object to the client.

```java
public List<AllUserPersistenceResponse> getAllUserChoice() {
    List<AllUserPersistenceResponse> listResponse = new ArrayList<>();

    List<PersistenceChoice> persistenceChoices =
persistenceChoiceRepository.findAll();

    userService.getAllUser().forEach(user ->
            listResponse.add(new AllUserPersistenceResponse(user.getUsername(),
new ArrayList<PersistenceResponse>() {{
                    persistenceChoices.stream()
                            .filter(persistenceChoice -> persistenceChoice
                                    .getUser()
                                    .getUsername()
                                    .equals(user.getUsername()))
                            .collect(Collectors.toList()).forEach(pChoice ->
                            add(new PersistenceResponse(pChoice.getId(),
pChoice.getActivity().getName()
                                    , pChoice.getAction())))
                    );
            }}
            ))
    );
    return listResponse;
}
```

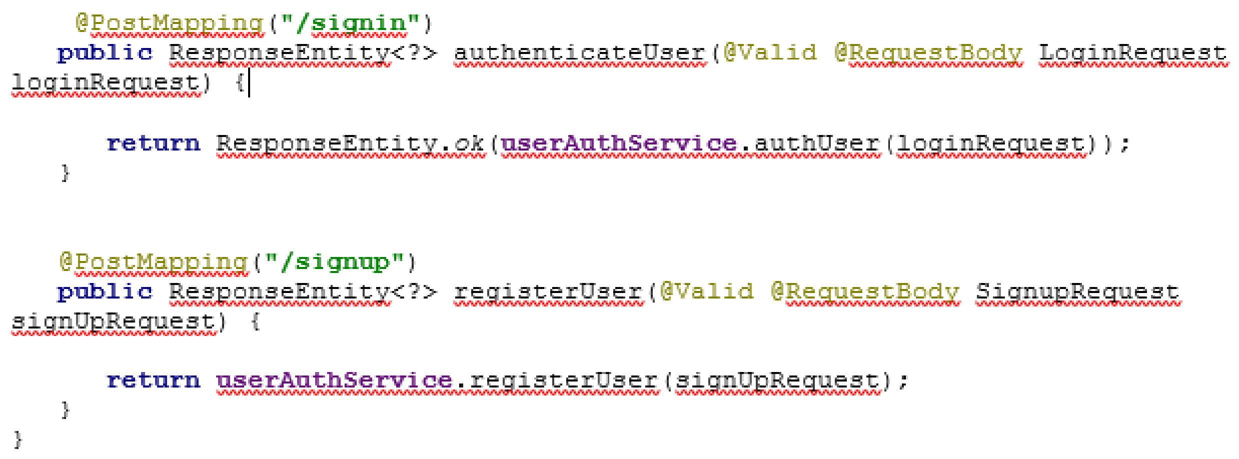
The DailyStatisticService class has a method for saving the amount of time spent on activity in one day. The method generates a request to save certain activities in the database, having previously received the object activityservice and userrepository.

```java
public void save(DailyStatisticResponse dailyStatisticResponse) {
    DailyStatistics dailyStatistics = new DailyStatistics();
    dailyStatistics.setActivity(activityService
            .getActivityByName(dailyStatisticResponse.getNameActivity()));
    dailyStatistics.setTime(dailyStatisticResponse
            .getTime());
    dailyStatistics.setUser_id(userRepository
            .findByUsername(dailyStatisticResponse.getNameUser())
            .orElseThrow(() -> new UsernameNotFoundException("User Not Found
with username: " + dailyStatisticResponse.getNameUser()))
            .getId());
    dailyStatistics.setDate(new Date(new java.util.Date().getTime()));
    dailyStatisticsRepository.save(dailyStatistics);
}
```

In the MainController class the method of receiving the list of all activities from a DB is created. The GET-method which addresses in activityService and gives JSON-object to the user by means of the @GetMapping annotation is presented.

```java
@GetMapping("/getAllAct")
public          List<ActivityResponse>          getAllAct()          {
    return                        activityService.getAllActivity();
}
```

In the UserController class methods of authorization and registration of users are created. Requests from the client come with the @PostMapping annotation. It is used to ensure that the name and password are stored in the hidden body of the request and are not visible in the URL.



```
    @PostMapping("/signin")
   public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest
loginRequest) {

       return ResponseEntity.ok(userAuthService.authUser(loginRequest));
   }


   @PostMapping("/signup")
   public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest
signUpRequest) {

       return userAuthService.registerUser(signUpRequest);
   }
}
```

4.2. The results of testing the prototype WHMS.

During the work on this section, a prototype of the system for monitoring the work processes of the technical department of the Internet provider was created. Next, I present the results of testing the work of the prototype WHMS.

Figure 4.1. shows screenshots of forms of registration and authorization of SMRC.

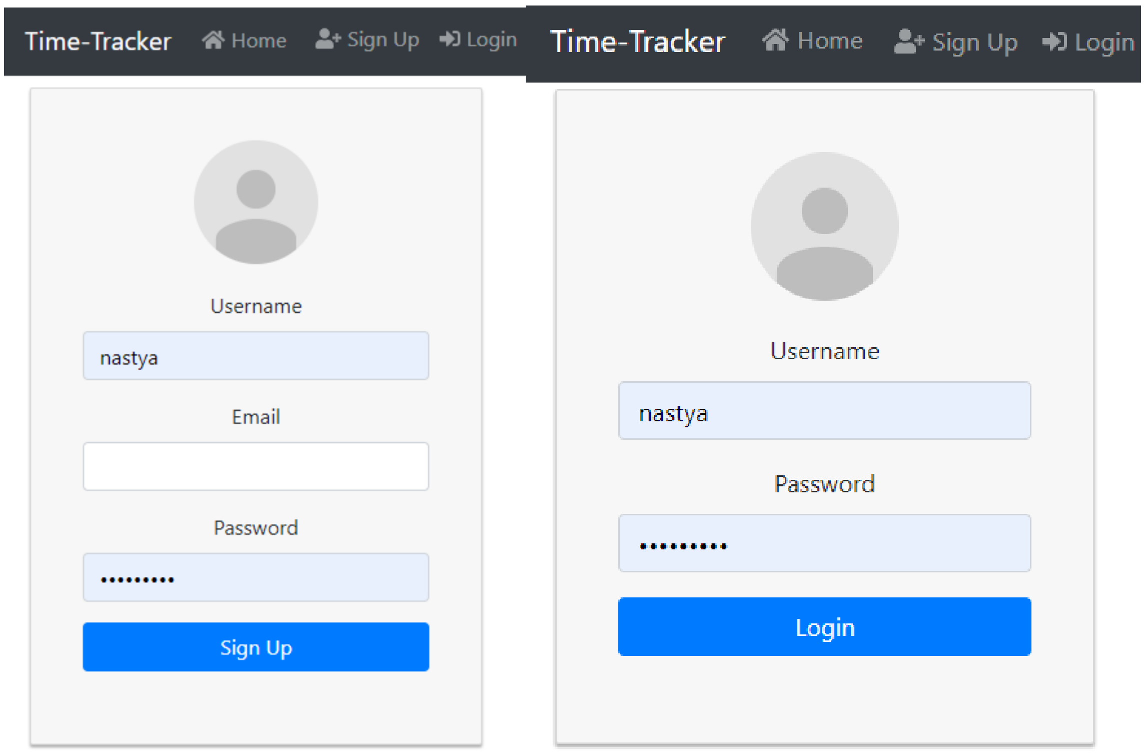
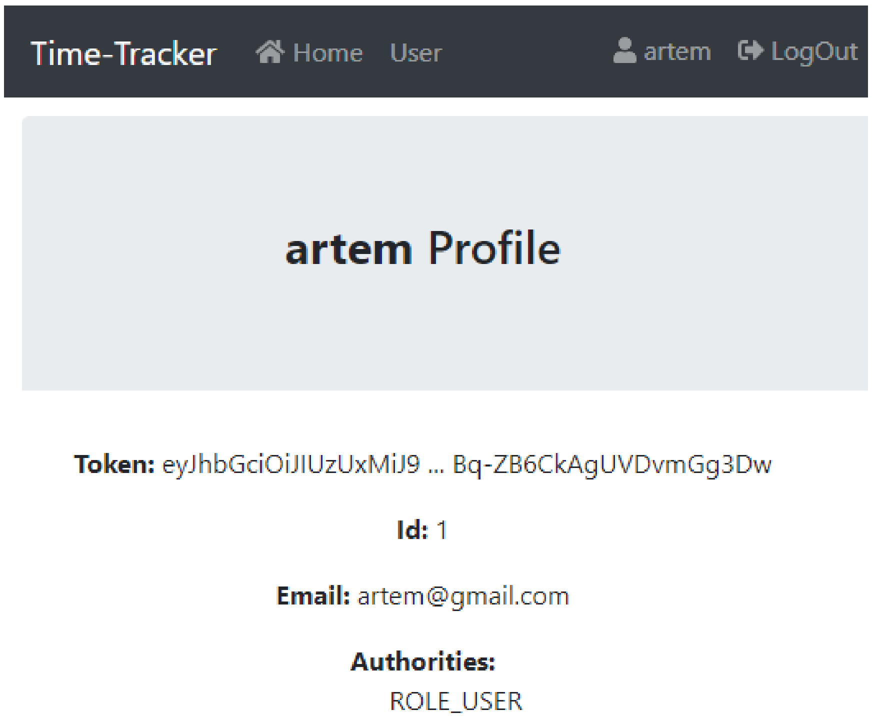


Fig.4.1. Forms of registration and authorization in WHMS

In order to be registered in the system, you need to enter your name, e-mail and password. To log in, the user must enter his login in the Username field and the password in the Password field.

Figure 4.2 shows the user's profile page on the system, which lists his login, id, mail and role.



artem Profile

Token: eyJhbGciOiJIUzUxMiJ9 ... Bq-ZB6CkAgUVDvmGg3Dw

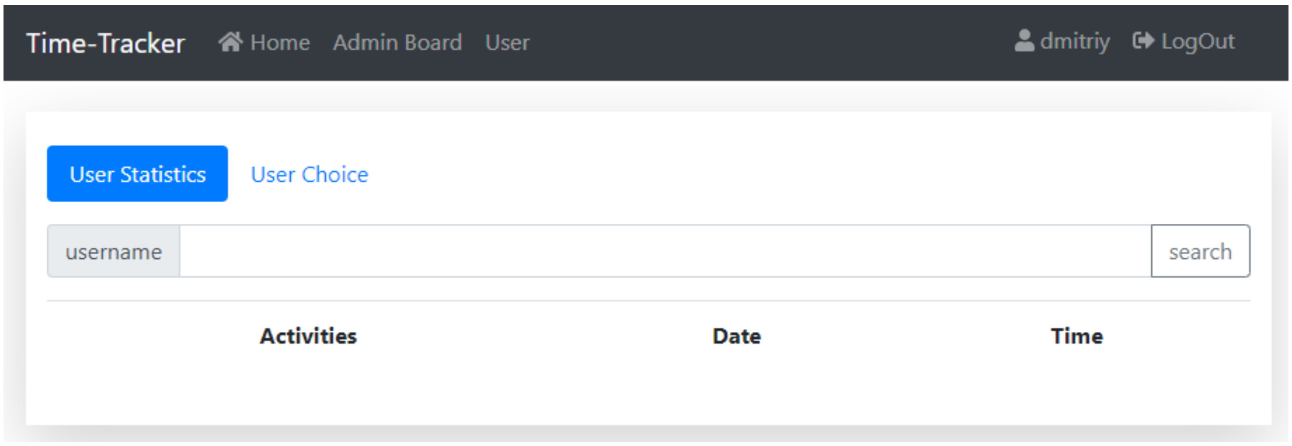
Id: 1

Email: artem@gmail.com

Authorities:

ROLE_USER

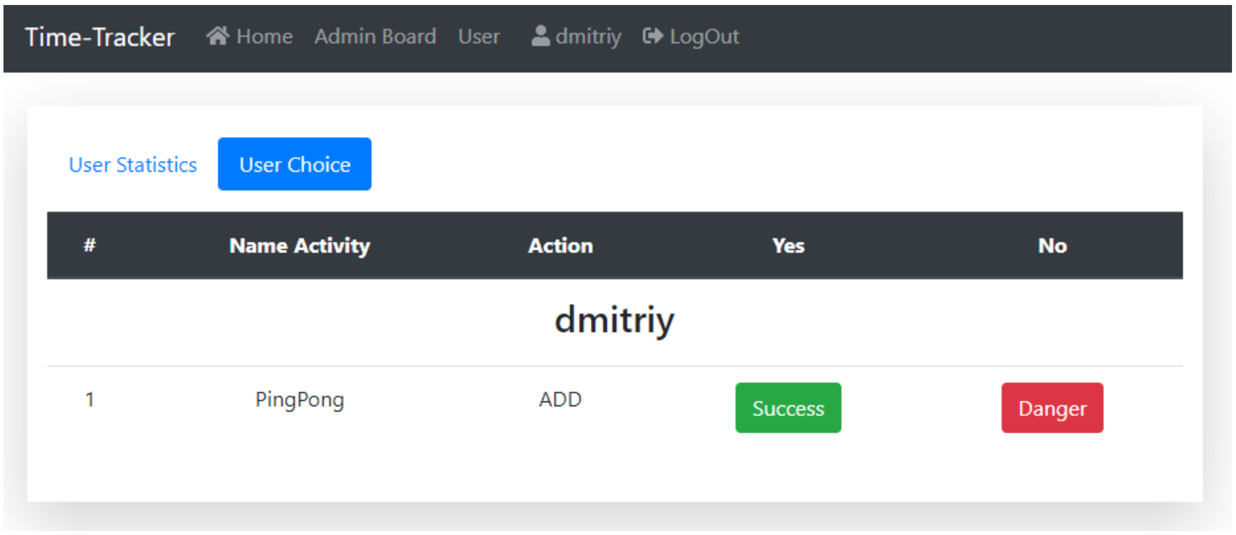
Fig.4.2. WHMS user profile page



Activities	Date	Time

Fig.4.3. Statistics request page for all users

Figure 4.4 shows a page that displays user requests to add or remove activities.



#	Name Activity	Action	Yes	No
dmitriy				
1	PingPong	ADD	Success	Danger

Fig.4.4. User requests page for adding / deleting activities

Figure 4.5 shows user activity. This is a page with all possible activities of the user of the CMS and an individual list of activities. You can request to add or delete a specific activity, as well as display statistics for a specific date.

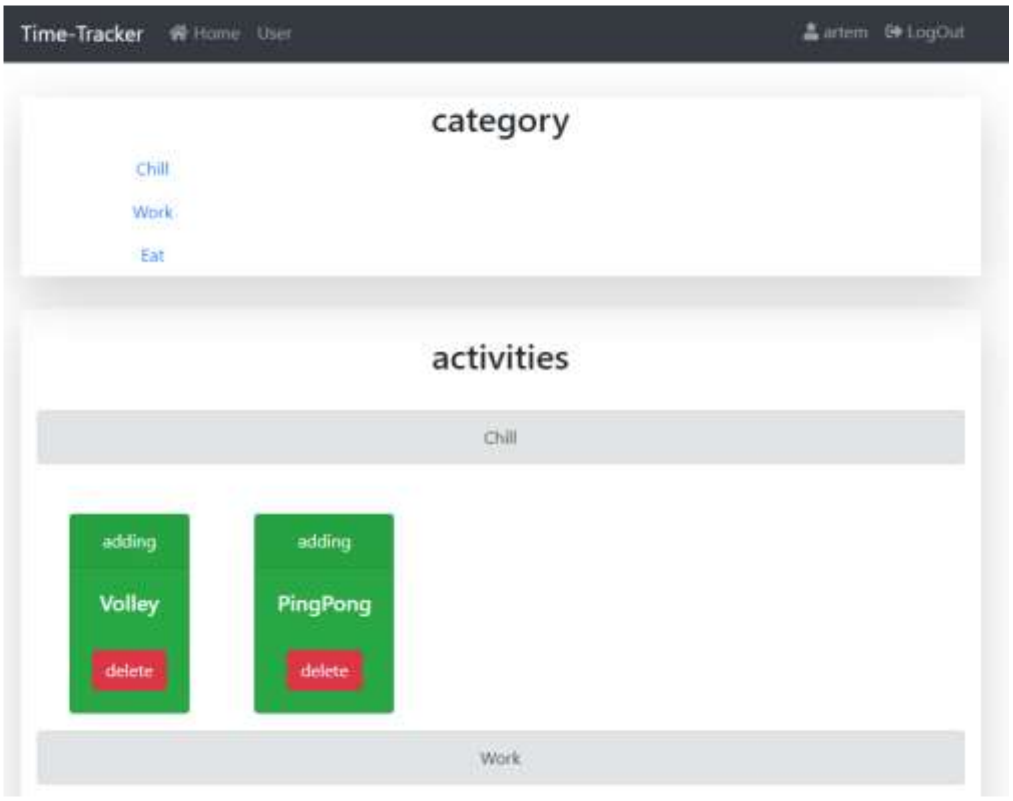
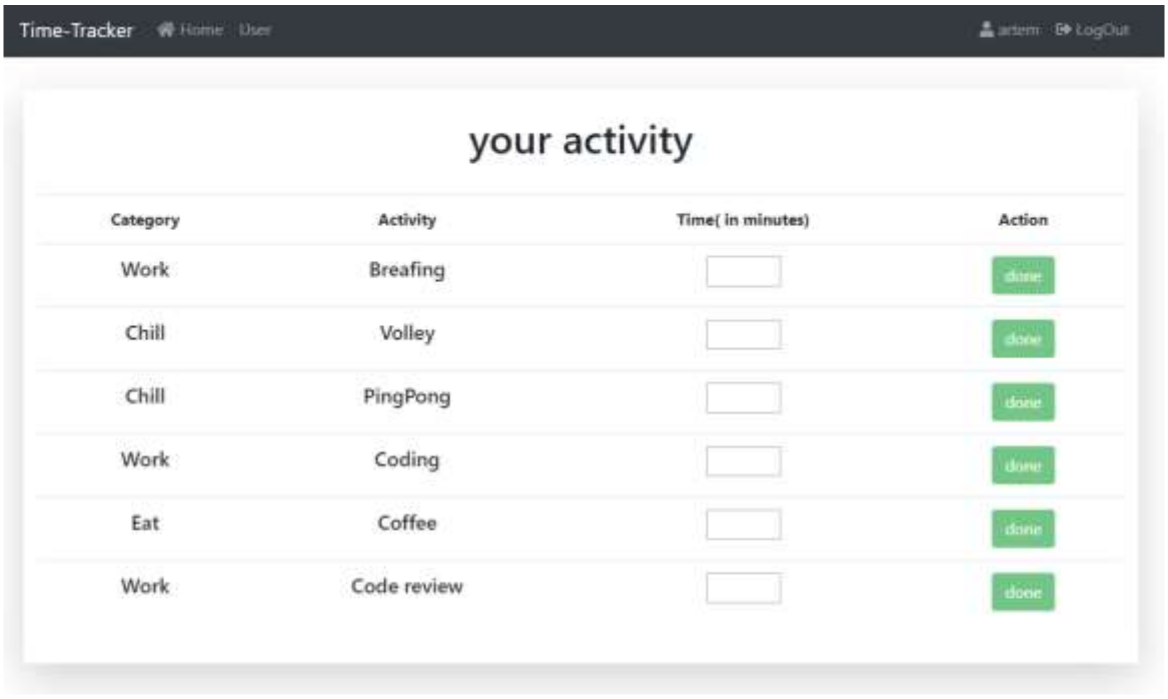


Fig.4.5. Possible user activity page

Figure 4.6 shows the user's daily activities. This is a page to record the time spent on one activity during the day.



Category	Activity	Time(in minutes)	Action
Work	Breafing		done
Chill	Volley		done
Chill	PingPong		done
Work	Coding		done
Eat	Coffee		done
Work	Code review		done

Fig.4.6. User's daily activities page

Conclusions

During the writing of the fourth section of the thesis, I created a prototype of the monitoring system of the technical department of the Internet provider. The prototype was implemented using object-oriented programming environments IntellIJ Idea and Visual Code.

To implement the prototype in Java, it was necessary to create 7 classes: activity and category classes, service class for user data operations, class for retrieving from the activity controller, class for obtaining a list of categories, class for issuing pages by user roles and a class that accepts requests from the client and sends them to the desired service. Three interfaces have also been created: one for receiving and sending JSON objects by the controller, the second for assembling objects of various formats, and the third for receiving the request body and headers.

The created prototype was tested, and the test results in the form of screenshots are given in this section.

CONCLUSION

In this diploma project, I analyzed the methods of controlling the working hours of employees of the technical department of the Internet provider. Both the employees and the manager need control over the work processes of the employees of the technical department of the Internet provider. The analysis revealed a number of problems that make it impossible to fully control the use of working time by workers - untimely return of the employee after a lunch break, periodic absence of the employee at work, downtime, etc.

When working on the diploma project, the functional requirements for each logical block of the system were determined - for registration and authorization of users, for the main page, for the user page, the user profile page and the administrator page. Non-functional system requirements were also identified and described - user interface requirements, software interfaces, performance, reliability, availability, security, traceability and portability.

To prepare for the software implementation of the prototype of the monitoring system of the technical department of the Internet provider, it was determined that the MVC pattern (model-representation-controller) will be used as an architectural template, as well as a diagram of project components, class diagram and web-system deployment diagram. .

The last section demonstrates the process of building a prototype of the monitoring system of the technical department of the Internet provider, as well as the results of testing the system.

REFERENCES

1. Alexey Vasiliev "Programming in Java", K .: Textbook - Bogdan, 2019. - 578p.

2. Robert Martin "Pure Architecture", H .: Fabula, 2019. - 416p.

3. Robert Martin "Pure Code", H .: Fabula, 2019. - 368p.

4. Spring 4 for professionals. Clarence Ho and others. M .: Williams, 2015. - 752p.

5. Herbert Schildt “Java. Complete guide, volume 1 ", M .: Dialectics, 2017. - 730p.

6. Benjamin J. Evans, David Flanagan “Java. Developer's Guide ", M .: Dialectics-Williams, 2019. - 592p.

7. Computer technology and information technology: Textbook. way. Kozlovsky AV and other. H .: Fabula, 2012. - 463p.

8. Goltzman VI «MySQL 5.0. Library of the programmer »[Electronic resource] // Sitforum. - 2019. - P.253 - Access mode https://www.citforum.at.ua/MySQL.pdf

9. Evgeny Morgunov «PostgreSQL. Fundamentals of the SQL language ", St. Petersburg: BHV-Petersburg, 2016. - 336p.

10. Paul Dubois "MySQL", M .: Williams, 2006. - 1168p.

11. Regulations on diploma theses (projects) of graduates of the National Aviation University Kyiv 2017. - 63p.