

Имена: Петьо Христов, Леонардо Петков, Радостин Хандъров

фн: 8MI0800298, 7MI0800174, 1MI0800209

Начална година: 2022

Програма: бакалавър, (КН)

Курс: 4

Тема: Система за организиране на събития

Дата: 2026-01-29

Предмет: w25prj_KN_final

имейл:

petyohristov2003@gmail.com, leonardopetkov32@gmail.com, handarov@uni-sofia.bg

преподавател: доц. д-р Милен Петров

Предаване: Задачата се предава в архив с попълнен настоящия документ, проекта/проектите с кодовете, README.txt файл, който описва съдържанието на архива; папка с допълнителни компоненти и използвани ресурси, **архива да се казва 9999_final.zip**. (Успех!). (Редактирайте **маркираните зони** в жълто с коректната информация)

ТЕМА: w25/38.1 Система за организиране на събития

1. Условие

Да се направи уеб базирана система, посредством която всеки потребител ще може да задава празници (например рожден ден, имен ден). Потребителят може да последва други потребители, както и да бъде последван. Може да се организира анонимно празнуване на определена дата, за което останалите последователи на „виновника“ да могат да се запишат и да се включат в купуване на подарък, направа на картичка или изпращане на музикален поздрав.

2. Въведение – извличане на изисквания

Роли: потребителите в системата;

Функционални изисквания:

1. Потребителите могат да виждат информация за профила си (последователи на профила, пълно име, имейл);
2. Потребителите могат да променят своето пълно име, както и паролата си;
3. Потребителите могат да въвеждат специални дати (имени дни, рождени дни, годишнина и т. н.);
4. Потребителите могат да последват/отследват други потребители;
5. Потребителите могат да организират събитие (празненство) на друг потребител, който следват;
6. Потребителите могат да виждат детайли за определено събитие, стига да следват „виновника“ на събитието;
7. Потребителите могат да се записват за/отписват от събития на друг потребител, който следват;
8. Ако потребител се е записал за дадено събитие, той може да добавя подаръци, които ще донесе на празненството;
9. Потребителите могат да се регистрират и да влизат в системата;

Нефункционални изисквания:

1. Системата е лесно преносима (Portable);
2. Системата е реализирана чрез MVC (Model-View-Controller) архитектурата, което позволява лесно модифицируемост (Modifiability);
3. Системата следва основни мерки за сигурност (backend-а е „скрит“ от сървъра, паролите се хешират) (Security);
4. Потребителският интерфейс на системата е интуитивен и потребителят лесно може да се ориентира в него (Usability);
5. Потребителят може лесно да свърши задачите, за които е влязъл в системата, без нужда от допълнителна помощ (Usability);

3. Теория – анализ и проектиране на решението

Системата е изградена като монолит. Използван е архитектурният стил MVC (Model-View-Controller), за да се раздели потребителският интерфейс от бизнес логиката. Така двете части могат да бъдат променяни по-лесно и независимо една от друга. Сега ще обясним слоевете, като започнем от най-ниския:

1. **Model** – Тези класове представляват обектите в нашата система. Те представят таблицата от базата данни в нашата система. Например обектът Потребител е един ред от таблицата Потребители в базата данни.
2. **Repository** – Тези класове си „говорят“ директно с базата данни, като използват моделите (models) в интерфейса на методите си. В проекта има UserRepository и EventRepository.
3. **Service** – Тези класове съдържат същинската бизнес логика на backend-а. Те обработват входната информация, използват Repository, за да извличат/променят информация от базата данни, и връщат подходящ резултат. В проекта има UserService (отговаря за потребителите и техните специални дати) и EventService (отговаря за събитията и всичко, свързано с тях – гости, подаръци и т. н.).
4. **Controller** – Тези класове обработват заявките. Взимат параметрите от заявката, извикват подходящите services и предават информацията на view-тата.
5. **View** – Това са include-ите в края на методите на controllers. Така „съшиваме“ различни шаблони (header, footer) с основната част от страницата. Те се намират в папка frontend.

Помощни класове, без които проектът няма да работи:

1. **AppConfig** – Този клас работи с конфигурацията на приложението. В нея са заложили детайлите за връзка с базата данни.
2. **Loader** – Неговата задача е да включва автоматично новоизползвани класове в изпълнението на скрипта.
3. **DBPool** – Той е сингълтън (singleton) клас, който играе ролята на хранилище на връзки към базата данни. Ако такива няма, той прави нова връзка, като взема информацията от AppConfig класа.
4. **Router** – Той обработва URI-а на заявката, извлича параметрите, създава services, създава подходящия контролер, инжектира му необходимите services и извиква подходящия метод, който ще се погрижи за заявката.
5. **SessionManager** – Той отговаря за управлението на потребителската сесия. По този начин можем да пазим състояние между HTTP заявките и да поддържаме автентикация.

4. Използвани технологии

Приложението може да работи на всяка операционна система, стига да има инсталиран уеб сървър с PHP интерпретатор на него. Използваните технологии са HTML, CSS и JS (за frontend-а) и PHP (за backend-а и части от frontend-а). Не са използвани външни библиотеки и фреймуърци (frameworks) за реализацията на проекта. Приложението може да работи с различни видове бази данни. Средата, на която е разработено и тествано приложението е Windows 11 с локално инсталиран XAMPP. Базата данни е MySQL.

5. Инсталация, настройки и DevOps

За инсталация на приложението е необходимо да бъдат изпълнени следните стъпки:

1. Инсталиране на уеб сървър с PHP интерпретатор и база данни. Например XAMPP.



2. Създаване на база данни за приложението.

БД

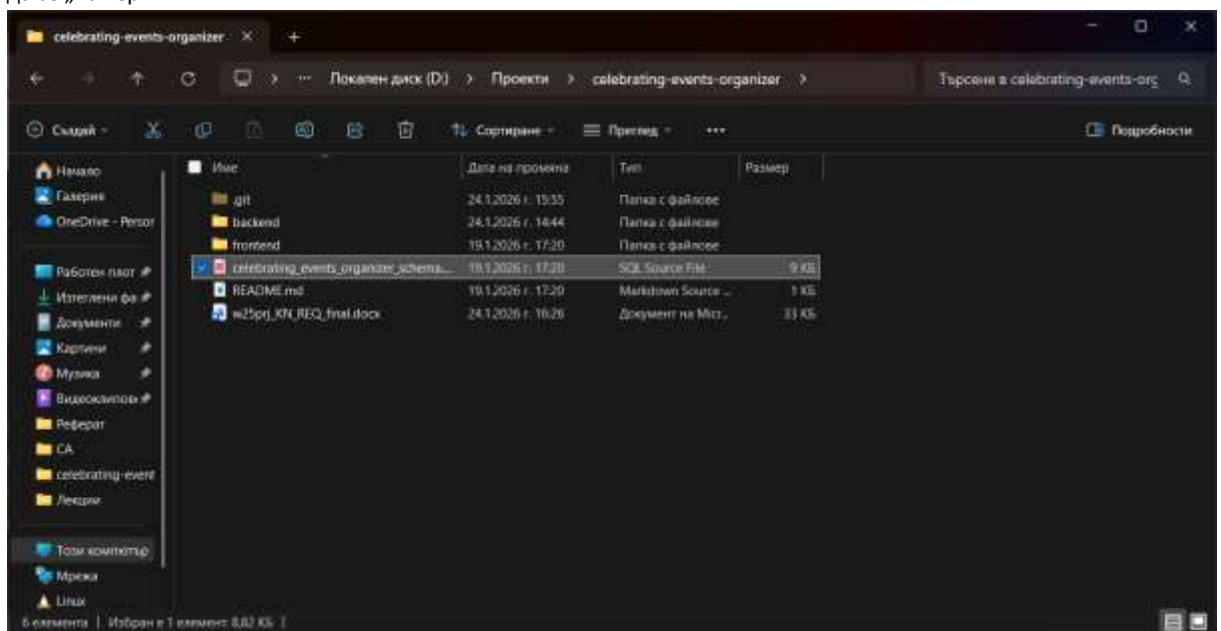
Създаване БД

Създаване

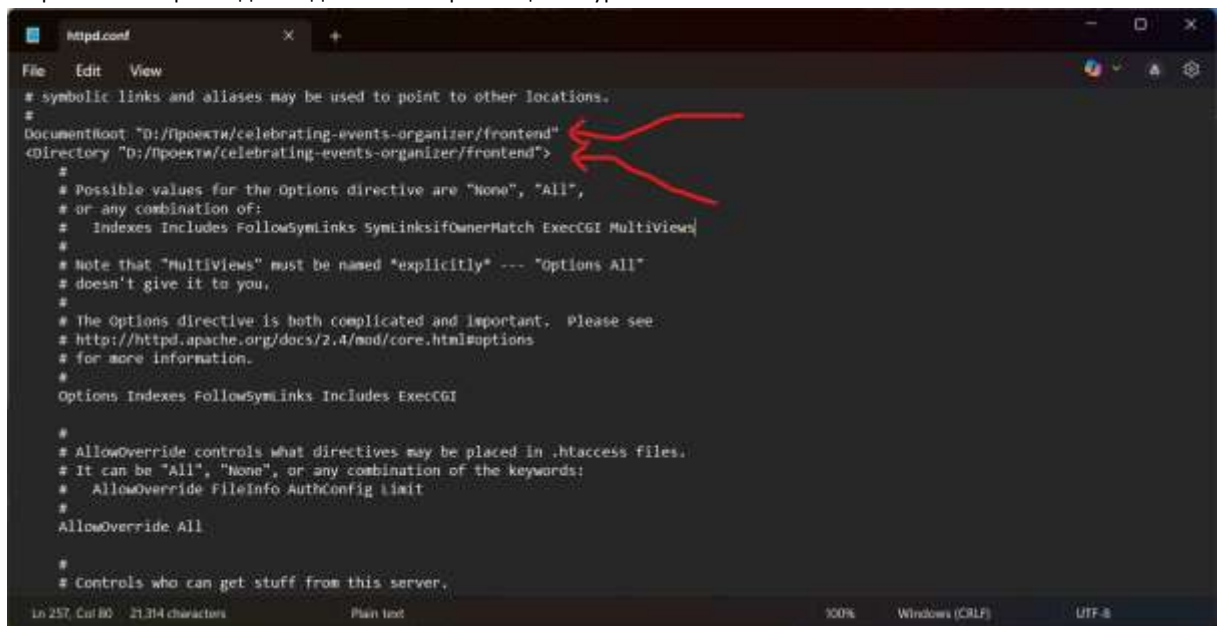
3. След това се изпълнява SQL скрипта от файла celebrating_events_organizer_schema.sql. Той ще създаде всички таблици, колони, индекси и примерни данни.

✓	celebrating_events_organizer_schema....	19.1.2026 г. 17:20	SQL Source File	9 КБ
---	---	--------------------	-----------------	------

4. Слагаме проекта в една папка. Важно е всичко да е на едно място, защото frontend-а и backend-а не могат да се „намерят“.

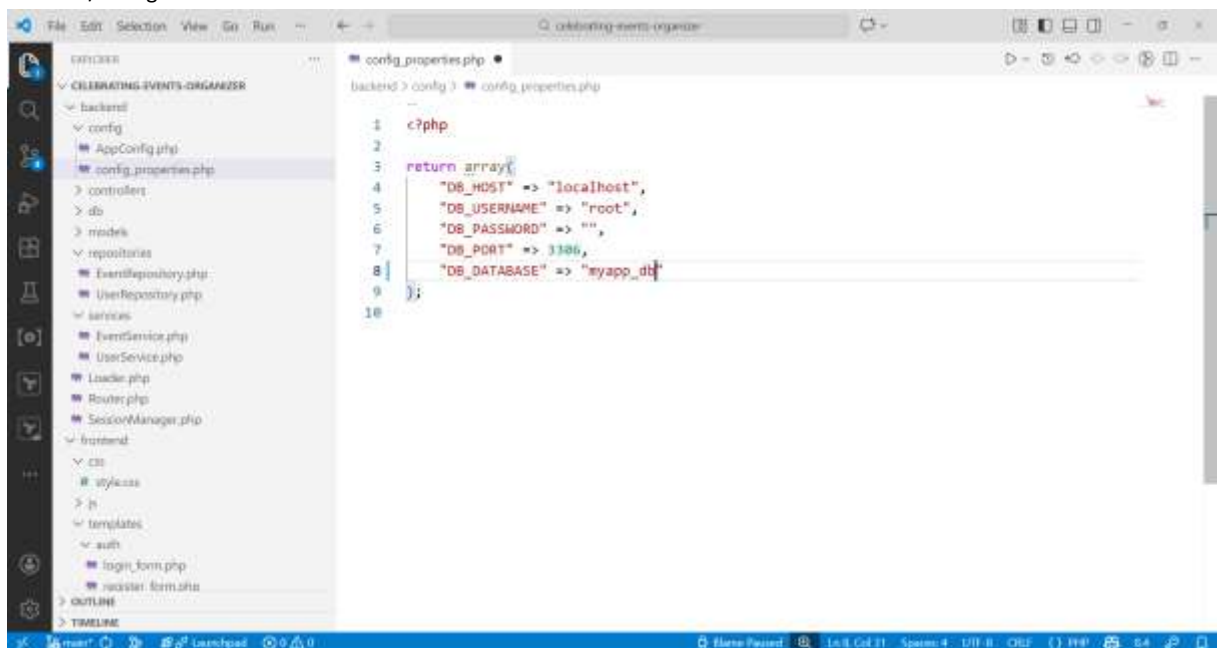


5. Настройваме DocumentRoot-а на уеб сървъра да сочи към папката frontend, защото там са ресурсите, които потребителите трябва да виждат. Това се прави с цел сигурност.



```
File Edit View
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "D:/Проекти/celebrating-events-organizer/frontend"
<Directory "D:/Проекти/celebrating-events-organizer/frontend">
#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes FollowSymLinks SymLinksIfOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named "explicitly" --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
Options Indexes FollowSymLinks Includes ExecCGI
#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   AllowOverride FileInfo AuthConfig Limit
#
AllowOverride All
#
# Controls who can get stuff from this server.
```

6. Въвеждат се данните за връзка с базата данни във файла config_properties.php, намиращ се в папките backend/config.

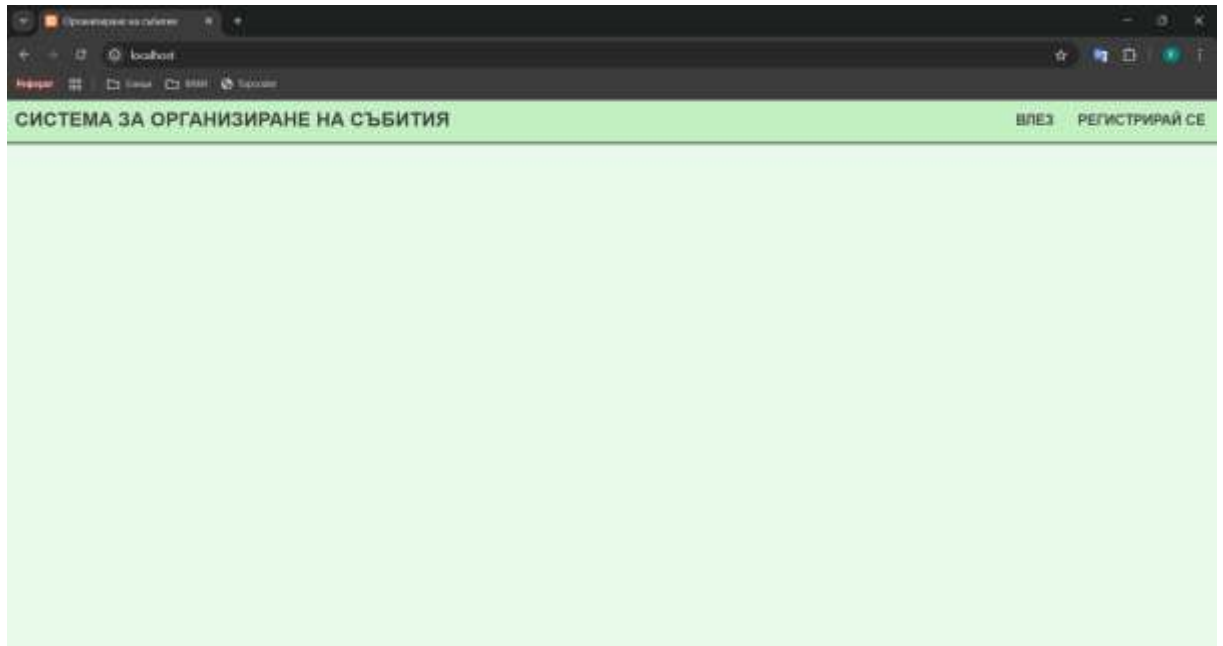


```
File Edit Selection View Go Run ...
celebrating-events-organizer

config_properties.php
backend > config > config_properties.php

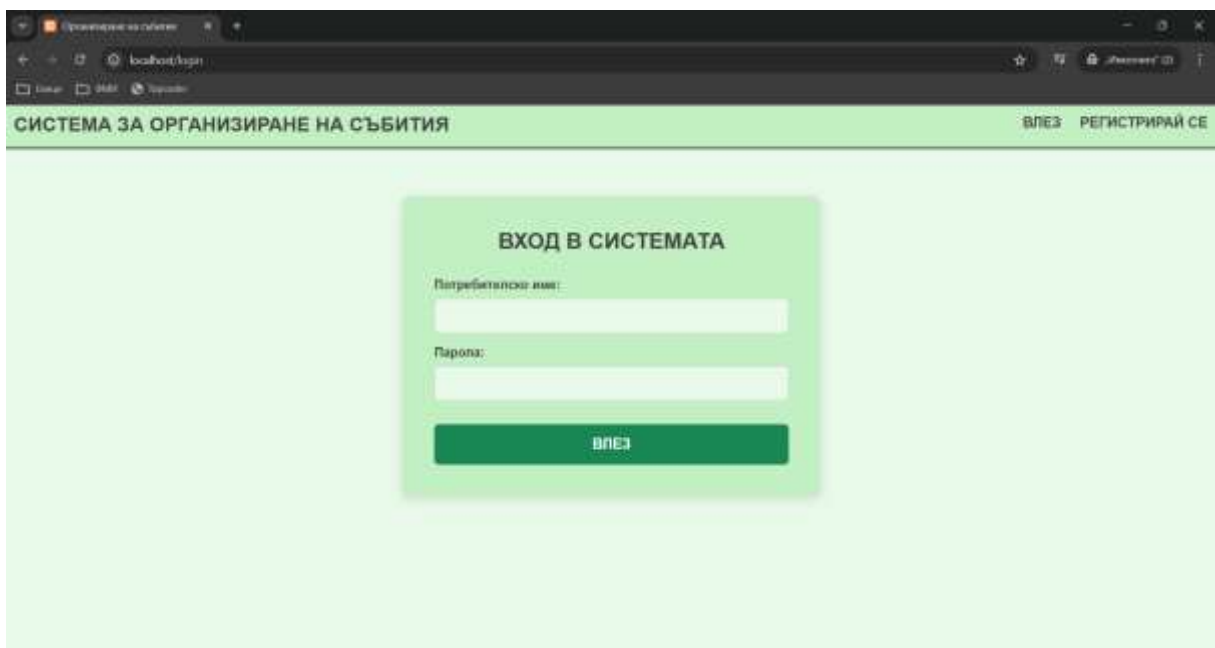
1 <?php
2
3 return array(
4     "DB_HOST" => "localhost",
5     "DB_USERNAME" => "root",
6     "DB_PASSWORD" => "",
7     "DB_PORT" => 3306,
8     "DB_DATABASE" => "myapp_db"
9 );
10
```

7. Ако всичко е наред, трябва да се появи началната страница на приложението.

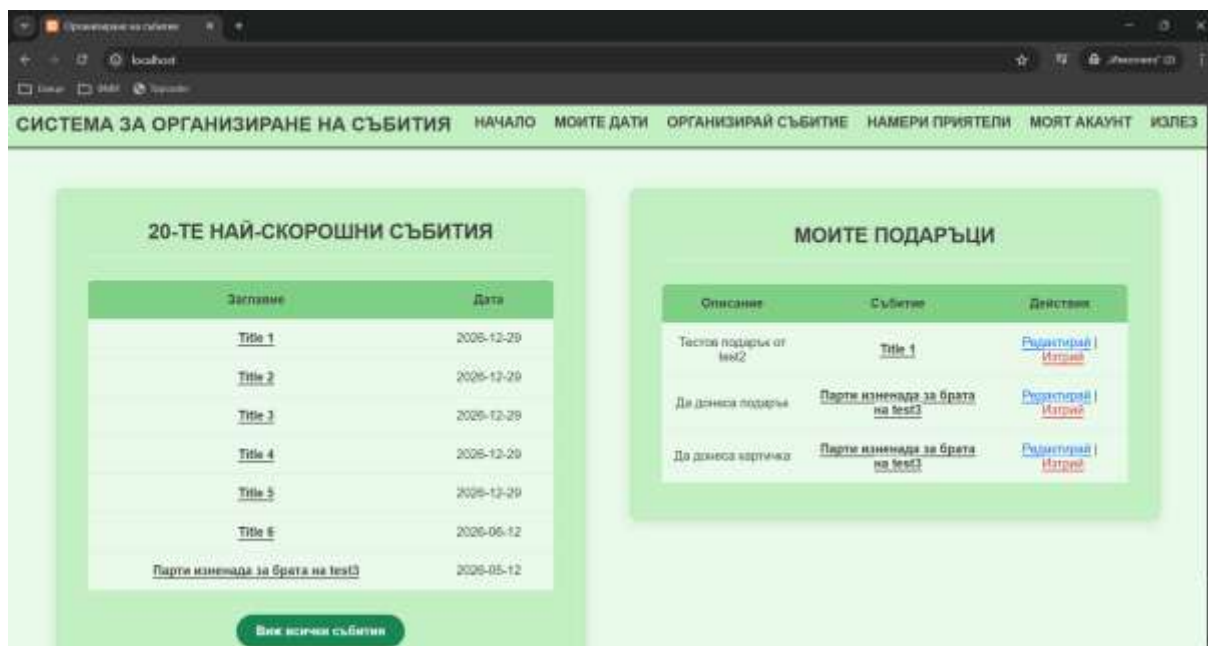


6. Кратко ръководство на потребителя

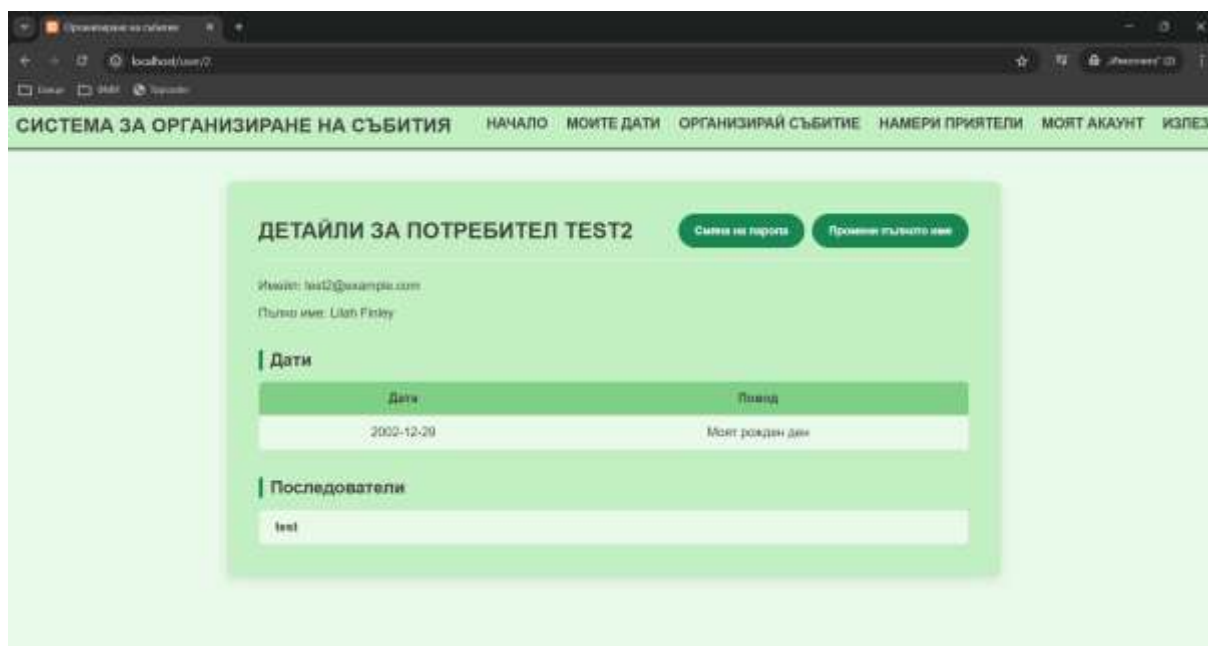
За да може потребителят да работи със системата, той трябва да влезе в нея. Ако няма профил, първо трябва да се регистрира.



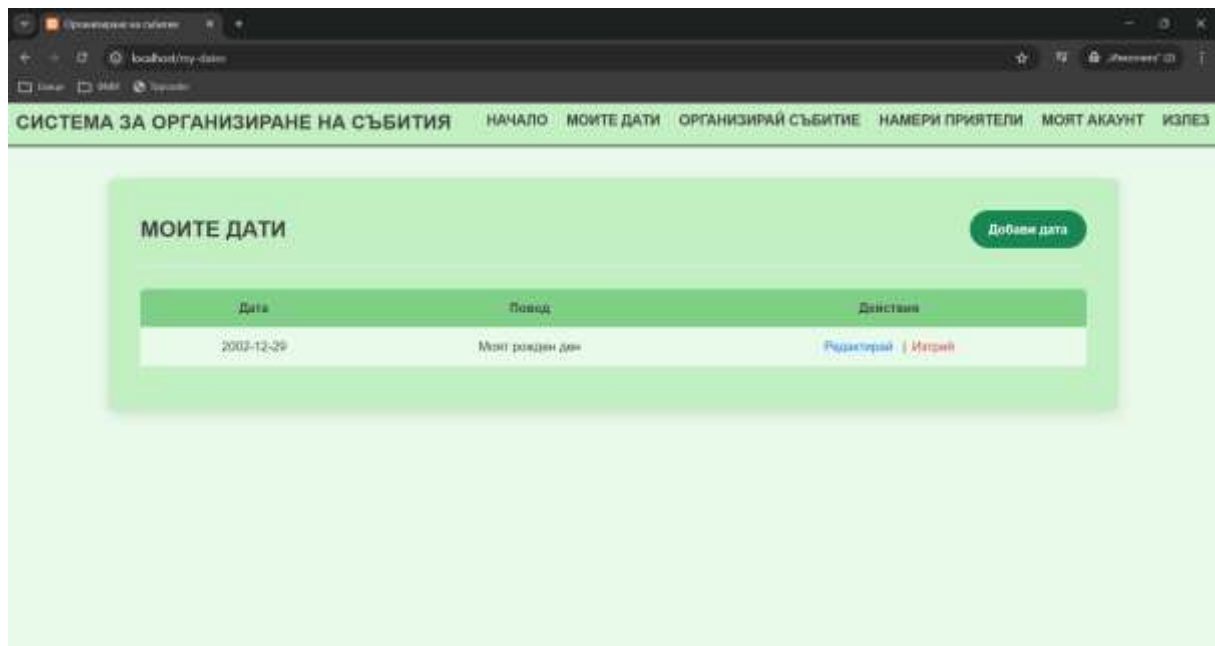
След като влезе, потребителят ще вижда 20-те най-скорошни събития, в които може да се включи, и всички подаръци, които ще донесе в бъдещето (така няма как да забрави някой подарък).



Натискайки на линка „МОЯТ АКАУНТ“, потребителят може да види информация за профила си, както и да смени пълното си име или паролата.

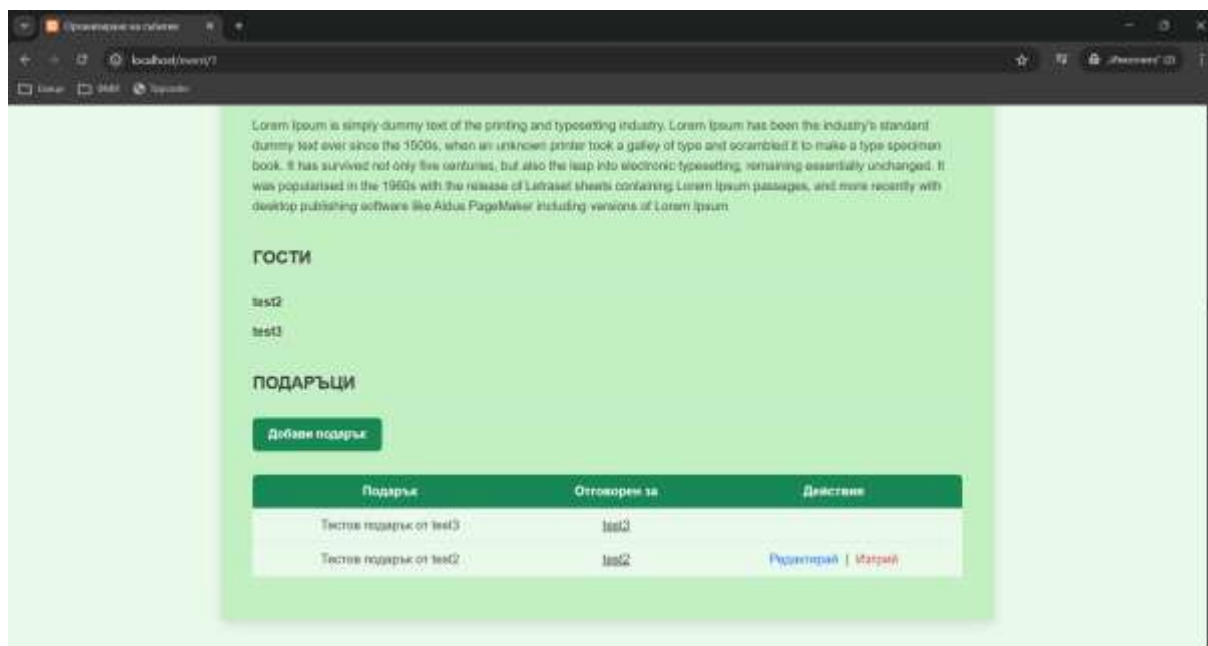


Потребителят може да види, промени, изтрие или въведе своите специални дати, като натисне линка „МОИТЕ ДАТИ“.



Натискайки върху заглавието на дадено събитие, системата показва детайлна информация за него.





Тук потребителят може да се запише/отпише от събитието, може да добавя своите подаръци, както и да научи повече детайли за събитието.

7. Примерни данни

Системата има няколко тестови акаунта с потребителския имена – test, test2, test3, test4, test5 и test6 и пароли – password, password2, password3, password3, password1234, password6 (не правете така на практика!). Първите 3 акаунта имат тестови дати, на които е станал някакъв повод, има тестови събития, гости и подаръци. Останалите акаунти просто са създадени, докато приложението е тествано.

8. Описание на програмния код

Ще опишем файловата структура на проекта, както и по-важните моменти от кода. Проектът е разделен в две папки:

1. backend – тук е цялата бизнес логика на приложението;
2. frontend – тук е потребителският интерфейс на приложението;

Папката backend се дели на следните подпапки:

1. config – Тук се намират конфигурационният файл (config_properties.php) и класът AppConfig (за по-лесна работа с конфигурационния файл).
2. controllers – Тук се намират всички контролери, които обслужват заявките. Сървисите се инжектират през конструктура на всеки контролер от рутера. Всеки метод на котролерите приема параметър *array \$params*, в който има всички параметри от URI-а на заявката. Масивът се предава от рутера.
3. db – Тук се намира класа DbPool – хранилището, от което останалите компоненти взимат връзки към базата данни.
4. models – Тук се намират всички модели. Те са обикновени класове, представящи таблиците в базата данни. Имат конструктор с параметри – всички колони на таблицата, гетъри и сетъри. При сетърите се прави филтриране и обработка на параметрите с цел сигурност и поради начина, по който работят сървисите.
5. repositories – Тук са всички репозитори, които си „говорят“ директно с базата данни. Връзката към базата данни се инжектира от сървиса, който ползва това репозитори.

Други файлове в папката backend:

1. Loader.php – Тук се намира loader-ът, който намира нов клас във файловата система и го включва в изпълнението на скрипта.
2. SessionManager.php – Тук става управлението на сесията на потребителя, като се използват функции от стандартната библиотека на PHP като *session_start()*, *session_end()*. За работа с параметрите на сесията се

използва глобалната променлива `$_SESSION`. В сесията се пази информация дали потребител е влязъл в системата и ако е влязъл – `id`-то на акаунта.

3. Router.php – Тук е логиката, свързана с рутирането на заявките към правилните контролери и методи, управление на сървисите и инжектирането им в контролера. URI-а на заявката, както и HTTP методът (в нашия случай – GET или POST) се извличат от глобалната променлива `$_SERVER` и след това се анализират, за да се рутира заявката към правилното място.

В папката frontend:

1. Index.php – Това е стартовата точка на приложението. Той регистрира loader-а, инициализира сесията, рутера и го пуска да обработва заявката.
2. .htaccess – В него има настройки всички адреси (например localhost/users/) да отиват към index.php.
3. header.php и footer.php – Хедъра и футъра на страницата.
4. home-page-logged.php – Тялото на началната страница, когато потребителят влезе в акаунта си.

В останалите подпапки на frontend се намират телата на останалите страници в приложението. Файловете са разделени в подпапки с цел по-добра организация. Динамичната информация се извлича от локалните променливи на методите на контролерите, тъй като те включват тези файлове в телата на методи си като отговор на заявката.

9. Приноси на студента, ограничения и възможности за бъдещо разширение

1. Петьо Христов – Разработих потребителския интерфейс на приложението, като използвах средствата на JavaScript за красивата валидация на данните. Също така говорих с колегите от екипа, за да мога да интегрирам успешно интерфейса с backend-а на приложението.
2. Леонардо Петков – Аз разработих логиката за автентикация, управление на акаунтите, поводите и (от)следване на потребителите в backend-а. Също така се погрижих и за управлението на потребителската сесия. Говорих си с колегите, за да мога безпрепятствено да интегрирам моите функционалности от backend-а с потребителския интерфейс и останалата част от backend-а.
3. Радостин Хандъров – Аз разработих цялата логика, свързана със събитията (организиране, гости, подаръци) в backend-а. Също така реализирах DbPool-а и Router-а. Занимавах се с конфигурирането на уеб сървър и .htaccess файла. Говорих усърдно с колегите, за да може процесът по интегриране на моите неща да бъде доколкото е възможно гладък.

Не сме се ограничавали само до нещата, описани по-горе. Ако някой има предложение за подобрене по работата на друг колега, мнението му е зачитано. Git ни помогна да правим промените и интеграцията много по-гъвкаво и лесно. Целият екип все участие в писането на тази документация.

10. Използване на AI – как и защо

Екипът използва изкуствения интелект на Gemini и ChatGPT за разработката на този проект. Интелектът не замества творческия процес и взимането на решенията за проекта. Той бе използван за вдъхновение, търсене на добри идеи, по-бързо търсене на източници в мрежата и помощ при дебъгване на кода с цел по-бързо откриване на проблема. При генерирането на информацията екипът я проверява за достоверност, т. е. въпреки употребата на изкуствен интелект, ние поемаме цялата отговорност за крайния резултат. По този начин езиковите модели на Google и OpenAI значително повишиха продуктивността на екипа в разработката на проекта.

11. Какво научих (най-важните неща, които сте научили по време на курса и при разработването на проекта-за всеки студент)

1. Петьо Христов - Проектът ми даде възможност да задълбоча познанията си в изграждането на цялостен потребителски интерфейс и интеграцията му със сървърната логика. Научих се как да създавам адаптивни дизайни и да подобрявам потребителското изживяване чрез модерни CSS техники и JavaScript валидация. Също така разбрах по-добре как да управлявам релации в базата данни при специфични функционалности като социални взаимодействия и управление на събития.
2. Леонардо Петков - Проектът ми помогна да придобия опит в разработването на програма в по-голям мащаб. Научих повече за шаблонен дизайн на софтуер, както и за интеграция на база данни в самия код, разработване на уеб страници и работа с git.

3. Радостин Хандъров – Разработването на този проект ми даде по-добро разбиране за това как работи WEB и MVC модела, който е широко използван в много съвременни фреймуърци (frameworks) и технологии. Освен това подобрих уменията си за работа в екип и уменията си за работа с git.

12. Dev(sec)Ops – подкарване на проекта - особености

Проектът заедно с тази документация са качени в Moodle курса като ZIP архив. Стъпките за инсталация са описани в т. 5 (Инсталация, настройки и DevOps).

13. Използвани източници

Материали от лекциите и упражненията в курса, PHP документацията, W3 Schools, StackOverflow и други ресурси в интернет.

- [1] Лекционни материали от курса „WEB технологии“, курс 2025-26, автор доц. д-р Милен Петров, ФМИ, СУ „Св. Климент Охридски“.
- [2] PHP Group, "PHP: Hypertext Preprocessor Documentation", официална документация на езика, разгледани секции за сесии, Prepared Statements, работа с масиви и функции от стандартната библиотека, [<https://www.php.net/docs.php>].
- [3] MDN Web Docs, "HTML, CSS and JavaScript documentation", предоставена от Mozilla, използвана за референции относно DOM манипулация и CSS Layouts, [<https://developer.mozilla.org/en-US/>].
- [4] Oracle Corporation, "MySQL 8.0 Reference Manual", официална документация за работа с релационни бази данни и SQL синтаксис, [<https://dev.mysql.com/doc/refman/8.0/en/>].
- [5] W3Schools, "PHP MySQL Database Tutorial", практически примери за интеграция между PHP и MySQL, [https://www.w3schools.com/php/php_mysql_intro.asp].
- [6] Stack Overflow, "How to MVC in php without the use of some framework", дискусия и примери за структура на контролери и модели, вдъхновение за имплементацията на MVC в проекта, [<https://stackoverflow.com/questions/7308361/how-to-mvc-in-php-without-the-use-of-some-framework>].
- [7] Apache Software Foundation, "Apache HTTP Server Tutorial: .htaccess files", ръководство за конфигуриране на URL пренаписване (URL rewriting), [<https://httpd.apache.org/docs/2.4/howto/htaccess.html>].

Предал (подпис):

/ 8MI0800298, Петьо Христов, КН, 7 гр./

Предал (подпис):

/ 7MI0800174, Леонардо Петков, КН, 7 гр./

Предал (подпис):

/ 1MI0800209, Радостин Хандъров, КН, 7 гр./

Приел (подпис):

/проф. д-р Милен Петров/