РІВНЕНСЬКИЙ ДЕРЖАВНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет математики та інформатики

Кафедра інформатики та прикладної математики

«До захисту допущено»

Завідувач кафедри

Батишкіна Ю. В.

(підпис)

« » 2017 р.

**Дипломний проект (робота)**

ступеня «**Магістр**»

з напряму підготовки (спеціальності) 6.040302 – інформатика

на тему: Мультикористувацька автоматизована програмна система

проведення інтернет‐олімпіад з програмування

Виконав: студент VI курсу, групи М-І-41

Єремейчук Владислав Анатолійович

(прізвище, ім’я, по батькові) (підпис)

Керівник доц., к.т.н., Пригорницький Д. О.

(посада, наукова ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доц., к.т.н., Сафоник А.П.

(посада, науковий ступінь, вчене звання, прізвище ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_\_\_\_\_\_\_

(підпис)

Рівне – 2016 року

**ЗМІСТ**

[ВСТУП 3](#_Toc497058570)

[РОЗДІЛ 1. МОДЕЛЬ ФУНКЦІОНУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ПРОВЕДЕННЯ ОЛІМПІАД З ІНФОРМАТИКИ 6](#_Toc497058571)

[РОЗДІЛ 2. ЗАГАЛЬНІ ВІДОМОВСТІ ПРО ВИКОРИСТАНІ ТЕХНОЛОГІЇ 9](#_Toc497058572)

[2.1 Веб сервери 9](#_Toc497058573)

[2.2 Мова програмування PHP 11](#_Toc497058574)

[2.3 Реляційна БД MySQL 12](#_Toc497058575)

[2.4 Шаблонізатор Twig 14](#_Toc497058576)

[РОЗДІЛ 3. СЕРВЕРНА ЧАСТИНА СИСТЕМИ. СЕРВЕР ПРОВЕДЕННЯ ОЛІМПІАДИ 17](#_Toc497058577)

[3.1 Особливості роботи 17](#_Toc497058578)

[3.2 База даних 18](#_Toc497058579)

[3.3 Доступ до системи 19](#_Toc497058580)

[3.4 Системний таймер 20](#_Toc497058581)

[3.5 Форма завантаження рішень 20](#_Toc497058582)

[3.6 Черга рішень 20](#_Toc497058583)

[3.7 Форма реєстрації 21](#_Toc497058584)

[РОЗДІЛ 4. МОДУЛЬ ПЕРЕВІРКИ ЗАВДАНЬ УЧАСНИКІВ 23](#_Toc497058585)

[4.1 Особливості роботи 23](#_Toc497058586)

[4.2 З’єднання з сервером системи 24](#_Toc497058587)

[4.3 Клас Task 25](#_Toc497058588)

[4.4 Доступ до інформації на сервері 25](#_Toc497058589)

[4.5 Класи Process, Potok. Передача даних для дочірніх процесів 26](#_Toc497058590)

[4.6 Розподіл роботи. Компіляція та перевірка рішень 27](#_Toc497058591)

[РОЗДІЛ 5. КЛІЄНТСЬКА ЧАСТИНА СИСТЕМИ. ІНТЕРФЕЙС КЛІЄНТА. ВИКОРИСТАННЯ СИСТЕМИ 29](#_Toc497058592)

[5.1 Головна сторінка. Форма авторизації 29](#_Toc497058593)

[5.2 Реєстрація в системі. Відновлення втраченого паролю 29](#_Toc497058594)

[5.3 Персональна сторінка користувача 30](#_Toc497058595)

[5.4 Відправка рішень 31](#_Toc497058596)

[5.5 Адміністрування 32](#_Toc497058597)

[ВИСНОВКИ 33](#_Toc497058598)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 34](#_Toc497058599)

# ВСТУП

Щороку в Україні проводяться велика кількість олімпіад, турнірів, конкурсів з навчальних дисциплін серед учнів шкіл та студентів. Такі змагання покликані реалізувати здібності талановитої молоді, популяризувати досягнення науки, техніки та новітніх технологій. Серед цих конкурсів можна виділити особливу групу змагань, які передбачають вирішення задач алгоритмічного характеру, ­­­– це олімпіади з інформатики та програмування. Кращі учасники таких олімпіад у майбутньому будуть не лише активними користувачами різного роду комп’ютерного обладнання і контенту, а й самі займатимуться розробкою новітнього програмного забезпечення, будуть генераторами нових ідей в галузі комп’ютерної інженерії. Для олімпіад з інформатики та програмування потрібно мати ефективні, надійні засоби, що забезпечуватимуть як підтримку самих олімпіад, так і об'єктивне оцінювання результатів.

Актуальність роботи пов’язана з необхідністю вирішення проблеми проведення олімпіад та конкурсів з навчальних дисциплін серед студентів та учнів, для виявлення найкращих серед них та залучення учнівської молоді до позакласної та наукової роботи. Проведення таких олімпіад здійснюється згідно з положеннями Міністерства освіти та науки України. Для оптимізації підготовчого та організаційного процесу розробляються відповідні рекомендації. Зокрема, відповідно до наказу Міністерства освіти і науки, молоді та спорту України №1099 від 22.09.2011 та листа Інституту інноваційних технологій і змісту освіти №14.1/10-3885 від 08.12.2014 стосовно методичних рекомендацій по проведенню ІІІ етапу та підготовки до ІV етапу Всеукраїнської олімпіади з інформатики у 2014-2015 навчальному році, пропонувалось проводити такі олімпіади за власними завданнями із використанням власної автоматизованої системи приймання, шифрування та перевірки робіт. Такий підхід дозволяє проводити олімпіади з так званого «спортивного програмування», що передбачає розв’язання завдань на швидкість. При цьому забезпечується об’єктивність перевірки результатів, а методи заміру процесорного часу виконання програми та обсягів використаної оперативної пам’яті дозволяють встановити не тільки правильність розв’язку, а й ефективність розробленого учасником алгоритму та його програмної реалізації.

**Метою дипломної роботи** є розробка універсальної багатокористувацької автоматизованої програмної системи проведення олімпіад з програмування у глобальних та локальних обчислювальних мережах.

У процесі реалізації сформованої мети необхідно вирішити наступні завдання:

1. підтримка процесу реєстрації учасників та адміністраторів змагань;
2. моніторинг результатів змагань;
3. проведення збору розв’язків завдань;
4. повна автоматизація процесу перевірки робіт учасників;
5. забезпечення можливості дистанційного проведення олімпіад та конкурсів в областях та по Україні на місцях, без необхідності виїзду учасників до фізичного місця проведення змагання.

**Об’єктом дослідження** є сучасні технології розробки web-сайтів з використанням серверу та бази даних.

**Предметом дослідження** є процес проектування і розробки веб-сервісу підтримки олімпіад та конкурсів з інформатики, що передбачають розв’язання задач алгоритмічного характеру.

Для досягнення поставлених завдань було проведено аналіз можливостей, ефективності та областей застосування імперативних мов програмування та технологій, що на них базуються. Серед них обрано наступні: HTML5, CSS3, PHP >= 5.6.3, MySQL 5.6, JavaScript, C++, Windows API, STL, Vagrant, Composer, Twig, Klein, EloquentORM, JQuery. На основі цього розроблено відповідну архітектуру програмного комплексу:

1. сервер проведення олімпіади, що містить інформацію про учасників змагання, завдання, набори тестів та результати робіт. Сервер розміщується на хостингу, доступному для всіх учасників;
2. модуль перевірки робіт учасників розміщується на окремому комп’ютері та отримує запити від сервера проведення олімпіади запити на перевірку робіт по мірі їх відправки учасниками олімпіади та повертає результати, які повідомляються учасникам;
3. клієнтська частина розміщується на домені в мережі Інтернет, при переході на який учасники, по проходженню ідентифікації, отримують доступ до особистої сторінки та можливості відправки завдань. При цьому забезпечується регламент змагання та збереження конфіденційної інформації про учасників.

Така структура дозволяє організовувати олімпіади та конкурси з інформатики серед учнів шкіл та студентів вищих навчальних закладів в дистанційному режимі та з автоматичною перевіркою робіт учасників.

**Апробація результатів.** Систему автоматизованого проведення олімпіад з інформатики було використано під час проведення III обласного етапу Всеукраїнської учнівської олімпіади з інформатики 10 - 11 січня 2015 р. на базі Рівненського державного гуманітарного університету.

**Структура роботи.** Дипломна робота складається з вступу, п’яти розділів, висновків та переліку використаних джерел. Перший розділ описує модель функціонування системи проведення олімпіад з інформатики та її основні складові. В другому розділі загально описані використані технології та їх переваги. Третій розділ присвячено серверу проведення олімпіади та базі даних, що містить інформацію про учасників змагання, завдання та тести до них. Також описуються особливості функціонування серверу. Четвертий розділ висвітлює особливості роботи модулю перевірки робіт учасників та його зв’язку з сервером проведення олімпіади. У п’ятому розділі розглядається клієнтська частина системи (інтерфейс клієнта) та процес відправки завдань учасниками під час проведення змагання. У висновку описано переваги автоматизованої системи проведення олімпіад з інформатики та аспекти її застосування. Повний обсяг роботи становить TODO сторінок. Список використаних джерел включає TODO найменувань.

# РОЗДІЛ 1. МОДЕЛЬ ФУНКЦІОНУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ПРОВЕДЕННЯ ОЛІМПІАД З ІНФОРМАТИКИ

Автоматизована система проведення олімпіад з інформатики дозволяє проводити олімпіади та конкурси, які передбачають розв’язок задач алгоритмічного характеру. Тобто основними результатами роботи учасника має бути код без помилок компіляції, що реалізує правильний та ефективний алгоритм.

Традиційно запропоновані задачі відповідають такій структурі:

1. розробити програму, яка за вхідним файлом визначеної структури буде формувати вихідний, згідно з умовами задачі;
2. розробити програму, що отримуватиме вхідні дані та повідомлятиме про результати їх обробки, інтерактивно взаємодіючи з бібліотекою тестів журі;
3. за відомими вхідними даними отримати результати, що відповідають умові задачі;
4. інші задачі алгоритмічного характеру, розв’язання яких не потребує використання архітектурних особливостей комп’ютерів, знань з області системного програмування, організації складного інтерфейсу користувача, використання нестандартних бібліотек.

Автоматизація – один з напрямів науково-технічного прогресу, спрямований на застосування саморегульованих технічних засобів, економіко-математичних методів і систем керування, що звільняють людину від участі в процесах отримання, перетворення, передачі і використання енергії, матеріалів чи інформації, істотно зменшують міру цієї участі чи трудомісткість виконуваних операцій. Разом із терміном автоматичний, використовується поняття автоматизований, що підкреслює відносно великий ступінь участі людини в процесі [1].

Мета автоматизації – підвищення продуктивності праці, поліпшення якості продукції, оптимізація управління.

Система автоматизації – інформаційно об'єднана сукупність програмованих пристроїв або програмних продуктів автоматизованого та автоматичного контролю, регулювання та управління.

Автоматизована система проведення олімпіад з інформатики реалізована з використанням Інтернет-інтерфейсу користувача, тобто шляхом переходу на відповідний сайт, користувач має змогу використовувати клієнтську частину системи. Ця особливість дозволяє уникнути необхідності використання учасниками складного програмного забезпечення та оптимізує час, відведений на проведення олімпіади. Розроблено окремий інтерфейс адміністратора, що дозволяє працювати з системою без роботи з файлами сервера або використанням мови програмування напряму. Це дозволяє забезпечити цілісність даних, регламент проведення олімпіади та уникнути можливих збоїв у роботі системи.

Серверна частина системи підтримує роботу сайту, забезпечує захист інформації, що є конфіденційною, зберігає результати діяльності учасників олімпіади, надає або забороняє доступ до ресурсів системи, забезпечує рівні умови участі в олімпіаді. Серверна частина системи може бути встановлена на будь-якому сервері (як з використанням хостингу так і на окремому сервері), що підтримує можливість розміщення файлів, містить необхідне програмне забезпечення для обробки запитів до цих файлів. Передбачається, що всі учасники змагання мають доступ до серверу проведення олімпіади.

Модуль перевірки завдань є окремою частиною системи та працює окремо від серверу, передаючи результати роботи через Інтернет мережу. Це дозволяє ефективно використовувати наявні обчислювальні ресурси та не залежати від можливостей серверу, що якісно відображається на роботі системи в цілому.

Завдяки модульній системі підключення компонент серверу (кожна компонента функціонує незалежно та може бути підключена в будь-якому місці клієнтської частини) інтерфейс є достатньо гнучким та може бути видозмінений відповідно до потреб адміністрації. Також до інтерфейсу користувача додано панель адміністрування, яка дозволяє проводити швидкі налаштування системи, такі як:

* встановлення часу відведеного на вирішення завдань;
* завантаження та редагування умов завдань;
* завантаження тестів;
* перегляд турнірних таблиць.
* TODO

Програмний комплекс реалізовано на основі поширених та ефективних технологій засобами HTML, CSS, PHP, MySQL, JavaScript, C++. Завдяки цьому можна уникнути використання складного вартісного програмного забезпечення та використовувати лише підключення до мережі Інтернет.

# РОЗДІЛ 2. ЗАГАЛЬНІ ВІДОМОВСТІ ПРО ВИКОРИСТАНІ ТЕХНОЛОГІЇ

Як було зазначено вище, система складається з 3-х основних частин:

1. Серверна частина – відповідає за зберігання, обробку та маніпулювання даними, які приходять від користувацької частини, та їх передачі у частину перевірки правильності розв’язків. Для реалізації надійної та швидкої серверної частини було взято NGINX веб-сервер та PHP v5.6.3 інтерпретатор для обробки запитів, а також відкриту СКБД MySQL v5.6 для зберігання інформації про поточний стан системи. Для підтримки стабільності під час модифікацій системи було написано ряд Unit, Acceptance та Functional тестів за допомогою пакету PHPUnit.
2. UI-дизайн, або користувацький інтерфейс – забезпечує зручну, легку, інтуїтивно-зрозумілу взаємодію користувача з серверною частиною. Це перше, що користувач бачить, заходячи на сайт, і те, що формує його перше враження про систему. Інтерфейс забезпечує доступ тільки до потрібного користувачу функціоналу, таким чином забезпечуючи захист серверної частини від зловмисників. Для зручної реалізації користувацької частини було використано Bootstrap UI-фреймворк з Material дизайном, який генерується за допомогою шаблонізатору (компілюючого обробника сирцевого коду) Twig.
3. Модуль перевірки завдань учасників – функціонує окремо від серверу та інтерфейсу користувача, при цьому використовуються обчислювальні ресурси сторонньої ЕОМ, що дозволяє зберігати ресурси серверу виключно для обробки запитів від користувачів. Модуль реалізований на C++13, з використанням MFC та STL.

## Веб сервери

Інтернет увірвався в наше життя на величезній швидкості, приручивши до себе та обплутавши всіх своєю павутиною і не збирається нас відпускати. Щодня гортаючи десятки, сотні сторінок сайтів, мало хто замислюється про те, що собою являє ця глобальна мережа Інтернет, як вона працює, що означають ті чи інші терміни.

Сукупність веб серверів є основою Інтернету, без яких, не було б всесвітньої павутини. Користувачі просто не змогли б спілкуватися один з одним, відшукувати потрібну їм інформацію, заводити і підтримувати свої сайти і блоги. Веб серверами можуть бути комп'ютери або спеціальні програми, які виконують роль сервера.

Головне завдання веб сервера приймати HTTP-запити від користувачів, обробляти їх, передавати їх на інтерпретатор, після чого видавати HTTP‑відповіді, перетворюючи їх з мільйонів нулів і одиничок в зображення, медіа-потоки, літери, HTML сторінки.

Для доступу до веб-серверу, користувач повинен мати зручний веб‑браузер, який передає веб серверу запити, в потрібному для нього вигляді.

Поряд зі стандартними функціями, деякі веб сервери мають додаткові. Так, наприклад, відповідне програмне забезпечення може фіксувати число звернень користувачів до того чи іншого ресурсу, записувати їх в окремий журнал. А ще вони можуть підтримувати HTTPS, що важливо для захищеного з'єднання між сайтом і користувачем.

Найбільш відомими і поширеними веб серверами є Nginx та Apache. Найпопулярнішим з них є Apache. Його програмне забезпечення використовується більш ніж на 60% всіх існуючих. Багато в чому це стало можливим завдяки тому, що Apache є безкоштовним. Розробники постійно аналізують помилки свого продукту і усувають їх. Недолік програми в тому, що за великим рахунком вона призначена для професійних програмістів і користувачів, тому обслуговувати і налаштовувати її досить важко [2].

І Apache і Nginx відмінно працюють з багатьма операційними системами, наприклад з Mac OS, Linux, різними модифікаціями Microsoft Windows і багатьма іншими. Ці два веб сервера стабільні в роботі, мають хорошу швидкість, до них можна підключити додаткові модулі.

В роботі був використаний Nginx веб сервер, так як він є більш швидким, ніж його напарник Apache, а також в нього є вбудована функція кешування.

## Мова програмування PHP

PHP - це широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом. Говорячи простіше, PHP це мова програмування, спеціально розроблена для написання веб -додатків (сценаріїв), що виконуються на веб-сервері.

Абревіатура PHP означає «Hypertext Preprocessor (Препроцесор Гіпертексту)». Синтаксис мови бере початок з C, Java і Perl. PHP досить простий для вивчення. Його перевагою є надання веб-розробникам можливості швидкого створення динамічно-генерованих веб-сторінок [3].

Значною відмінністю PHP від ​​будь-якого коду, що виконується на стороні клієнта, наприклад JavaScript, є те, що PHP-скрипти виконуються на стороні сервера. Ви навіть можете налаштувати свій сервер таким чином, щоб HTML‑файли оброблялися процесором PHP, так що клієнти навіть не зможуть дізнатися, чи отримують вони звичайний HTML-файл або результат виконання скрипта.

PHP дозволяє створювати якісні веб-додатки за дуже короткі терміни, отримуючи продукти, легко модифікуються і підтримувані в майбутньому.

Загалом, існують три основні області, де використовується PHP [4]:

1. Найпоширенішою областю є створення скриптів для виконання на стороні сервера.
2. Створення скриптів для виконання в командному рядку. Ви можете створити PHP-скрипт, здатний запускатися незалежно від веб-сервера та браузера. Такий спосіб використання PHP ідеально підходить для скриптів, які повинні виконуватися регулярно, наприклад, за допомогою cron (на платформах \*nix або Linux) або за допомогою планувальника завдань (Task Scheduler) на платформах Windows. Ці скрипти також можуть бути використані в задачах простий обробки текстів. Детальніше про це йдеться тут.
3. Створення додатків GUI, що виконуються на стороні клієнта. Можливо, PHP є не найкращою мовою для створення подібних додатків, але, якщо ви дуже добре знаєте PHP і хотіли б використовувати деякі його можливості в своїх клієнт-додатках, ви можете використовувати PHP-GTK для створення таких додатків. Подібним чином Ви можете створювати і крос-платформні додатки. PHP-GTK є розширенням PHP і не поставляється разом з дистрибутивом PHP.

## Реляційна БД MySQL

MySQL - це одна з найпопулярніших і найпоширеніших СКБД (система контролю баз даних) в інтернеті. Вона не призначена для роботи з великими обсягами інформації, але її застосування ідеально для інтернет сайтів [5].

MySQL відрізняться хорошою швидкістю роботи, надійністю, гнучкістю. При роботі з нею, як правило, не викликає великих труднощів. Підтримка сервера MySQL автоматично включається в модулі PHP.

Важливим фактором є її безкоштовність. MySQL поширюється на умовах загальної ліцензії GNU (GPL, GNU Public License).

Раніше для довготривалого зберігання інформації ми працювали з файлами: поміщали в них деяку кількість рядків, а потім витягували їх для подальшої роботи. Завдання тривалого зберігання інформації дуже часто зустрічається в програмуванні веб-додатків: підрахунок відвідувачів, зберігання повідомлень у форумі, віддалене управління змістом інформації на сайті і т.д [6].

Тим часом, професійні прийоми роботи з файлами дуже трудомісткі: необхідно піклуватись про переміщення в них інформації, її сортування, розширення, при цьому не потрібно забувати, що всі ці дії будуть відбуватися на сервері хостинг-провайдера, де з дуже великою ймовірністю стоїть один з варіантів Unix – отже, потрібно так само піклуватись і про права доступу до файлів і їх розміщення. При цьому обсяг коду значно зростає, і зробити помилку в програмі дуже просто.

Всі ці проблеми вирішує використання бази даних. Бази даних самі дбають про безпеку інформації та її сортування і дозволяють отримувати і розміщувати інформацію за допомогою одного рядка‑запиту. Код з використанням бази даних виходить більш компактним, і зневаджувати його набагато легше. Крім того, не потрібно забувати і про швидкість, адже вибірка інформації з бази даних відбувається значно швидше, ніж з файлів.

Додаток на РНР, що використовує для зберігання інформації базу даних (зокрема MySql) завжди працює швидше додатків, побудованих на файлах. Справа в тому, що бази даних написані на мові C ++, і написати на PHP програму, яка працювала б з диском ефективніше бази даних - завдання нерозв'язне за визначенням, оскільки програми на PHP в принципі працюють повільніше, ніж програми на C ++, так як РНР - інтерпретатор, а С ++ - компілятор.

Таким чином, основна перевага бази даних полягає в тому, що вона бере на себе всю роботу з жорстким диском і робить це дуже ефективно.

Існують такі різновиди баз даних [7]:

* Ієрархічна база даних заснована на структурі дерева зберігання інформації. У цьому сенсі ієрархічні бази даних дуже нагадують файлову систему комп'ютера.
* У реляційних базах даних дані зібрані в таблиці, які в свою чергу складаються з стовпців і рядків, на перетині яких розташовані осередки. Запити до таких баз даних повертає таблицю, яка повторно може брати участь в наступному запиті. Дані в одних таблицях, як правило, пов'язані з даними інших таблиць, звідки і пішла назва "реляційні".
* В об'єктно-орієнтованих базах даних дані зберігаються у вигляді об'єктів. З об'єктно-орієнтованими базами даних зручно працювати, застосовуючи об'єктно-орієнтоване програмування. Однак, на сьогоднішній день такі бази даних ще не досягли популярності реляційних, оскільки поки значно поступаються їм в продуктивності.
* Гібридні СУБД поєднують в собі можливості реляційних і об'єктно-орієнтованих баз даних.

У веб-додатках, як правило, використовуються реляційні бази даних. Коротко її особливості можна описати таким чином:

1. Дані зберігаються в таблицях, що складаються із стовпців і рядків;
2. На перетині кожного стовпця і рядка стоїть в точності одне значення;
3. У кожного стовпця є своє ім'я, яке служить його назвою, і всі значення в одному стовпці мають один тип;
4. Стовпці розташовуються в певному порядку, який визначається при створенні таблиці, на відміну від рядків, які розташовуються в довільному порядку. У таблиці може не бути не одного рядка, але обов'язково повинен бути хоча б один стовпець;
5. Запити до бази даних повертають результат у вигляді таблиць, які теж можуть виступати як об'єкт запитів.

## Шаблонізатор Twig

Twig – компілльований обробник шаблонів з відкритим кодом, написаний на мові програмування PHP. Армін Ронахер написав Twig в 2008 році для платформи блогів Chyrp. Він більше не повертався до розробки і більшою мірою займався розробкою на Python. Синтаксис мови шаблонів Twig бере початок від двигунів шаблонів Jinja і Django, перший з яких також створений Ронахером. Ідею даного шаблонізатора розвиває і підтримує Фаб'єн Потенсье, провідний розробник і ідеолог фреймворка Symfony, в якому Twig використовується за умовчанням [8].

Як правило, всі шаблонізатори схожі між собою і розрізняються тільки якимись специфічними особливостями і деталями. Головне завдання шаблонізатора - розділити бізнес логіку програми та виведення даних на сторінку, таким чином, дозволивши розробникам і дизайнерам працювати одночасно, не заважаючи один-одному.

Багато PHP фреймворків, включаючи: Zend Frameworkd, Agavi, CackPHP і CodeIgniter, по-своєму реалізують поділ бізнес логіки і виведення даних. Однак, якщо ви не любите фреймворки або ваш проект занадто малий для їх використання, то ви можете скористатися якоюсь окремою системою побудови шаблонів. На щастя, нам є з чого вибирати: Smartym Savant, Dwoo ... цей список можна продовжувати і продовжувати, однак в дипломній роботі був вибраний Twig за [9]:

* феноменально швидка робота (в порівнянні з іншими шаблонізаторами);
* компіляція шаблонів в добре оптимізований PHP код;
* застосування "наслідування" (можна визначити базовий і дочірні шаблони);
* дозволяє створювати і використовувати макроси.

Існує безліч способів встановлення Twig. Найпростіший і найшвидший – це скачати компонент з GitHub, після чого архів необхідно розпакувати, і перекинути каталог lib в папку з нашим проектом.

Звичайний PHP додаток складається з цілого набору сторінок, які включають в себе як статичний HTML код (меню, списки, зображення і т.д.), так і динамічний контент (вивід даних з БД, xml файлів, сервісів, ...). За допомогою Twig ми можемо розділити дані процеси, створюючи шаблони зі спеціальними змінними, замість яких в подальшому буде вставлятися динамічний контент.

Значення для даних змінних формуються в основному PHP скрипті, там же відбувається спілкування з базою даних, xml парсинг та інші всілякі операції. Таким чином, наша сторінка буде будуватися на основі 2-х джерел: шаблону з спеціальними вставками і PHP скриптів, де ми зберігаємо основний функціонал. Це дає можливість PHP розробникам і дизайнерам одночасно працювати над одними і тими ж сторінками.

Звичайно, для PHP написано велику кількість шаблонизаторів. Але більшість з них написані під PHP4 і не використовують кращі особливості розробки:

1. Можливість розширення: Twig – це гнучкий движок для будь-яких ваших потреб, навіть найскладніших. Завдяки відкритій архітектурі, ви можете визначати нові мовні конструкції (теги і фільтри) для створення свого власного шаблону.
2. Юніт-тестування: Twig повністю покритий тестами. Бібліотека стабільна і готова до використання у великих проектах.
3. Документація: Twig повністю документований, вся документація доступна на сайті, і, звичайно, має повний опис API.
4. Безпека: Щодо безпеки, є кілька абсолютно унікальних особливостей:
   1. Автоматичне екранування виведення: Для безпечного виведення даних, ви можете включити екранування як глобально, так і для окремих блоків.
   2. Пісочниця: Twig дозволяє використовувати для будь-якого шаблону "пісочницю", де користувачі мають обмежений набір тегів, фільтрів і методів об'єктів, визначених розробником. Режим "пісочниці" може бути включений як глобально, так і локально, для певних шаблонів.
5. Чисті повідомлення про помилки: Кожен раз, коли у вас виникають синтаксичні помилки в шаблоні, шаблонізатор виводить повідомлення про ім’я файлу з помилкою і номер рядка, яка її викликала. Це дуже спрощує зневадження.
6. Швидкий: Одна з цілей створення даного шаблонізатора - зробити його настільки швидким, наскільки це можливо. Для досягнення максимальної швидкості роботи, Twig компілює шаблони в оптимізований PHP код. Використання пам'яті в порівнянні зі звичайним PHP кодом зведено до мінімуму.

# РОЗДІЛ 3. СЕРВЕРНА ЧАСТИНА СИСТЕМИ. СЕРВЕР ПРОВЕДЕННЯ ОЛІМПІАДИ

Серверна частина системи є її основною та забезпечує взаємозв’язок між інтерфейсом та модулем перевірки завдань

## Особливості роботи

Серверна частина системи є набором php-інструкцій, що генерують відповідні html-сторінки для користувачів, забезпечують захист та конфіденційність робіт учасників, тестів для перевірки робіт, регламентують час проведення олімпіади та керують доступом учасників до сервісу.

Серверна частина системи може бути встановлена на будь-якому сервері (як з використанням хостингу так і на окремому сервері), що підтримує можливість розміщення файлів, містить необхідне програмне забезпечення для обробки запитів до цих файлів, підтримує мову PHP та роботу з базою даних на основі SQL. Так, наприклад, може бути використано відкритий веб-сервер Apache, що підтримує UNIX-подібні, Microsoft Windows, Novell NetWare, OS/2, інші операційні системи та є найпоширенішим веб-сервером з 1996 року. Саме на налаштування веб-сервера покладається захист доступу до файлів програми, бази даних учасників та їх робіт, захист від можливих DDOS-атак під час проведення олімпіади та ведення системного журналу.

Система реалізована на об'єктно-орієнтованому представлені MVC (Model View Controller – Модель-Вид-Контролер) парадигми за допомогою Klein-роутінгу, який підтримує REST‑стандарт. Кожна сторінка має свій контролер-клас, в якому формується набір даних які представляються користувачу у зрозумілому для нього вигляді. Це дає можливість уникнути залежності від конкретного користувацького інтерфейсу, інкапсулювати окремі набори команд та додавати серверні інструкції в будь-які елементи інтерфейсу формуючи нові можливості системи без необхідності корегування функціонуючих. Також для можливості введення мультимовності системи всі текстові константи отримуються за допомогою єдиного інтерфейсу, який повертає користувачу перевизначені константи, які збережені в окремих файлах.

## База даних

Для збереження інформації про учасників, їх подальшої ідентифікації, збереження умов завдань та наборів тестів до кожного з них використовується реляційна база даних на основі відкритої СКБД MySQL. В базі даних зберігаються відомості про:

* Адміністраторів проведення олімпіади. Адміністратор має доступ до адміністративного інтерфейсу, що дозволяє встановлювати дату проведення та тривалість олімпіади, додавати та редагувати завдання та тести до цих завдань, переглядати відомості про учасників та їх успішність під час проведення олімпіади та по закінченню. В базі даних зберігається логін та пароль для входу та електронна пошта адміністратора, що може бути використана для відновлення доступу при втраті пароля.
* Учасників олімпіади. Кожен учасник олімпіади має доступ до клієнтського інтерфейсу системи, що дозволяє переглядати завдання та надсилати розв’язки до них, також учасник може переглядати рейтингову таблицю. Доступ до клієнтського інтерфейсу надається лише після початку олімпіади. Можливість відправки завдань надається лише під час проведення олімпіади. В базі даних зберігаються основні дані: логін та пароль учасника, електронна скринька(використовується для відновлення пароля при його втраті та для зв’язку з адміністрацією), ПІБ учасника, відомості про його навчальний заклад та ПІБ особи, що проводила його підготовку до олімпіади. Також зберігаються його загальна оцінка, кількість та номери розв’язаних завдань, величина штрафу по кожному з них.
* Завдання. В базі даних зберігається текст кожного завдання, вхідні та вихідні дані для його перевірки.

Зверення до бази даних проходять автоматично та згідно з правами доступу кожного користувача (адміністратора або учасника олімпіади).

## Доступ до системи

Для обмеження несанкціонованого доступу до систем було використано технологію SHA1 хешування паролів, тому ніхто (включаючи адміністрацію) не має доступу до паролів учасників олімпіади, технологію HTTP-Cookie та додаткове кодування паролю на сервері, що забезпечує авторизацію по принципу один авторизований користувач - один комп’ютер (для учасників олімпіади).

Процес авторизації учасника олімпіади проводиться в декілька етапів:

1. При введенні користувачем персонального логіну та паролю проводиться інкапсулювання заборонених символів, для усунення можливих SQL-ін’єкцій зі сторони зловмисників.
2. До паролю користувача додається відповідний випадково-генерований рядок, що генерується при кожній авторизації користувача.
3. Перевіряється наявність комбінації логін-пароль в базі даних (реально логін – SHA1-хеш паролю з випадково-генерованим рядком, згідно принципу конфіденційності паролів).
4. Генерується новий випадковий рядок, що додається до паролю та записується в базу даних замість попереднього, вже не дійсного, запису.
5. Додається інформація про учасника у PHP-сесію, яка зберігається на сервері.
6. Перевіряється умова початку олімпіади.
7. Здійснюється перехід на персональну сторінку користувача.

При помилці на будь-якому з етапів система забороняє доступ до персональної сторінки користувача та надає інформацію про зміст помилки.

При будь-якій операції на персональній сторінці, перевіряються дані про користувача, записані в сесіях, а також умова кінця олімпіади. При повторній авторизації або виході з персональної сторінки сесія знищується, що автоматично закриває доступ до сторінки. Також по закінченню олімпіади зникає можливість відправки рішень (відповідний блок персональної сторінки).

## Системний таймер

Система використовує серверний таймер та PHP функції для роботи з часом для встановлення та визначення часових рамок проведення олімпіади. Так можна уникнути особливостей часового поясу при проведенні міжнародних олімпіад. Дані про початок олімпіади та її тривалість зберігаються в UNIX‑timestamp форматі, що дозволяє працювати з всіма необхідними одиницями (роки, місяці, дні, години, хвилини) при необхідності.

## Форма завантаження рішень

Під час проведення олімпіади кожен учасник має доступ до форми завантаження рішень, що реалізована засобами HTML5, обробляється засобами PHP, MySQL та JavaScript. При завантаженні нового файлу перевіряється на допустимість його ім’я, розмір та розширення (система підтримує розширення ‘.cpp’, ‘.c’, ‘.pas’). На основі розширення файлу буде обрано відповідний компілятор. Далі формується шлях до файлу для його збереження на сервері. Всі завантажені рішення зберігаються на сервері в папці користувача для обробки можливих апеляцій зі сторони учасників по закінченню олімпіади.

## Черга рішень

Кожна задача може знаходитись в одному з 7 станів:

* 0 – не оброблювалась
* 1 – в черзі задач
* 2 – компілюється
* 3 – успішно скомпільована
* 4 – помилка компіляції
* 5 – помилка виконання
* 6 – успішно прийнята

При відправці рішення учасником олімпіади, воно автоматично відправляється в чергу завдань на компіляцію, при помилці компіляції учасник отримує відповідне попередження (та штрафується на певну кількість балів, якщо це обумовлено правилами проведення олімпіади), в іншому випадку завдання перевіряється на множині тестів та повертається результат виконання: помилка з вказаним номером тесту або успіх. У разі успіху учасник отримує певну кількість балів (відповідно до налаштувань системи).

## Форма реєстрації

Для реєстрації учасників шляхом прямого вказання персональних даних (формування списку учасників), розроблено форму реєстрації, що обробляється відповідними PHP та SQL інструкціями. На етапі реєстрації формується запис про учасника у базі даних, що дозволить авторизуватись у системі для участі у олімпіаді.

Форма складається з набору основних полів: (TODO)

* Персональний логін
* Пароль
* Повторний ввід пароля
* Адреса електронної скриньки
* Прізвище
* Ім’я
* По-батькові
* Відомості про навчальний заклад
* Клас/група
* ПІБ викладача/особи, що проводила підготовку учасника до олімпіади
* Поле для вводу CAPTCHA

Перевірка форми проводиться в декілька етапів:

1. Перевірка вірності введеного коду CAPTCHA, що забезпечить додатковий захист від можливих атак зловмисників;
2. Перевірка довжини та складу персонального логіну учасника. Він повинен складатися з не менше ніж 5 та не більше ніж 30 букв, цифр та знаків ‘-‘,’\_’.
3. Перевірка довжини паролю, що повинен містити не менше ніж 6 знаків.
4. Перевірка еквівалентності рядків в полях «Пароль» та «Повторний ввід пароля», що є додатковим захистом на коректність вводу пароля для користувача.
5. Перевірка наявності даних в кожному з полів, оскільки всі вони є обов’язковими для заповнення.
6. Перевірка допустимості введених в поля даних. Для запобігання SQL-ін’єкцій встановлено обмеження на ввід інформації в поля. Так в поля «Логін», «Пароль», «Адреса електронної скриньки» дозволено вводити латинські букви, цифри та знаки ‘-‘,’\_’ (знаки ‘@’, ‘.’ для поля «Адреса електронної скриньки»). В поля «Прізвище», «Ім’я», «По-батькові» - кириличні символи та знак апострофу. В поле «Відомості про навчальний заклад» - кириличні символи, цифри, апостроф, знаки ‘«’, ‘»’, ‘-’, ‘№’ та пробіл. В поле «ПІБ викладача/особи, що проводила підготовку учасника до олімпіади» - кириличні символи та пробіл.
7. Перевірка поля «Адреса електронної скриньки» на відповідність масці вводу для електронних скриньок.
8. Перевіряється наявність записів в базі даних з ідентичним полем «Логін» (для уникнення дублювання логінів користувачів).
9. Формується випадковий рядок з латинських букв, цифр, знаків ‘~!@#$%^&\*()’ та додається до введеного паролю користувача.
10. Сформований пароль хешується та відповідний запис з відомостями про користувача додається до бази даних.

# РОЗДІЛ 4. МОДУЛЬ ПЕРЕВІРКИ ЗАВДАНЬ УЧАСНИКІВ

Модуль перевірки завдань учасників функціонує окремо від серверу та інтерфейсу користувача, при цьому використовуються обчислювальні ресурси сторонньої ЕОМ, що дозволяє зберігати ресурси серверу виключно для обробки запитів від користувачів. Це покращує швидкодію системи та дозволяє оптимально підібрати ЕОМ, на якій буде розміщено модуль перевірки завдань, відповідно до запланованих навантажень (кількості користувачів системи, кількості тестів, що обробляються, складності рішень учасників).

1. 1. Особливості роботи

На відміну від серверів, що найчастіше використовують UNIX-подібні системи, до ряду переваг яких належать відкритість програмного коду та розповсюдження на умовах ліцензії GPL, інші ЕОМ (персональні комп’ютери, портативні та кластерні системи) найчастіше використовують системи Windows, що є найпоширенішими та інтуїтивними у користуванні.

Для модулю перевірки завдань учасників обрано технологію Windows API та мову програмування С++ з використанням бібліотек STL. Це дозволяє ефективно взаємодіяти з ресурсами системи, максимально використовуючи процесорний час.

Запущена на будь-якому комп’ютері (з використанням ОС Windows) програма автоматично з’єднується з сервером системи автоматизованої системи проведення олімпіад з інформатики та шляхом GET запитів з відповідної сторінки (недоступної користувачам системи) отримує доступ до рішень учасників та наборів тестів (вхідних та вихідних даних, що використовуються для перевірки рішень). Оскільки рішення учасників, відправлені на перевірку шляхом завантаження їх на сервер, перебувають в черзі рішень, програма також зчитує дані з черги по мірі її поповнення. Кожне рішення компілюється відповідним компілятором (залежно від розширення файлу рішення) та запускається як дочірній процес з власною адресною областю. Це дозволяє зберегти ресурси, виділені на виконання рішення учасника, та повністю контролювати ввід і вивід запущеного процесу. Запущена користувацька програма перевіряється на множині тестів, результати виконання програми (її вихідні дані) звіряються з еталонними та повертається результат її виконання (успіх або невдача з зазначенням номеру тесту, на якому програма повернула результат, що вважається невірним). Максимізації використання обчислювальних ресурсів ЕОМ, а саме процесорного часу реалізується за рахунок багато поточного виконання процесу компіляції та тестування користувацьких програм з черги рішень.

Кількість запущених потоків обчислюється за рахунок кількості процесорів (процесорних ядер) в системі. Тому з метою використання істинного паралелізму число запущених потоків дорівнює числу процесорів в системі. Кожен потік відповідає одному рішенню учасника з черги рішень, що повертає результат виконання програми у вигляді успіху чи невдачі.

* 1. З’єднання з сервером системи

З’єднання з сервером системи інкапсульовано в окремий клас та реалізується за допомогою класів Microsoft Foundation Classes (надбудови до Windows API), а саме CInternetSession (одно з класів, похідних від WinInet). До ряду переваг використання такого способу підключення до сервера належать:

* автоматичне створення підключення по замовчуванню, що дозволяє уникнути залежностей від типу підключення конкретної системи;
* використання HTTP-Cookie, що забезпечує безпеку з’єднання з сервером, використовуючи особистий логін та пароль;
* можливість зчитувати HTML сторінки у вигляді об’єктів класу CStdioFile (похідний від CFile).

Метод OpenURL() класу CInternetSession класу забезпечує з’єднання з сторінкою доступу до черги рішень та множини тестів, тип з’єднання автоматично визначається як HTTP на основі рядку адреси сторінки, що стандартно починається з «http://» результатом роботи методу є об’єкт класу CStdioFile заповнений даними з сторінки, шлях до якої передається в першому з параметрів методу.

* 1. Клас Task

Клас Task використовується в формуванні програмної черги рішень при зчитуванні інформації з сервера, його об’єкт представляє собою одне із рішень учасника олімпіади відправлене на компіляцію. Кожен об’єкт класу містить:

* шляхи до файлу рішення учасника. Кожне рішення знаходиться в папці користувача на сервері у відповідному форматі;
* шлях до файлу програми, що буде тестуватись. Програма на цьому етапі ще не скомпільована, проте, опираючись на те, що вона буде розміщуватись в папці користувача, шлях до неї можна сформувати за допомогою методів обробки рядків класу CString;
* ідентифікатор рішення згідно серверної черги рішень;
* кількість тестів для даного рішення;
* тип рішення. Тип файлу рішення визначається за допомогою методів класу CString з шляху до файлу рішення. Тип рішення буде використано при його компіляції.
  1. Доступ до інформації на сервері

Програма зчитує інформацію з відповідної сторінки сервера, що формується набором PHP-інструкцій згідно запиту програми. Програма може відправляти на сервер GET-запити 3 типів:

* запит на зчитування набору тестів. У відповідь на цей запит сервер формує сторінку з блоками інформації «div», що помічені відповідними індексами «task\_inp», «task\_out» та містять множини вхідних і вихідних даних відповідно. Ідентифікатори використовуються програмою при формуванні множини тестів. Метод GetTest класу Connection використовуючи структуру даних vector<vector<pair<CString, CString>> зчитує множину тестів (текст сформованої сервером HTML сторінки у відповідь на запит програми). Множина тестів зчитується один раз при запуску програми та використовується при перевірці рішень учасників. Передбачається, що множина тестів не буде змінюватись адміністрацією під час проходження олімпіади.
* запит на доступ до черги рішень. У відповідь на цей запит сервер формує сторінку з блоками інформації «div», що помічені відповідними індексами «task», «path» та містять шляхи до файлів користувацьких рішень згідно порядку черги. Метод GetQueue класу Connection використовуючи структуру даних queue<Task> зчитує дані з сторінки та формує програмну чергу шляхів до рішень, що будуть оброблятись не порушуючи порядку черги.
* результат компіляції програми. Цей запит змушує сервер змінити відомості про рішення в базі даних, а саме стан цього рішення 5 або 3 (згідно таблиці станів) відповідно до унікального ідентифікатора рішення в базі даних.
  1. Класи Process, Potok. Передача даних для дочірніх процесів

Основною задачею класу Process є обробка вхідних та вихідних даних користувацьких програм, збереження дескрипторів потоків вводу/виводу та security структур для каналів передачі даних.

Кожне користувацьке рішення запускається програмою у вигляді дочірнього процесу. Це дозволяє використовувати технологію комунікації між процесами – Pipes (*англ.*труби). Кожна така труба забезпечує передачу даних між батьківським та дочірнім процесом або між двома дочірніми процесами, використовуючи дескриптори для потоків вводу та виводу.

Програма використовує неіменовані (односторонні) канали зв’язку, що дозволяють передавати або зчитувати дані з програмних потоків вводу (stdin) та виводу (stdout) відповідно. Кожен об’єкт класу Process за допомогою функції CreatePipe() отримує доступ до потоків вводу та виводу через відповідні дескриптори. Методи WriteInPipe() та ReadOutPipe() дозволяють передавати дані на вхідний потік користувацької програмі та зчитувати їх з вихідного потоку у форматі об’єктів класу CString. Зчитування відбувається посимвольно.

Клас Potok відповідає за контроль над множиною створених програмою потоків, містить дескриптори для кожного з них, отримує доступ до системної інформації, а саме кількості процесорів в системі та відповідає за створення дерева каталогів в папці користувача, що дозволяє контролювати кількість відправлених на перевірку рішень та полегшує пошук необхідних при апеляціях зі сторони учасників олімпіади (всі рішення, відправлені на перевірку зберігаються у відповідних каталогах).

## Розподіл роботи. Компіляція та перевірка рішень

На початку роботи програми створюються об’єкти основних класів, формується підключення до серверу системи, ініціалізуються структури даних для множини тестів, черги рішень, визначається кількість процесорів в системі, формуються масиви для збереження інформації про потоки та дескриптори вводу/виводу користувацьких програм, ініціалізуються критичні секції. Механізм критичних секцій використовується для синхронізованого зчитування інформації потоками з черги рішень. Функція CreateThread() використовується для запуску потоків роботи програми, кожен з яких отримує унікальний ідентифікатор та екземпляр функції компіляції та обробки користувацьких рішень.

Черга рішень та множина тестів є глобальними, що спрощує доступ до цих структур з потоків програми. Кожен потік зчитує рішення (об’єкт класу Task) з черги рішень (вступаючи при цьому в критичну секцію, для забезпечення цілісності операції). Далі створюється каталог для збереження нового варіанту рішення у папці користувача, завантажується файл та згідно типу рішення (відповідне поле об’єкта класу Task) визначається компілятор. Після компіляції потік повідомляє сервер про результати (за допомогою відповідного GET-запиту).

Тестування користувацького рішення проводиться на множині тестів відповідно до інформації отриманої з серверу. Для кожного окремого тесту формуються канали передачі даних (для роботи з вхідним та вихідним потоками даних), запускається користувацька програма з обмеженням на час виконання та черговим тестом, переданим на вхідний потік програми. В разі часу виконання, що перевищує вказаний, процес завершується з основної програми, це гарантує захист від можливих зациклень в користувацьких програмах. Далі зчитуються дані з вихідного потоку програми та звіряються з еталонними. У разі успішного проходження всіх тестів або невірних результатів для будь-якого з них формується відповідний GET-запит (стан 6 або номер тесту при якому програма повернула невірний результат відповідно) та відправляється на сервер.

Програма завершує роботу по команді адміністратора. В цьому випадку всі потоки завершують свою роботу (цикл обробки користувацького рішення для кожного з них обов’язково закінчується), закриваються потоки та звільняються зайняті системні ресурси.

# РОЗДІЛ 5. КЛІЄНТСЬКА ЧАСТИНА СИСТЕМИ. ІНТЕРФЕЙС КЛІЄНТА. ВИКОРИСТАННЯ СИСТЕМИ

Інтерфейс клієнта автоматизованої системи проведення олімпіад з інформатики реалізований у вигляді набору HTML сторінок. Завдяки модульній системі підключення компонент серверу (кожна компонента функціонує незалежно та може бути підключена в будь-якому місці клієнтської частини) інтерфейс є достатньо гнучким та може бути видозмінений відповідно до потреб адміністрації.

* 1. Головна сторінка. Форма авторизації

Після переходу за адресою сайту, який обрано інтерфейсом системи користувач потрапляє на головну сторінку, єдиним обов’язковим елементом якої є форма авторизації (на рис. 5.1.1 форму авторизації виділено темно-сірим прямокутником TODO), також там часто розміщують правила проведення олімпіади, час проведення, контактні дані адміністрації або центру підтримки користувачів.

TODO

Рис. 5.1.1 Приклад головної сторінки інтерфейсу системи

Форму авторизації обслуговують PHP-інструкції авторизації користувача в системі, що забезпечують перевірку коректності введеного логіну та паролю користувача, захист від можливих SQL-ін’єкцій, повторної авторизації в системі (забезпечення принципу один користувач – один комп’ютер), захист від несанкціонованого доступу до персональних сторінок користувачів. Форма реєстрації може містити посилання для переходу на сторінки реєстрації та відновлення паролю, якщо це передбачено адміністрацією олімпіади.

* 1. Реєстрація в системі. Відновлення втраченого паролю

Обов’язковим елементом сторінки реєстрації користувача є форма реєстрації, реалізована засобами HTML <form>. Форма реєстрації обслуговується відповідними засобами PHP, що забезпечують коректність введених даних, запобігають можливим помилкам вводу, захищають від SQL-ін’єкцій збоку зловмисників, створенню великої кількості фіктивних користувачів (захист CAPTCHA). Детальніше в п. 3.7.

TODO

Рис. 5.2.1 Приклад використання форми реєстрації

TODO

Рис. 5.2.2 Приклад коректного заповнення форми реєстрації

Система підтримує відновлення втраченого паролю користувача. У разі втрати основного паролю авторизації, користувач може отримати новий. По заповненню форми відновлення паролю користувача.

TODO

Рис. 5.2.3 Приклад форми відновлення паролю користувача

Якщо введені дані (логін та адреса електронної скриньки користувача, вказані при реєстрації) співпадають з будь-яким із записів в базі даних, по натисненню «Відновити пароль» буде згенеровано новий випадковий пароль користувача та вислано його на електронну скриньку. Змінити такий пароль можна буде на відповідній сторінці редагування персональних даних, якщо така можливість буде передбачена адміністрацією олімпіади.

* 1. Персональна сторінка користувача

При успішній авторизації в системі користувач отримує доступ до персональної сторінки.

TODO

Рис. 5.3.1 Авторизація в системі

Обов’язковими елементами якої є блок доступу до завдань олімпіади, форма відправки рішень, блок користувацьких налаштувань, в якому відображаються деякі персональні дані користувача, його загальний результат, адреси сторінок з таблицею результатів користувачів, перегляду та редагування (якщо це дозволяється згідно правил проведення змагання) персональних даних, кнопка виходу з персональної сторінки. Також на сторінці розміщується таймер зворотного відліку часу проведення олімпіади.

TODO

Рис. 5.3.2 Персональна сторінка користувача

Таймер реалізовано засобами JavaScript та PHP, це дозволяє коректно відображати поточний час до закінчення олімпіади не перезавантажуючи сторінку після кожного інтервалу часу.

Блок доступу до завдань реалізовано засобами PHP та HTML, що дозволяє вчасно оновлювати стан рішень (відповідно до таблиці рішень), оскільки заголовок завдання (вкладки з завданням) змінює колір відповідно до стану рішення:

* сірий – рішення не оброблялось;
* жовтий – рішення компілюється;
* червоний – помилка компіляції;
* зелений – рішення успішно прийнято.
  1. Відправка рішень

Відправку рішень реалізовано засобами HTML <form> та PHP. Вибір задачі, для якої користувач бажає відправити рішення реалізується шляхом переходу по відповідній вкладці блоку доступу до завдань олімпіади.

TODO

Рис. 5.4.1 Вибір завдання «В»

По натиску на «Завантажити» блоку відправки рішень, з’являється стандартне діалогове вікно, що дозволяє вибрати файл з жорсткого диску комп’ютера.

TODO

Рис. 5.4.2 Приклад завантаження файлу рішення на сервер

Далі файл автоматично потрапляє в чергу рішень, користувач може слідкувати станом його обробки шляхом оновлення сторінки (файл буде відображатись в блоці відправки рішень у форматі «ідентифікатор назва.формат стан».

TODO

Рис. 5.4.3 Приклад успішно завантаженого файлу

* 1. Адміністрування

Адміністрація олімпіади має доступ до панелі адміністрування, що дозволяє встановлювати дату, час та тривалість олімпіади, додавати та редагувати завдання та тести, переглядати відомості про учасників та таблицю результатів, додавати адміністраторів олімпіади.

Доступ до сторінки адміністрування можна отримати шляхом переходу в папку admin в адресі сайту-інтерфейсу системи, наприклад http://example.com/admin/, та вводу персонального логіну та паролю в формі авторизації аналогічній до користувацької. Засоби PHP, що обслуговують цю форму авторизації здійснюють пошук записів в таблиці адміністраторів (в користувацькій формі – таблиці учасників олімпіади).

TODO

Рис. 5.5.1 Приклад використання адміністративної форми авторизації

При успішній авторизації, адміністратор має можливість користуватись адміністративним функціоналом системи.

TODO

Рис. 5.5.2 Приклад панелі встановлення дату, часи та тривалості олімпіади

TODO

Рис. 5.5.3 Приклад редактора завдань для учасників олімпіади

# ВИСНОВКИ

Розроблений програмний комплекс, є ефективним рішенням для проведення олімпіад та конкурсів з інформатики в дистанційному режимі та/або з автоматичною перевіркою робіт учасників. Такий підхід сприяє підвищенню інтересу до поглибленого вивчення інформатики, пошуку обдарованої молоді, реалізації її здібностей, популяризації досягнень науки, техніки та новітніх технологій, забезпеченню можливості участі в олімпіадах та конкурсах осіб, що не мають фізичної можливості дістатись до місця проведення заходу.

Програмний комплекс реалізовано на основі поширених та ефективних технологій засобами PHP, SQL, C++, WinAPI, JavaScript. Завдяки цьому можна уникнути використання складного вартісного програмного забезпечення та використовувати лише підключення до мережі Інтернет.

Модульна система організації функціоналу сервера забезпечує можливість реалізувати гнучкий інтерфейс користувача відповідно до потреб адміністрації заходу, підключати та відключати окремі можливості для підготовки рівних умов участі.

Використання окремої програми компіляції та перевірки рішень учасників перекладає найскладніші обчислення на окрему обчислювальну машину, звільняючи при цьому сервер системи для обробки запитів користувачів. Такий підхід дозволяє стабілізувати роботу системи, забезпечити максимальну швидкість роботи з користувачем, підвищити гнучкість у виборі обчислювальних ресурсів, що будуть використані для компіляції та перевірки рішень учасників відповідно до запланованої їх кількості.

Система автоматизованого проведення олімпіад та конкурсів з інформатики дає можливість ефективно організувати та провести будь-які заходи, що передбачають розв’язок алгоритмічних завдань, забезпечуючи при цьому рівні умови участі, безпеку персональних даних та мінімізуючи видатки на проведення заходу.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

|  |  |
| --- | --- |
| [1] | «Автоматизація,» 26 12 1026. [В Интернете]. Available: https://uk.wikipedia.org/wiki/Автоматизація. [Дата обращения: 28 12 2017]. |
| [2] | ВМЛАБ, «Що таке Веб-сервер?,» 04 12 2011. [В Интернете]. Available: https://vmlab.ru/Articles/What\_is\_web\_server. [Дата обращения: 29 10 2017]. |
| [3] | А. Транский, «Что такое PHP?,» 14 05 2006. [В Интернете]. Available: http://www.php.su/php/?php. [Дата обращения: 29 10 2017]. |
| [4] | В. В., Zend Framework: разработка приложений на PHP, Санкт-Петербург: Питер, 2012, p. 432. |
| [5] | Т. Коннолли, К. Бегг и А. Страчан, Базы данных: Проектирование, реализация и сопровождение. Теория и практика, Москва: Вильямс,, 2000, p. 1093. |
| [6] | Д. М. А., «Что такое база данных MySQL?,» 28 03 2009. [В Интернете]. Available: http://blogwork.ru/chto-takoe-baza-dannyx-mysql/. [Дата обращения: 29 10 2017]. |
| [7] | Г. М., Понимание SQL, Москва: ДКМ Прес, 1993, p. 291. |
| [8] | Вікіпедія, «Twig,» 26 02 2017. [В Интернете]. Available: https://ru.wikipedia.org/wiki/Twig. [Дата обращения: 29 10 2017]. |
| [9] | Twig, «Русскоязычная документация по Twig,» 15 02 2016. [В Интернете]. Available: http://x-twig.ru/. [Дата обращения: 29 10 2017]. |
| [10] | Б. В.А., Win32 API. Программирование. Учебное пособие, Санкт-Петербург: СПб, 2009, p. 90. |
| [11] | Щ. Ю.А., Win32 API Эффективная разработка приложений, Санкт-Петербург: СПб, 2007, p. 572. |