МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій Форсюк	_____ Сергій Чумаченко
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«___» _____ 2022 р.	«___» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми Інформаційні управляючі системи та технології
на тему: Дослідження та розроблення інформаційної системи збору та обробки даних про матеріальні потоки на сільськогосподарському підприємстві

Виконав: здобувач 2 курсу, групи ІС-2-4М

Бондаренко Даніїл Анатолійович
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Сільвестров Антон Миколайович _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент Микола Островерхов _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ - 2022р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем
Освітній ступінь магістр
Спеціальність 122 «Комп'ютерні науки»
(код і назва)
Освітньо-професійна програма _____
_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач
кафедри_____
_____
"____" _____2022 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Бондаренко Даніїл Анатолійович
(прізвище, ім'я, по батькові)

1. Тема роботи _____
_____
_____
керівник роботи Сільвестров Антон Миколайович, професор, д-р техн. наук,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "___"_____2022 року №_____
2. Строк подання здобувачем роботи_____
3. Вихідні дані до роботи _____
Звітна документація
_____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Провести аналіз роботи системи зважування
2. Проаналізувати дані, які надсилає система зважування до головної програми
_____
3. Дослідити потреби компаній, для яких створюється інформаційна система, та скласти список майбутніх функцій системи
4. Створити програму-додаток
_____
_____
5. Перелік графічного матеріалу
1. Організаційна структура компанії
2. Скріншоти інтерфейсу програми
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ	Сільвестров А.М	28.10.2021	28.10.2021
Розділ 1	Сільвестров А.М	30.10.2021	30.10.2021
Розділ 2	Сільвестров А.М	24.11.2021	24.11.2021
Розділ 3	Сільвестров А.М	08.01.2022	08.01.2022
Висновок	Сільвестров А.М	28.01.2022	28.01.2022
Джерела	Сільвестров А.М	29.01.2022	29.01.2022
Додатки	Сільвестров А.М	30.01.2022	30.01.2022

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ З№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Системний аналіз компанії	28.10-04.11	Виконано
2	Постановка задачі на розроблення системи	04.11-28.12	Виконано
3	Створення додатку користувача	28.12-14.01	Виконано
4	Оформлення пояснювальної записки	14.01-01.02	Виконано
5	Оформлення презентації	01.02-03.02	Виконано

Здобувач _____ Бондаренко Д.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Сільвестров А.М.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

GPS моніторинг – це універсальна, зручна, досить проста система контролю за розташуванням та станом транспорту, а також за іншими об'єктами: людьми, обладнанням, вантажами.

Головним завданням GPS моніторингу є оптимізація робіт пов'язаних із перевезеннями та транспортом загалом. Останнім часом, майже кожна компанія, що має автопарк, почала усвідомлювати потреби контролю його функціонування, враховуючи необхідність оптимізації роботи та зменшення витрат. Перелік того, що може торкатися такого контролю, досить великий, але в основному це стеження:

- за транспортом та його маршрутом (контроль витрати палива, контроль пробігу);
- за людьми (контроль роботи);
- за переміщенням та надійністю вантажів.

Саме тому транспортні компанії вводять супутникові системи GPS моніторингу, які дозволяють у реальному часі здійснювати стеження за становищем транспортного засобу на карті, контролювати маршрут, отримувати точну статистику за швидкісними режимами та простоями, бачити реальний пробіг, здійснювати контроль витрати палива.

Таким чином, використовуючи на своєму підприємстві систему супутникового стеження, Ви отримуєте можливість контролювати та виявляти всі зупинки та заїзди транспортного засобу під час шляху, виключати незаплановані рейси, а також використання транспорту у неробочий час.

Введення в транспортну компанію GPS моніторингу окупається в середньому за декілька місяців, а Ви швидко відчуваєте його переваги:

- скорочення витрат, необхідних підтримки нормального функціонування транспортного підприємства;
- спрощення обліку витрат палива;
- привчання водіїв до дисципліни.

Для компанії «АйТіБі КМ» виникла необхідність в додатковому ПЗ для роботи з зважуванням техніки. Так як компанія вже працює з системою моніторингу, але в ній відсутнє потрібне ПЗ, було поставлено завдання на розробку інформаційної системи для роботи з приладами для зважування техніки з сільськогосподарською продукцією. Розробка інформаційної системи доцільна, тому що:

- не потрібно купувати нову системи;
- зручність використання зв’язаних систем;
- не потрібно додатково навчати персонал працювати з новою програмою;
- створення системи спеціально під потреби компанії, без нагромадження непотрібними функціями.

Також було розроблено алгоритм для пришвидшення зважування, за рахунок переведення системи зважування в роботу динамічного визначення ваги.

1 РОЗДІЛ

1.1 Загальна характеристика компанії

Компанія «АйТіБі КМ» успішно працює на ринку України 3 роки.

Компанія спеціалізувалася на послугах та обладнанні для GPS-моніторингу транспорту.

Маючи досвід ведення бізнесу, навички встановлення GPS-трекерів, систематичне вивчення потреб клієнтів, дослідження пристроїв та програм, компанія продовжує успішно розвивати GPS-моніторинг транспорту, як окремий напрямок бізнесу. Кількість реалізованих GPS-трекерів складає вже 7000 одиниць.

Команда дуже ретельно підходить до своєї роботи, детально вивчає не лише пристрої, а й платформи моніторингу, які представлені на ринку. Відстежуючи тренди, інноваційні технології та підходи до керування автопарком, вони дійшли висновку, що GPS-моніторинг транспорту зараз є невід'ємною частиною ефективною керування бізнесом.

«АйТіБі КМ» полегшує керування транспортними ресурсами, тим самим допомагаючи розвивати ваш бізнес. Компанія прагне надавати лише найякісніші продукти, рішення та послуги.

Місія компанії – підвищити ефективність бізнесу, використовуючи моніторинг автотранспорту.

Цінності компанії – видатна якість, турбота та безпека.

Переваги роботи з компанією:

- надійність та висока якість продукції;
- багаторічний досвід;
- висока точність роботи пристроїв та платформи;
- наявність технічної підтримки;
- оптимальне співвідношення вартості та якості обладнання.

Професіоналізм команди оцінюється за рівнем довіри клієнтів та довгостроковості партнерських відносин. У портфелі клієнтів є як державні, так і приватні компанії.

Компанія продовжує вивчати тенденції ринку, удосконалювати свої послуги та піклуватися про інтереси своїх клієнтів, акцентуючи увагу на зручності, простоті використання та довгостроковості рішень з відмінною підтримкою та постійним розвитком.

1.2 Організаційна структура компанії

Рис 1. Організаційна структура компанії

Весь апарат управління по вертикалі ділиться на ступені, що визначають послідовність підлеглості органів управління знизу догори. Кожною ланкою керує начальник, який здійснює загальне керівництво, несе відповідальність за результати праці, підпорядковується вищому керівництву і отримує розпорядження лише від нього.

1.3 Дослідження діяльності структурного підрозділу

До обов’язків фахівця технічної підтримки, як правило, входить:

- оперативна віддалена технічна підтримка користувачів;
- прийом і реєстрація заявок про несправності, обробка інформації;
- первинна допомога в установці/ налаштуванні/ усуненні несправностей і збоїв за заявками користувачів, перенаправлення заявки на 2-3 лінії технічної підтримки;
- взаємодія з програмістами, тестерами, аналітиками.

Вимоги:

- вища/ неповне вища/ середня фахова освіта (технічна);
- упевнений користувач ПК;
- знання апаратної частини ПК, знання периферії та оргтехніки;
- високі комунікативні навички;
- володіння англійською мовою на базовому рівні.

Віковий діапазон найбільш затребуваних ринком праці фахівців технічної підтримки 20-30 років. Фахівці технічної підтримки до 30 років становлять 63% від загальної кількості шукачів, від 30 до 40 років – 23%, від 40 до 50 – 13%, старше 50 років – 2%.

Робота фахівця технічної підтримки в більшості організацій передбачає повний робочий день. Рівень оплати праці даних фахівців визначається добробутом компанії, переліком посадових обов’язків, досвідом роботи за спеціальністю та рівнем розвитку професійних навичок.

1.4 Вивчення діючих комп’ютерних систем

Wialon – платформа для GPS/ГЛОНАСС моніторингу та IoT, яка дозволяє користувачам відстежувати мобільні та стаціонарні об'єкти (транспортні засоби, техніку, будівлі, мобільний персонал, домашніх тварин тощо) у 130+ країнах світу. Розробляється приватною білоруською компанією

Gurtam з головним офісом у Мінську та поширюється через мережу інтеграторів систем супутникового моніторингу.

Партнерська мережа Wialon складається із 1900+ компаній, які надають комплексні послуги супутникового моніторингу транспорту.

У листопаді 2016 року до Wialon було підключено мільйонний автомобіль, у травні 2019-го таких авто було 2 мільйони, а в липні 2020-го – 2,5 мільйона.

У 2020 році аналітичне агентство Berg Insight визнало Wialon системою №1 для моніторингу та управління транспортом у Східній Європі, країнах СНД та Росії.

Функціонал:

- відстеження місцезнаходження об'єкта та його пересування на карті;
- моніторинг таких параметрів об'єкта, як швидкість, рівень палива, температура та ін;
- керування об'єктом (виконання команд, автоматичне виконання завдань) та контроль водіїв (SMS, дзвінки, призначення);
- отримання повідомлень про активність об'єкта;
- відстеження руху об'єкта за заданим маршрутом;
- формування звітності (таблиці, графіки) з урахуванням отриманої від об'єкта інформації.

Всі дані платформа отримує від датчиків, встановлених на об'єкті моніторингу, обробляє їх та подає у вигляді звітів та графіків. Отримані дані також можна експортувати до файлів різних форматів.

Wialon поширюється у двох форматах:

- Software as a service (SaaS) – Wialon Hosting, встановлений у серверному центрі Wialon.
- Ліцензоване ПЗ – Wialon Local для встановлення на власний сервер.

Крім десктопної версії, існує мобільний додаток Wialon для iOS та Android, доступний для хмарного рішення Wialon Hosting та серверної версії платформи Wialon Local.

Для закриття специфічних галузевих завдань Gurtam розробили нішеві програми на базі Wialon:

- Hecterra – рішення для агробізнесу, яке відстежує польові роботи за допомогою телематичних даних.
- NimBus – рішення для контролю за громадським пасажирським транспортом.
- Fleetrun – рішення контролю техобслуговування.
- Logistics – рішення для організації процесу доставки.
- Eco Driving – рішення для контролю якості керування.

Для розробки рішень з урахуванням платформи розробники надають SDK. Відкритий API дозволяє інтегрувати Wialon з іншими сервісами, такими як системи обліку, наприклад.

Елементи інтерфейсу Wialon можна персоналізувати, щоб скористатися системою під своїм брендом. Міняти можна:

- колірну гаму
- шрифт інтерфейсу
- логотип та фон сторінки авторизації
- тексти (заголовок сторінки, посилання на документацію, посилання на службу технічної підтримки та ін.)

Щоб стати сертифікованим спеціалістом, необхідно пройти сертифікаційне тестування. Сертифікати бувають кількох рівнів, від першого до четвертого, залежно від відсотка правильних відповідей у тесті. Кожен рівень рекомендований для різних відділів компанії-інтегратора. Усі спроби пройти сертифікаційне тестування безкоштовні.

1.5 Проблеми, виявлені в результаті дослідження

Через те що на ринку та у системі Wialon немає готових рішень, які підходять компанії, постала проблема у розроблені інформаційної системи для роботи з приладами для зважування техніки з сільськогосподарською продукцією.

Тому виникає необхідність в створені інформаційної системи, яка буде збирати дані з системи Wialon та інтерпретувати їх у більш зручний та зрозумілий вигляд для клієнта, з подальшою їх обробкою та формуванням у звіт.

1.6 Задачі автоматизації

В результаті дослідження, було виявлено необхідність автоматизувати наступні функції:

- автоматичний збір інформації з головної системи;
- редагування інформації;
- забезпечення пошуку;
- захист даних;
- формування звітної документації.

1.7 Аналіз аналогів систем розробки

На ринку існують аналоги, які працюють окремо від Wialon. Тому ці рішення не підходять компанії, так як немає прямої інтеграції з головною системою.

Також мінусами є:

- покупка нової системи;
- покупка нового обладнання, яке працює з новою системою;
- незручність використання двох різних систем;
- навчання персоналу роботі з новою програмою;
- нагромадження непотрібними функціями.

Тому доцільніше створити систему, яка буде відповідати вимогам компанії та задовольняти її потреби.

1.8 Цілі та призначення створення інформаційної системи

Після проведеного аналізу постала мета в розробленні інформаційної системи. Система повинна забезпечувати захист, редагування та пошук даних, а також створення звіту [1].

Для досягнення цієї мети потрібно вирішити наступні завдання:

- провести аналіз роботи системи зважування;
- проаналізувати дані, які надсилає система зважування до головної програми;
- дослідити потреби компаній, для яких створюється інформаційна система, та скласти список майбутніх функцій системи;
- створити програму-додаток.

Для зручної роботи потрібно створити інтерфейс користувача. Інтерфейс повинен відповідати стандарту GUI (Graphical user interface), бути зрозумілим та зручним.

1.9 Вимоги до технічного забезпечення

Інформаційна система напряму буде зв’язана з Wialon, тому технічне забезпечення повинно відповідати вимогам Wialon.

Оснащеність і потужність комп'ютера впливають на швидкість роботи браузера і, відповідно, Wialon. Основну роль у продуктивності браузера відіграє центральний процесор та обсяг оперативної пам'яті. Для більшості браузерів немає значення кількість ядер процесора. Винятком є браузер Google Chrome, який може використовувати більше одного ядра процесора.

На підставі описаного вище, можна сформувати такі мінімальні вимоги до комп'ютера:

- центральний процесор із тактовою частотою 1,6 ГГц;
- оперативна пам'ять 512 Мб.

Рекомендовані характеристики такі:

- центральний процесор з тактовою частотою від 2,4 ГГц (при використанні браузера Google Chrome рекомендується процесор з двома ядрами і більше);
- оперативна пам'ять 2 Гб чи більше.

Також слід враховувати розмір та роздільну здатність монітора (вважається, що браузер використовується у повноекранному режимі). Чим більша роздільна здатність монітора, тим більше інформації центральний процесор запитує з сервера і обробляє. Тому можуть виникнути ситуації, коли на моніторі з діагоналлю 17 дюймів програма працює нормально, а на моніторі 22 дюйми починає «гальмувати». Можливим рішенням є перемикання браузера з повноекранного режиму до звичайного. Ця проблема особливо часто зустрічається за низької швидкості інтернет-підключення.

Антивірусні програми, які встановлені на комп'ютері та контролюють мережевий трафік, можуть сповільнити роботу браузера, а також отримання актуальних даних по об'єктах. Якщо система моніторингу повільно працює, то в налаштуваннях антивірусної програми можна додати Wialon у виключення, щоб мережевий трафік системи моніторингу не перевірявся, або просто відключити мережевий моніторинг антивірусної програми на час використання Wialon. Також можна створити правило, яке дозволяє системі Wialon будь-яку мережну активність.

Для нормальної роботи Wialon на одному комп'ютері достатньо 1-мегабітного каналу підключення до Інтернету. Якщо у системі моніторингу одночасно буде кілька операторів, то визначити відповідну швидкість можна на підставі суб'єктивних тестів.

1.10 Вимоги до функцій, покладених на інформаційну систему

Інформаційна система повинна забезпечувати такі функції:

- редагування даних;
- пошук та фільтрація даних;

- розрахунок нетто ваги;
- формування звіту.

1.11 Вхідні та вихідні дані інформаційної системи

Вхідні данні системи:

- вага техніки;
- назва техніки;
- логін та пароль користувача;
- брутто вага;
- номер RFID картки.

Вихідними даними буде звітня інформація.

1.12 Вимоги до розробки інтерфейсу

Інтерфейс користувацького застосунку має відповідати наступним вимогам:

- зрозумілість - користувач має легко розбиратися в деталях керування навіть без попереднього навчання.
- відповідність - всі елементи інтерфейсу мають реалізовувати саме ті функції, які відображають.
- достатність - кожна дія повинна виконуватися повністю, максимально забезпечуючи користувача інформацією та даючи достатню інформацію.
- простота - інтерфейс не має бути перевантажений зайвими кнопками, полями або прикрашальними частинами.
- нормативність - інтерфейс повинен відповідати міжнародним нормативам gui [3].

1.13 Вимоги до збереження інформації в разі аварії

Так як вся інформація буде зберігатися на серверах Wialon, інформаційна система не потребує додаткових заходів щодо збереження інформації [2].

2 РОЗДІЛ

2.1 Методи вирішення поставленої задачі

Для розв’язання цієї задачі було вирішено розробити сайт за допомогою HTML, CSS, JS. Також було використано API, який запропоновано розробниками Wialon. Для зручної роботи з HTML, CSS та JS було використано програму Brackets.

2.2 HTML

HTML (від англ. HyperText Markup Language – «мова гіпертекстової розмітки») - стандартизована мова розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою HTML різні конструкції, зображення та інші об'єкти, такі як інтерактивна веб-форма, можуть бути вбудовані в сторінку, що відображається. HTML надає засоби для створення заголовків, абзаців, списків, посилань, цитат та інших елементів. Елементи HTML виділяються тегами, записаними з використанням кутових дужок. Такі теги, як <img/> та <input/>, безпосередньо вводять контент на сторінку. Інші теги, такі як <p>, оточують і оформляють текст у собі і можуть включати інші теги в якості піделементів. Браузери не відображають HTML-теги, але використовують їх для інтерпретації вмісту сторінки.

Мова XHTML є суворішим варіантом HTML, він слідує синтаксису XML і є додатком мови XML в області розмітки гіпертексту.

У HTML можна вбудувати програмний код мовою програмування JavaScript, для управління поведінкою та змістом веб-сторінок. Також включення CSS в HTML описує зовнішній вигляд та макет сторінки.

Мова гіпертекстової розмітки HTML була розроблена британським ученим Тімом Бернерсом-Лі приблизно в 1986-1991 роках у стінах ЦЕРНу в Женеві у Швейцарії [4]. HTML створювався як мова для обміну науковою та технічною документацією, придатний для використання людьми, які не є фахівцями в галузі верстки. HTML успішно справлявся із проблемою складності SGML шляхом визначення невеликого набору структурних та семантичних елементів – дескрипторів. Дескриптори також часто називають «тегами». За допомогою HTML можна легко створити відносно простий, але красиво оформлений документ. Крім спрощення структури документа, HTML внесена підтримка гіпертексту. Мультимедійні можливості було додано пізніше.

Першим загальнодоступним описом HTML був документ «Теги HTML», вперше згаданий в Інтернеті Тімом Бернерсом-Лі наприкінці 1991 [5] [6]. У ньому описуються 18 елементів, що становлять початковий, відносно простий дизайн HTML. За винятком тега гіперпосилання, на них сильно вплинув SGMLguid, внутрішній формат документації, заснований на стандартній узагальненій мові розмітки (SGML), у CERN. Одинадцять із цих елементів усе ще існують у HTML 4 [7].

Спочатку мова HTML була задумана і створена як засіб структурування та форматування документів без їх прив'язки до засобів відтворення (відображення). В ідеалі, текст з розміткою HTML повинен був без стилістичних та структурних спотворень відтворюватися на обладнанні з різним технічним оснащенням (кольоровий екран сучасного комп'ютера, монохромний екран органайзера, обмежений за розмірами екран мобільного телефону або пристрою та програми голосового відтворення текстів). Однак сучасне застосування HTML дуже далеке від його початкового завдання. Наприклад, тег <table> призначений для створення в документах таблиць, але іноді використовується для оформлення розміщення елементів на сторінці. З часом основна ідея платформонезалежності мови HTML була принесена в жертву сучасним потребам у мультимедійному та графічному оформленні.

Текстові документи, що містять розмітку на мові HTML (такі документи традиційно мають розширення .html або .htm), обробляються спеціальними програмами, які відображають документ у форматованому вигляді. Такі програми, які називаються «браузерами» або «браузерами Інтернету», зазвичай надають користувачеві зручний інтерфейс для запиту веб-сторінок, їх перегляду (і виведення на інші зовнішні пристрої) і, при необхідності, відправки введених користувачем даних на сервер. Найбільш популярними на сьогоднішній день браузерами є Google Chrome, Mozilla Firefox, Opera, Internet Explorer та Safari.

Офіційної специфікації HTML 1.0 немає. До 1995 року існувало безліч неофіційних стандартів HTML. Щоб стандартна версія відрізнялася від них, їй одразу надали другий номер.

Версія 3 була запропонована Консорціумом Всесвітньої павутини (W3C) у березні 1995 року та забезпечувала багато нових можливостей, таких як створення таблиць, «обтікання» зображень текстом та відображення складних математичних формул, підтримка gif формату. Навіть при тому, що цей стандарт був сумісний з другою версією, його реалізація була складна для браузерів того часу. Версія 3.1 офіційно ніколи не пропонувалася, і наступною версією стандарту HTML стала 3.2, в якій було опущено багато нововведень версії 3.0, але додано нестандартні елементи, що підтримуються браузерами Netscape Navigztor і Mosaic.

У версії HTML 4.0 відбулося деяке «очищення» стандарту. Багато елементів були відзначені як застарілі та не рекомендовані (англ. deprecated). Зокрема, тег <font>, який використовується для зміни властивостей шрифту, був позначений як застарілий (замість нього рекомендується використовувати таблиці стилів CSS).

У 1998 році Консорціум Всесвітньої павутини розпочав роботу над новою мовою розмітки, заснованою на HTML 4, але відповідним синтаксису XML. Згодом нова мова отримала назву XHTML. Перша версія XHTML 1.0

схвалена як Рекомендація консорціуму Всесвітнього павутиння 26 січня 2000 року.

Запланована версія XHTML 2.0 мала розірвати сумісність зі старими версіями HTML і XHTML, але 2 липня 2009 року Консорціум Всесвітньої павутини оголосив, що повноваження робочої групи XHTML2 закінчуються наприкінці 2009 року. Таким чином, було припинено всю подальшу розробку стандарту XHTML 2.0 [8].

В даний час Консорціум Всесвітньої мережі розробив HTML версії 5. Чорновий варіант специфікації мови з'явився в Інтернеті 20 листопада 2007 року.

Спільнотою WHATWG (англ. Web Hypertext Application Technology Working Group), починаючи з 2004 [4], розробляється специфікація Web Applications 1.0, часто неофіційно звана «HTML 5», яка розширює HTML (втім, маючи і сумісний з XHTML 1.0 XML-синтаксис) для кращого представлення семантики різних типових сторінок, наприклад, форумів, сайтів аукціонів, пошукових систем, онлайн-магазинів і т. д., які не дуже вдало вписуються в модель XHTML 2.0.

HTML - тегова мова розмітки документів. Будь-який документ на мові HTML є набором елементів, причому початок і кінець кожного елемента позначається спеціальними позначками - тегами. Елементи можуть бути порожніми, тобто не містять жодного тексту та інших даних. У цьому випадку зазвичай не вказується тег, що закриває (наприклад, тег перенесення рядка
 — одиночний і закривати його не потрібно). Крім того, елементи можуть мати атрибути, що визначають будь-які їх властивості (наприклад, атрибут href=" у посилання). Атрибути вказуються у тезі, що відкриває. Ось приклади фрагментів HTML-документу:

- Текст між двома тегами — відкриває та закриває.
- Тут елемент містить атрибут href, тобто гіперпосилання.
- а ось приклад порожнього елемента:

Регістр, у якому набране ім'я елемента та імена атрибутів, HTML значення не має (на відміну від XHTML). Елементи можуть бути вкладені.

Окрім елементів, у HTML-документах є й сутність (англ. entities) — «спеціальні символи». Сутність починається з символу амперсанда і має вигляд &ім'я; або &#NNNN;, де NNNN - код символу в Юнікоді в десятковій системі числення.

Наприклад, © - Знак авторського права (©). Як правило, сутності використовуються для представлення символів, відсутніх у кодуванні документа, або для представлення спеціальних символів: & - Амперсанда (&), < — символи «менше» (<) та > — символу «більше» (>), які некоректно записувати «звичайним» чином, через їхнє особливе значення в HTML.

У середині 1990-х років основні виробники браузерів - компанії Netscape і Microsoft - почали впроваджувати власні набори елементів HTML-розмітку. Створилася плутанина з різних конструкцій для роботи у Всесвітній мережі, доступних для перегляду то в одному, то в іншому браузері. Особливо великі труднощі були під час створення крос-браузерних програм мовою JavaScript. Веб-майстрам доводилося створювати кілька варіантів сторінок або вдаватися до інших хитрощів. На якийсь час проблема втратила актуальність із двох причин:

- через витіснення браузером Internet Explorer решти браузерів. Відповідно, проблема веб-майстрів ставала проблемою користувачів альтернативних браузерів.
- завдяки зусиллям виробників інших браузерів, які або дотримувалися стандартів W3C (як Mozilla та Opera), або намагалися створити максимальну сумісність із Internet Explorer.

На сучасному етапі можна констатувати зростання популярності браузерів, наступних рекомендаціям W3C (це Mozilla Firefox та інші браузери на движку Gecko; Safari, Google Chrome, Opera та інші браузери на движку WebKit). Частка Internet Explorer на січень 2016 становить менше 15% [5].

У сучасній практиці існує можливість спростити розробку крос-браузерних програм мовою JavaScript за допомогою різноманітних бібліотек та фреймворків. Наприклад, таких як jQuery, sIFR та Sin.

2.3 CSS

CSS (англ. Cascading Style Sheets, укр. Каскадні таблиці стилів) — це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних.

CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript [9].

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації.

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.).

Один і той самий HTML або XML документ може бути відображений по-різному залежно від використаного CSS. Стилі для відображення сторінки можуть бути:

- стилі автора (інформація надана автором сторінки):
    1. Зовнішні таблиці стилів (англ. stylesheet), найчастіше окремий файл або файли .css;
    2. Внутрішні таблиці стилів, включені як частина документу або блоку;
    3. Стилі для окремого елементу.
- стилі користувача:
    1. Локальний .css-файл, вказаний користувачем для використання на сторінках і вказаний в налаштуваннях браузера (наприклад Opera).
- стилі переглядача (браузера):
    1. Стандартний стиль переглядача, наприклад стандартні стилі для елементів, визначені браузером, використовуються коли немає інформації про стиль елемента або вона неповна.

Стандарт CSS визначає порядок та діапазон застосування стилів, тобто, в якій послідовності і для яких елементів застосовуються стилі. Таким чином, використовується принцип каскадності, коли для елементів вказується лише та інформація про стилі, що змінилася або не визначена загальнішими стилями.

Переваги:

- інформація про стиль для усього сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;
- різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;
- сторінки зменшуються в об'ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з урахуванням їх;
- прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам'ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

CSS має порівняно простий синтаксис і використовує небагато англійських слів для найменування різних складових стилю.

Стилі складаються зі списку правил. Кожне правило має один або більше селектор (англ. selector) та блок визначення (англ. declaration block). Блок визначення складається з оточеного фігурними дужками списку властивостей.

Властивості в списку оформлюються у вигляді назва властивості, двокрапка (:), значення, крапка з комою (;).

Приклад:

```css
p {
  font-family: Verdana, sans-serif;
}
h2 {
  font-size: 110%;
  color: red;
  background: white;
}
.note {
  color: red;
  background: yellow;
  font-weight: bold;
}
p.warning {
  background: url(warning.png) no-repeat fixed top;
}
#paragraph1 {
  margin: 0;
}
a:hover {
  text-decoration: none;
}
#news p {
  color: red;
}
```

В прикладі використано 7 правил, селектори p, h2, .note, p.warning, #paragraph1, a:hover та #news p. Приклад властивості: color: red, де властивості з назвою color присвоєно значення red.

В перших двох правилах визначаються властивості HTML-елементів p (абзац, скорочення від англ. paragraph) та h2 (Заголовок другого рівня, скорочення від англ. header). Абзац буде відображено шрифтом Verdana або, у разі відсутності Verdana, іншим шрифтом sans-serif. Заголовок другого рівня буде відображено червоним кольором на білому тлі.

Третє правило буде застосовано до всіх елементів, властивість class визначена як 'note'. Наприклад:

Абзац буде відображено червоним жирним шрифтом на жовтому тлі.

Третє правило також можна записати як *.note.

Щоб третє правило застосовувалося лише до абзаців, його слід переписати як:

```css
p.note {
  color: red;
  background: yellow;
  font-weight: bold;
}
```

Четверте правило стосується тих елементів рівня абзацу (p), що мають властивість class рівну 'warning'. Такі абзаци матимуть фонове зображення warning.png згори.

Властивість .class може мати не лише порівняння, але й перелік. Коли властивість class містить перелік значень, розділених пробілом, то до цього елементу застосовуються правила усіх перелічених класів. Наприклад, до <p class="note warning"> будуть застосовані правила note та warning.

П'яте правило застосовується лише до одного елементу в документі HTML, що має ідентифікатор id рівний paragraph1. Цей елемент не буде мати межі навколо, тому що властивість margin дорівнює 0. Наприклад:

```
<p id="paragraph1">Цей абзац не має межі тому що межа
 дорівнює 0. На сторінці може бути лише один такий елемент.
 </p>
```

Шосте правило визначає стиль наведення мишкою (hover) для a (anchor) елементів. Стандартно більшість браузерів підкреслюють елементи a (посилання в межах одного документа або на інші HTML-документи). Це правило прибирає підкреслювання з посилань, коли користувач наводить на них мишкою.

Останнє правило застосовується до тих абзаців, що знаходяться всередині (мають батьківський елемент) з ідентифікатором news. Це приклад наслідування властивостей.

Завдяки каскадній структурі CSS, ці абзаци (p) будуть наслідувати font-family: Verdana, sans-serif; як і всі елементи p (абзаци). Ті з них, що мають class рівний warning будуть мати також і фонове зображення.

CSS може також містити коментарі. Синтаксис коментарів подібний до синтаксису, що використовується в багатьох мовах програмування (наприклад C, PHP). Слід звернути увагу на те, щоб в коментарях CSS-файлів не використовувалися кириличні літери (деякі браузери некоректно обробляють CSS-файли з українськими літерами [10]).

CSS має спеціальні позначення (селектори, від англ. selector) для визначення діапазону застосування правила.

Для точнішого визначення стилю можуть використовуватися псевдо-класи (англ. pseudo-classes). Мабуть найвідомішим є псевдо-клас :hover, що застосовується коли користувач вкаже на елемент, зазвичай наведенням курсору. Стиль визначається для a:hover або #elementid:hover. Інші псевдо-класи, наприклад, :first-line (перший рядок тексту), :first-letter (перша літера) :visited (посилання, що вже відкривалось) або :before (визначає стиль перед елементом). Спеціальний псевдо-клас :lang(c) буде застосовано до елементів якщо їхня мова (англ. language) «c».

Підтримка псевдо-класів залежить від браузеру, наприклад Internet Explorer 6 підтримує псевдо-клас :hover лише для посилань.

Селектори можуть поєднуватися й іншим чином для досягнення більшої гнучкості (див. визначення селекторів на сайті W3C). В наступному прикладі кілька селекторів об'єднуються через кому [11]. Правило встановлює шрифт для HTML-заголовків всіх рівнів.

```
h1, h2, h3, h4, h5, h6 {
  font-family: "Arial", sans-serif;
}
```

Щоб включити таблицю стилів CSS, потрібно зберегти код CSS у файл (це можна зробити будь-яким текстовим редактором), наприклад example.css, і потім включити або імпортувати його в HTML або XHTML-сторінку.

Включення CSS-файлу до сторінки (XHTML):

```
<link rel="stylesheet" href="example.css"
type="text/css" />
```

Включення CSS-файлу до сторінки (HTML):

```
<link rel="stylesheet" href="example.css"
type="text/css">
```

Імпорт CSS-файлів до HTML та XHTML-сторінок:

```
<style type="text/css">
@import "example.css";
</style>
```

CSS може бути визначений в <head> частині сторінки або для елементу сторінки через style.

Включення CSS-файлу до XML-сторінки:

```
<?xml-stylesheet type="text/css" href="example.css"?>
```

Таблиці стилів в тому чи іншому вигляді існували з зародження SGML в 1970-тих. Каскадні таблиці стилів розроблялися для спрощення процедури додавання інформації про стилі для веб-сторінок.

З розвитком HTML з'явилася можливість використовувати різні механізми для додавання стилів до елементів сторінки. Еволюція HTML дала веб-дизайнерам більше можливостей для створення вигляду сайту, але HTML-код ставав складнішим для написання та зміни. Через різницю у відображення сторінки в різних браузерах збереження стилю сторінки було складним, користувачі мали менше контролю над відображенням контенту.

На розгляд W3C було запропоновано дев'ять різних варіантів таблиць стилів. Після обговорення в спеціальному списку розсилки було обрано два, вони створили основу для того, що стало CSS: англ. Cascading HTML Style Sheets (CHSS) та англ. Stream-based Style Sheet Proposal (SSP). Спочатку в жовтні 1994, Хокон Віум Лі (зараз генеральний технічний директор Opera Software) запропонував Cascading HTML Style Sheets (CHSS), що дещо подібний до сучасного CSS. Bert Bos працював над браузером Argo, що використовував власний варіант таблиць стилів, Stream-based Style Sheet Proposal (SSP). Lie і Bos почали співпрацювати для вироблення стандарту CSS (літера 'H' була виключена з назви, оскільки таблиці стилів могли застосовуватися до інших мов розмітки, не лише HTML).

На відміну від наявних тоді таблиць стилів, таких як DSSSL та FOSI, CSS дозволяв застосування різних таблиць стилів до документу (сторінки). Таблиці стилів могли наслідувати правила з інших (тобто створювати каскади), що

дозволило контролювати використання стилів як дизайнером сайту так і користувачем (наприклад в браузері Opera).

Пропозиції до стандарту CSS обговорювалися на конференціях в 1994 та 1995 роках.

В 1994 році було створено World Wide Web Consortium W3C, серед інших питань W3C займався також і CSS. Робочу групу очолив Steven Pemberton, як провідні технічні спеціалісти до неї входили Хокон Віум Лі та Bert Bos.

В грудні 1996 було опубліковано CSS рівня 1 — CSS level 1 Recommendation. В квітні 2016, опубліковано CSS рівня 2 — Cascading Style Sheets Level 2 Revision 2 (CSS 2.2), робота над яким ще продовжується.

2.4 JavaScript

JavaScript (JS) — динамічна, об'єктно-орієнтована [12] прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

- написання сценаріїв вебсторінок для надання їм інтерактивності;
- створення односторінкових та прогресивних вебзастосунків (React, AngularJS, Vue.js);

- програмування на боці сервера (Node.js(Express.js));
- стаціонарних застосунків (Electron, NW.js);
- мобільних застосунків (React Native, Cordova);
- сценаріїв в прикладних програмах (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
- всередині PDF-документів тощо.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme. [13]

1995 року компанія Netscape поставила завдання вбудувати мову програмування Scheme чи «якусь схожу» в браузер Netscape. Для цього був запрошений Брендан Айк, американський розробник, що спеціалізувався на системному програмуванні. Також, для прискорення розробки, Netscape почали співробітництво з компанією Sun Microsystems.

З часом, концепція розроблюваної мови програмування була розширена до можливості використання безпосередньо в HTML-коді сторінки. Компанії мали на меті створити мову, що могла зв'язати різні частини вебсайтів: зображень, Java-аплетів, об'єктної моделі документа. Ця мова повинна була стати зручною для вебдизайнерів та некваліфікованих програмістів. Робочою назвою нової мови була Mocha, яка була змінена на LiveScript в перших двох бета-версіях браузера Netscape 2.0. А дещо пізніше, користуючись популярністю бренду Java, LiveScript був перейменований на JavaScript і третя бета-версія (2.0B3) Netscape 2.0 вже вийшла з сучасною назвою [14] [15]. Для цього була придбана відповідна ліцензія у компанії Sun Microsystems, що володіла брендом Java.

1992 року компанією Nombas була розроблена скриптова мова програмування Cmm (англ. C-minus-minus, гра слів навколо мови C++), яка пізніше була перейменована на ScriptEase та могла вбудовуватися в

вебсторінки. Існує хибна думка, що JavaScript створено під впливом Cmm. Насправді Брендан Айк ніколи не чув про Cmm до того, як він створив LiveScript [12]. Пізніше, Nombas зупинили розробку Cmm та почали використовувати JavaScript, а згодом брали участь у групі зі стандартизації JavaScript.

У листопаді 1996 року Netscape заявила, що відправила JavaScript в організацію Ecma International для розгляду мови як промислового стандарту. В результаті подальшої роботи з'явилась стандартизована мова з назвою ECMAScript. У червні 1997 року, Ecma International опублікувала першу редакцію специфікації ECMA-262. Рік по тому, у червні 1998 року, щоб адаптувати специфікацію до стандарту ISO/IEC-16262, були внесені деякі зміни і випущена друга редакція. Третя редакція побачила світ в грудні 1999 року. [13]

Четверта версія стандарту ECMAScript так і не була закінчена і четверта редакція не вийшла [14]. Тим не менш, п'ята редакція з'явилася в грудні 2009 року.

У червні 2015 року [15] вийшла шоста версія, починаючи з якої комітет ECMAScript прийняв рішення перейти на щорічні оновлення і нова версія отримала назву ES2015. Вона отримала цілу низку нововведень, серед яких: об'єкт Promise для зручного асинхронного виконування коду, деструктуруюче присвоювання, стрілочні функції, функції-генератори, шаблонні рядки, оператори оголошення змінних let та const тощо.

Версія ES2016 вийшла у червні 2016 року[12], серед нововведень оператор піднесення до степеня ** та метод Array.prototype.includes, який перевіряє, чи міститься переданий аргумент в масиві.

Версія ES2017, що вийшла в червні 2017 року[13], додала можливість використання асихронних функцій, «висячих» ком в параметрах функцій, об'єкт Atomics, декількох нових методів для роботи з рядками.

Версія ES2018 вийшла у червні 2018 року[14], додала можливість здійснювати асинхронні ітерації, оператор Spread (...) для роботи з об'єктами

та масивами, декілька нових можливостей для регулярних виразів, метод Promise.prototype.finally, який спрацьовує по отриманню Promise'ом статусу "виконаний".

Версія ES2019 вийшла у червні 2019 року[15], серед нововведень: новий тип даних Symbol, нові методи для роботи з рядками та масивами, перетворення об'єктів в масиви і навпаки за допомогою Entries.

Актуальною на даний момент є версія ES2020, що вийшла у червні 2020 року [12]. Вона додала до мови новий тип даних BigInt, оператор ?? для перевірки на null та undefined, можливість використання опціональних значень в об'єкті, динамічні імпорти, об'єкт globalThis, методи String.prototype.matchAll для пошуку у рядку за допомогою регулярних виразів та Promise.allSettled для спрацювання після виконання усіх Promise'ів.

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті [13]. В перші роки існування, більшість професійних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-аматорів [14]. Поява AJAX змінила ситуацію та звернула увагу професійної спільноти до мови, а її подальші модифікації за стандартами ES6+ внесли багато корисних можливостей, яких не вистачало для ефективного програмування. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має кілька властивостей, притаманних функціональним мовам, — функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання (closures) — що додає мові додаткову гнучкість.

JavaScript має C-подібний синтаксис, але в порівнянні з мовою C має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;
- функції як об'єкти першого класу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичне збирання сміття;
- анонімні та стрілочні функції.

JavaScript містить декілька десятків вбудованих об'єктів[15], які поділяються на групи: фундаментальні (Object, Function, Boolean, Symbol), помилки (група об'єктів Error), числа та дати (Number, BigInt, Math Date), текстові (String, RegExp), індексовані (група об'єктів Array), ключові (Map, Set, WeakMap, WeakSet), для роботи з структурованими даними (ArrayBuffer, Atomics, DataView, JSON), абстрактні (Promise, Generator), рефлекційні (Reflect, Proxy), групи Intl та WebAssembly. Крім того, JavaScript містить набір вбудованих операцій, що керують логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від C), але спрощений у порівнянні з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, наприклад, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може знаходитися в тексті програми після неї.

Семантика мови схожа з мовою Self.

Приклад оголошення і використання класу в JavaScript:

```
class MyClass {
  constructor() {
    this.myValue1 = 1;
    this.myValue2 = 2;
  }
}

const mc = new MyClass();
mc.myValue1 = mc.myValue2 * 2;
```

Одна з популярних технологій, що дозволила зробити сторінки динамічнішими і забезпечити нові можливості — це динамічне завантаження і вставка даних в документ, що отримала назву AJAX.

При використанні в рамках технології DHTML JavaScript код включається в HTML-код сторінки і виконується інтерпретатором, вбудованим в браузер. Код JavaScript вставляється в теги <script></script> , хоча в більшості браузерів мова сценаріїв за умовчанням саме JavaScript.

Скрипт, що виводить модальне вікно з класичним написом «Hello, World!» усередині браузера:

```
<script>
  alert("Hello, World!");
</script>
```

Є ще одна можливість підключення JavaScript — написати скрипт окремим файлом, та підключити його за допомогою конструкції:

```
<script src="шлях/до/файлу/зі/скриптом.js"></script>
```

Браузери, які дотримуються концепції інтеграції JavaScript в існуючі системи, підтримують включення скрипту, наприклад, у значення атрибуту події:

```
 <a href="delete.php" onclick="return confirm('Ви
впевнені?');">Видалити</a>
```

Після натискання на посилання, функція confirm() викликає модальне вікно з написом «Ви впевнені?», а return false блокує перехід за посиланням. Цей код працюватиме тільки якщо в браузері вбудована та ввімкнена підтримка JavaScript, інакше перехід за посиланням відбудеться без попередження.

Атрибути async і defer використовуються для ввімкнення асинхронного порядку завантаження скриптів.

Підтримується всіма браузерами, крім IE9-. Скрипт виконується асинхронно. Тобто, елемент <script async src = "..."> , виконується в момент його отримання браузером.

Підтримується всіма браузерами, включно з найстаршими версіями IE. Скрипт також виконується асинхронно, не змушує чекати сторінку, але є дві відмінності від async.

Перша — браузер гарантує, що відносний порядок скриптів з defer буде збережений.

Тобто, в такому коді (з async) першим працюватиме той скрипт, котрий швидше завантажиться.

```
<script src="1.js" async></script>
<script src="2.js" async></script>
```

А в такому коді (з defer) першим спрацює завжди 1.js, а скрипт 2.js, навіть якщо завантажився раніше, буде його чекати.

```
<script src="1.js" defer></script>
<script src="2.js" defer></script>
```

Тому атрибут deferвикористовують в тих випадках, коли другий скрипт 2.js залежить від першого 1.js, наприклад — використовує щось, описане першим скриптом.

Друга відмінність — скрипт з defer спрацює, коли весь HTML-документ буде оброблений браузером.

Наприклад, якщо документ досить великий…

```
<script src="async.js" async></script>
<script src="defer.js" defer></script>
```

…то скрипт async.js виконається, щойно завантажиться, можливо, до того, як весь документ готовий. А defer.js почекає готовності всього документа.

Це буває зручно, коли ми в скрипті хочемо працювати з документом, і повинні бути впевнені, що він цілком отриманий.

Оскільки JavaScript є інтерпретованою мовою програмування, без строгої типізації, і може виконуватися в різних середовищах, кожне зі своїми власними особливостями сумісності, програміст має бути уважним, і повинен перевіряти, що його код виконується як очікується в широкому переліку

можливих конфігурацій. Типізація вважається одною з ключових проблем JavaScript, тому восени 2012 року, компанія Microsoft презентувала мову програмування TypeScript, що компілюється в JavaScript та містить декілька важливих для програмістів доповнень, що полегшують розробку.

При розробці великих і нетривіальних вебзастосунків з використанням JavaScript, критично важливим є доступ до інструментів відлагодження. Оскільки браузери, від різних виробників, дещо відрізняються у поведінці JavaScript і реалізації Об'єктної моделі документа, необхідно мати відлагоджувач для кожного браузера, якщо вебзастосунок орієнтовано на нього.

На даний час Firefox, Opera, Google Chrome, Edge та Safari мають зневаджувачі для себе.

Також існують такі корисні інструменти, як:

- ESLint — перевірка якості коду, що сканує JavaScript-програму, шукаючи вади у коді;
- Prettier — автоматичне форматування коду у коректний вигляд;
- Babel — компілятор JavaScript-коду до старіших версій стандарту ECMAScript, який допомагає розробникам використовувати найновіші можливості мови для оточення, що не встигло реалізувати останній стандарт.

Кожен блок сценарію інтерпретатор розбирає окремо. На вебсторінках, коли треба комбінувати блоки JavaScript та HTML, синтаксичні помилки знайти легше, якщо зберігати функції сценарію в окремому блоці коду, або (ще краще) використовувати багато малих, пов'язаних між собою .js файлів. Таким чином, синтаксична помилка не спричинятиме «падіння» всієї вебсторінки та дозволить сповістити користувача про проблему.

2.5 API

Прикладний програмний інтерфейс (інтерфейс програмування застосунків, інтерфейс прикладного програмування) (англ. Application Programming Interface, API) — набір визначень підпрограм, протоколів

взаємодії та засобів для створення програмного забезпечення. [16] Спрощено — це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані. Програмісти використовують переваги API у функціональності, таким чином їм не доводиться розробляти все з нуля. API є абстрактним поняттям — програмне забезпечення, що пропонує деякий API, часто називають реалізацією (англ. implementation) даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані. [17]

Високорівневі API часто програють у гнучкості. Виконання деяких функцій нижчого рівня стає набагато складнішим, або навіть неможливим.

Наприклад, в мові Java, якщо програміст хоче використовувати клас «Scanner» (клас, який зчитує інформацію від користувача у програмах, орієнтованих на текстові операції), він імпортує бібліотеку «java.util.Scanner», щоб використовувати методи класу «Scanner» (у даному прикладі nextLine() і close()). Це приклад з API, що дозволяє взаємодіяти з бібліотеками в мові Java.

```
import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        System.out.println("Enter your name: ");
        Scanner input = new Scanner(System.in);
        String name = input.nextLine();
        System.out.println("Your name is " + name + ".");
                input.close();
        }

}
```

В об'єктноорієнтованих мовах прикладний програмний інтерфейс зазвичай включає в себе опис набору визначень класу, з набором форм поведінки, пов'язаних з цими класами. Це абстрактне поняття пов'язане з реальними функціями, які надані або надаватимуться, класами, які реалізуються в методах класу.

Прикладний програмний інтерфейс в цьому випадку можна розглядати як сукупність всіх методів, які публічно доступні в класах (зазвичай званий інтерфейс класу). Це означає, що прикладний програмний інтерфейс вказує методи, за допомогою яких взаємодіє з об'єктами, отриманими з визначень класів і обробляє їх.

У більш загальному плані можна визначити Прикладний Програмний Інтерфейс як сукупність усіх видів об'єктів, які можна вивести з визначення класу, і пов'язаних з ними можливих варіантів поведінки.

Наприклад: клас, що представляє Stack, може просто виставити публічно два методи Push() (для додавання нового елемента в стек) і Pop() (для вилучення останнього пункту, ідеально розташований на вершині стека).

У цьому випадку Прикладний Програмний Інтерфейс може бути інтерпретованим як два методи pop() і push(), або, більш широко, використовується варіант, коли можна використовувати елемент типу Stack, який реалізує поведінку стека, надаючи йому можливість для додавання / видалення елементів з вершини. Друга інтерпретація видається більш доречною в дусі об'єктноорієнтованого підходу.

Якість документації, пов'язаної з Прикладним Програмним Інтерфейсом, є часто ключовим фактором, що визначає його успішність з точки зору простоти використання.

ППІ, як правило, пов'язаний із бібліотеками програмного забезпечення: ППІ описує і вказує очікувану поведінку в той час, як бібліотека є фактичною реалізацією даного набору правил. Один ППІ може мати декілька реалізацій (або жодної, будучи абстрактним) у вигляді різних бібліотек, які мають такий же інтерфейс.

Прикладний програмний інтерфейс також може бути пов'язаним з платформами програмування: платформа може бути заснована на кількох бібліотеках реалізує декілька інтерфейсів ППІ, але на відміну від звичайного використання ППІ, доступ до поведінки вбудований в платформу опосередкований шляхом розширення його змісту новими класами і вставлений в саму платформу. Крім того, загальний потік управління програми може бути під контролем абонента.

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

Коли ППІ реалізує протокол, він може бути заснованим на проксі-методах віддалених викликів, що засновані на протоколі зв'язку. Роль ППІ може полягати саме в тому, щоб приховати деталі транспортного протоколу. Наприклад: RMI є ППІ, який реалізує протокол або JRMP IIOP як RMI-IIOP.

Протоколи, як правило, розподіляються між різними технологіями і зазвичай дозволяють різним технологіям обмінюватися інформацією, діючи як абстракція між двома світами. ППІ, як правило, є специфічним для конкретної технології: звідси, інтерфейси даної мови не можуть бути використані на інших мовах, якщо виклики функції не будуть перетворені з конкретної адаптації бібліотеки.

Деякі мови, серед яких такі, що працюють на віртуальних машинах (наприклад: мови, сумісні з NET CLI середовища CLR і JVM сумісних мов у віртуальній машині Java) можуть ділитися програмними інтерфейсами.

У цьому випадку віртуальна машина дозволяє мові взаємодії завдяки спільному знаменнику віртуальної машини, що абстрагується від конкретної мови, використовувати проміжний байт-код і його мову.

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, ППІ визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об'єктного запису JavaScript (JSON). У той час як прикладний програмний інтерфейс у Web історично був практично

синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсоорієнтованої архітектури (ROA).[16] Частина цієї тенденції пов'язана з рухом Семантичного веб-ресурсу до Опису Платформ (RDF), Концепції розвитку веб-технологій інженерних онтологій. [17] Прикладні програмні інтерфейси у Web, що дозволяють комбінувати декількома прикладними програмними інтерфейсами в нові додатки називають гібридними. [16]

Існує два основних варіанти впровадження прикладного програмного інтерфейсу:

- захист інформації про програмний інтерфейс від широкого загалу. Наприклад, компанія Sony дозволила розробляти програмний інтерфейс для PlayStation 2 лише ліцензованим розробникам. Це дозволило Sony контролювати, хто розробляв ігри для PlayStation 2. Такий варіант дозволяє компаніям переважаючий контроль якості за випуском продукції, і також надає можливості для додаткового доходу від ліцензування.
- розробка програмного інтерфейсу існує також у вільному доступі. Наприклад, компанія Microsoft робить програмний інтерфейс до Microsoft Windows загально доступним, а компанія Apple, своєю чергою, впроваджує прикладні програмні інтерфейси Carbon та Cocoa, для того, щоб дозволити писати програмне забезпечення під свої платформи.

У 2010 році Oracle подала до суду на Google, за поширення нової версії Java, вбудованої у нову версію ОС Android без дозволу на використання JavaAPI, хоча аналогічний договір був наданий на використання проєкту OpenJDK. Суддя виніс рішення у справі Oracle проти Google, про те, що даний програмний інтерфейс не може бути захищеним авторськими правами у США.[17]

2.6 Brackets

Brackets — текстовий редактор від компанії Adobe, призначений для редагування JavaScript, HTML і CSS. Сирцевий код Brackets написаний з використанням веб-технологій (JavaScript, HTML і CSS) і поширюється під ліцензією MIT. Редактор оформлений у вигляді відокремленого настільного застосунка, для установки якого підготовлені deb-, dmg- і msi- пакети для Linux, OS X і Windows.

Brackets підтримує режим Live-розробки, при якому редагований контент (JavaScript, HTML і CSS) у міру зміни відразу відображається в синхронізованому з редактором вікні браузера — розробник може змінювати вміст і відразу спостерігати до яких наслідків приводять дані зміни. Налагодження також може виконуватися синхронно із браузером, розробник може встановити точку зупину або відкотитися на крок назад при перегляді результатів. Є вбудована підтримка препроцесорів LESS і SCSS. В інтерфейсі застосовується система контекстно-залежних інструментів, що з'являються в міру необхідності в основному вікні розробки. Для розширення можливостей редактора розвивається система доповнень.

2.7 Створення інтерфейсу користувача

Для створення системи було використано JavaScript API, який дозволяє отримати доступ до даних та функцій Wialon у власному веб-додатку за допомогою JavaScript.

Для реалізації входу до системи було використано наступний JavaScript код, який запропоновано розробниками, як прикладний програмний інтерфейс:

```
// Print message to log
function msg(text) { $("#log").prepend(text + "<br/>"); }

// Login to server using entered username and password
function login() {
    var sess = wialon.core.Session.getInstance(); // get instance of current Session
    var user = sess.getCurrUser(); // get current User
    if( user ) { // if user exists - you are already logged, print username to log
        msg("You are logged as '" + user.getName()+"', click logout button first");
        return;
    }

    // if not logged
    var token = $("#token").val(); // get token from input
    if (!token) { // if token is empty - print message to log
        msg("Enter token");
        return;
    }

    msg("Trying to login with token '"+ token +"'");
    sess.initSession("https://hst-api.wialon.com"); // initialize Wialon session
    sess.loginToken(token, "", // trying login
        function (code) { // login callback
            if (code) msg(wialon.core.Errors.getErrorText(code)); // login failed, print error
            else msg("Logged successfully"); // login succeed
        }
    );
}

// Logout
function logout() {
    var user = wialon.core.Session.getInstance().getCurrUser(); // get current user
    if (!user){ msg("You are not logged, click 'login' button"); return; }
    wialon.core.Session.getInstance().logout( // if user exist - logout
        function (code) { // logout callback
            if (code) msg(wialon.core.Errors.getErrorText(code)); // logout failed, print error
            else msg("Logout successfully"); // logout suceed
        }
    );
}
```

```
// Get current user and prints its name to log
function getUser() {
    var user = wialon.core.Session.getInstance().getCurrUser(); // get current user
    // print message
    if (!user) msg("You are not logged, click 'login' button"); // user not exists
    else msg("You are logged as '" + user.getName() + "'"); // print current user name
}

// execute when DOM ready
$(document).ready(function(){
    // For more info about how to generate token check
    // http://sdk.wialon.com/playground/demo/app_auth_token
    $("#token").val("5dce19710a5e26ab8b7b8986cb3c49e58C291791B7F0A7AEB8AFBFCEED7DC03BC48FF5F8");
    // bind actions to buttons click
    $("#login_btn").click( login );
    $("#logout_btn").click( logout );
    $("#user_btn").click( getUser );
});
```

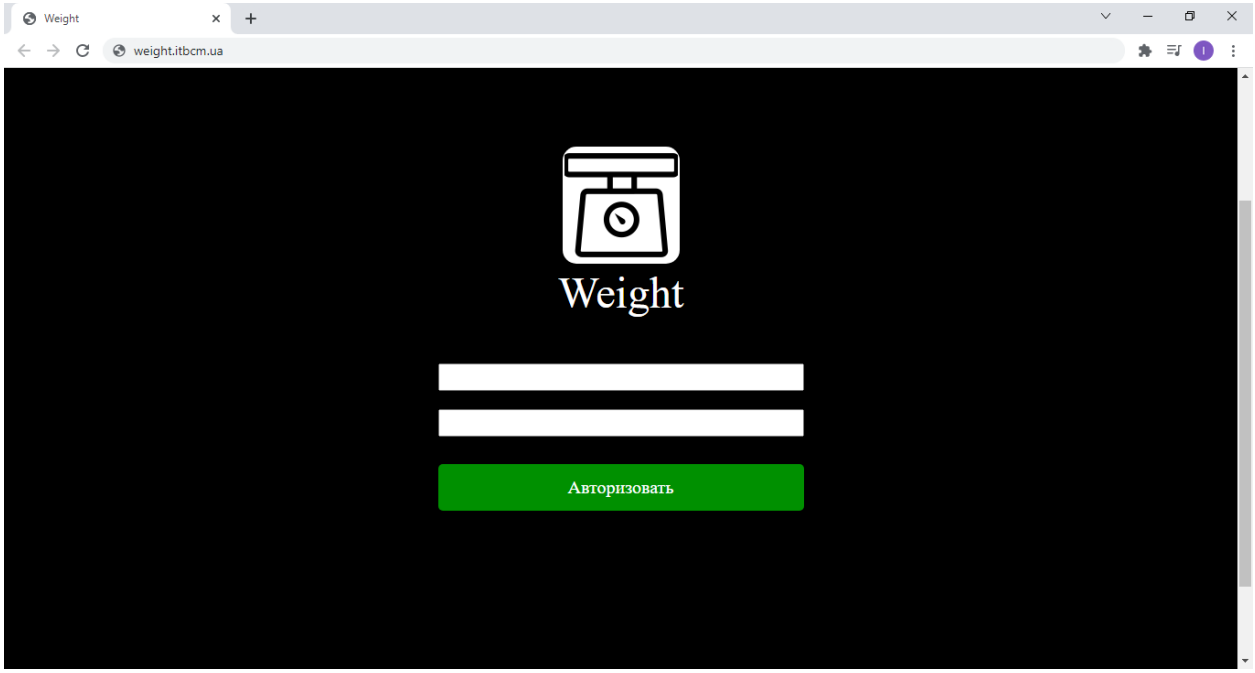


Рис. 2. Вхід до системи

Для експорту даних з головної системи, використано такий код:

```
// Print message to log
function msg(text) { $("#log").prepend(text + "<br/>"); }

function init() { // Execute after login succeed
    // flags to specify what kind of data should be returned
    var flags = wialon.item.Item.dataFlag.base;
    wialon.core.Session.getInstance().updateDataFlags( // load items to current session
        [{type: "type", data: "avl_unit", flags: flags,mode: 0}], // Items specification
        function (code) {  // updateDataFlags callback
            if (code) { msg(wialon.core.Errors.getErrorText(code)); return; } // exit if error code
            initData(); // execute if items load succeed
    });
}

function initData() { // Execute after items load to Session
    var sess = wialon.core.Session.getInstance(); // get instance of current Session
    var units = sess.getItems("avl_unit"); // get loaded 'avl_unit's items
    if (!units || !units.length){ msg("Units not found"); return; } // check if units found

    for (var i=0; i<units.length; i++) // construct Select object using found units
        // append option with current unit to select
        $("#units").append("<option value='" + units[i].getId() + "'>" + units[i].getName() + "</option>");
}

function loadMessages(){ // load messages function
    var sess = wialon.core.Session.getInstance(); // get instance of current Session
    var to = sess.getServerTime(); // get ServerTime, it will be end time
    var from = to - 3600*24; // get begin time ( end time - 24 hours in seconds )

    var unit = $("#units").val(); // get selected unit id
    if(!unit){ msg("Select unit first"); return; } // exit if no unit selected
    var ml = sess.getMessagesLoader(); // get messages loader object for current session
    ml.loadInterval(unit, from, to, 0, 0, 100, // load messages for given time interval
        function(code, data){ // loadInterval callback
            if(code){ msg(wialon.core.Errors.getErrorText(code)); return; } // exit if error code
            else { msg(data.count +" messages loaded. Click 'Show messages'");} // print success message
        }
    );
}
```

```
function showMessages(from, to){ // print given indicies (from, to) of messages
    $("#messages").html(""); // clear message container
    // get messages loader object for current session
    var ml = wialon.core.Session.getInstance().getMessagesLoader();
    ml.getMessages(from, to, //get messages data for given indicies
        function(code, data){ // getMessages callback
            if(code){ msg(wialon.core.Errors.getErrorText(code)); return; } // exit if error code
            else if(data.length == 0){ // exit if no messages loaded
                msg("Nothing to show. Load messages first"); return;}
            var from_index = from; // counter for display
            for(var i=0; i<data.length; i++) // display result cycle
                $("#messages").append( // append current message row to result table
                    "<tr"+ (i%2==1?" class='odd' ":"") +"><td>"+ (from_index++) +"</td>"+
                    // print Json data of current message
                    "<td>"+wialon.util.Json.stringify(data[i])+"</td></tr>");
            msg(data.length + " messages shown from "+ from+" to "+ to); // Print message to log
        }
    );
}

// execute when DOM ready
$(document).ready(function () {
    // bind actions to button clicks
    $("#load_btn").click( loadMessages );
    $("#show_btn").click( function(){ showMessages($("#show_from").val(),$("#show_to").val()); } );

    wialon.core.Session.getInstance().initSession("https://hst-api.wialon.com"); // init session
    // For more info about how to generate token check
    // http://sdk.wialon.com/playground/demo/app_auth_token
    wialon.core.Session.getInstance().loginToken("5dce19710a5e26ab8b7b8986cb3c49e58C291791B7F0A7AEB8AFBFCEED7DC03BC48FF5F8", "", // try to login
        function (code) { // login callback
            // if error code - print error message
            if (code){ msg(wialon.core.Errors.getErrorText(code)); return; }
            msg("Logged successfully"); init(); // when login suceed then run init() function
    });
});

```

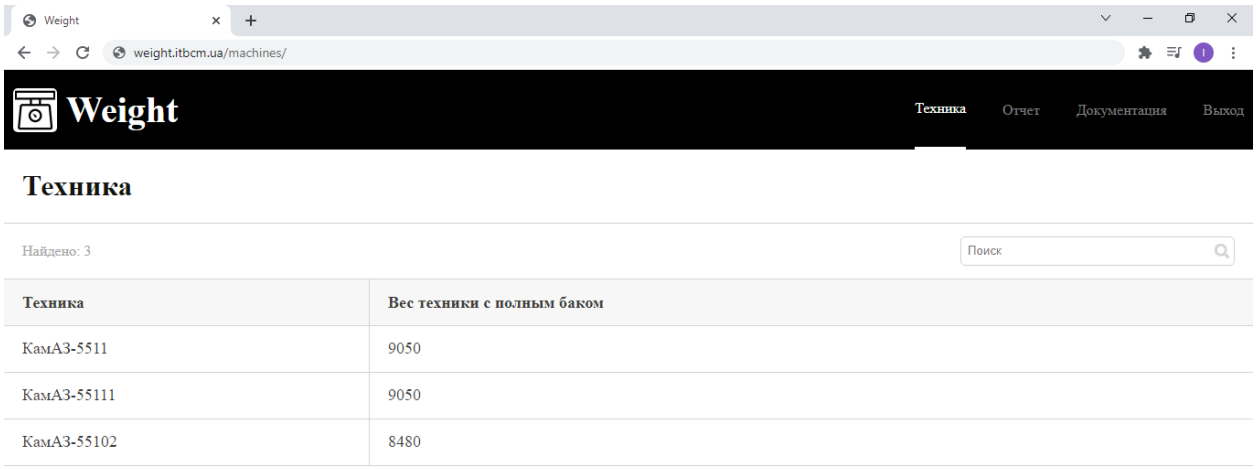


Рис. 3. Список техніки користувача

Звіт було створено за допомогою, такого коду:

```
// Print message to log
function msg(text) { $("#log").prepend(text + "<br/>"); }

function init() {// Execute after login succeed
    // specify what kind of data should be returned
    var res_flags = wialon.item.Item.dataFlag.base | wialon.item.Resource.dataFlag.reports;
    var unit_flags = wialon.item.Item.dataFlag.base;

    var sess = wialon.core.Session.getInstance(); // get instance of current Session
    sess.loadLibrary("resourceReports"); // load Reports Library
    sess.updateDataFlags( // load items to current session
        [{type: "type", data: "avl_resource", flags:res_flags , mode: 0}, // 'avl_resource's specification
         {type: "type", data: "avl_unit", flags: unit_flags, mode: 0}], // 'avl_unit's specification
        function (code) { // updateDataFlags callback
            if (code) { msg(wialon.core.Errors.getErrorText(code)); return; } // exit if error code

            var res = sess.getItems("avl_resource"); // get loaded 'avl_resource's items
            if (!res || !res.length){ msg("Resources not found"); return; } // check if resources found
            for (var i = 0; i< res.length; i++) // construct Select object using found resources
                $("#res").append("<option value='" + res[i].getId() + "'>" + res[i].getName() + "</option>");

            getTemplates(); // update report template list

            $("#res").change( getTemplates ); // bind action to select change

            var units = sess.getItems("avl_unit"); // get loaded 'avl_units's items
            if (!units || !units.length){ msg("Units not found"); return; } // check if units found
            for (var i = 0; i< units.length; i++) // construct Select object using found units
                $("#units").append("<option value='"+ units[i].getId() +"'>"+ units[i].getName()+ "</option>");
        }
    );
}

function getTemplates(){ // get report templates and put it in select list
    $("#templ").html("<option></option>"); // ad first empty element
    var res = wialon.core.Session.getInstance().getItem($("#res").val()); // get resource by id
    // check user access to execute reports
    if (!wialon.util.Number.and(res.getUserAccess(), wialon.item.Item.accessFlag.execReports)){
        $("#exec_btn").prop("disabled", true); // if not enough rights - disable button
        msg("Not enought rights for report execution"); return; // print message and exit
    } else $("#exec_btn").prop("disabled", false); // if enough rights - disable button

    var templ = res.getReports(); // get reports templates for resource
    for(var i in templ){
        if (templ[i].ct != "avl_unit") continue; // skip non-unit report templates
        // add report template to select list
        $("#templ").append("<option value='"+ templ[i].id +"'>"+ templ[i].n+ "</option>");
    }
}
```

```
function executeReport(){ // execute selected report
    // get data from corresponding fields
    var id_res=$("#res").val(), id_templ=$("#templ").val(), id_unit=$("#units").val(), time=$("#interval").val();
    if(!id_res){ msg("Select resource"); return;} // exit if no resource selected
    if(!id_templ){ msg("Select report template"); return;} // exit if no report template selected
    if(!id_unit){ msg("Select unit"); return;} // exit if no unit selected

    var sess = wialon.core.Session.getInstance(); // get instance of current Session
    var res = sess.getItem(id_res); // get resource by id
    var to = sess.getServerTime(); // get current server time (end time of report time interval)
    var from = to - parseInt( $("#interval").val(), 10); // calculate start time of report
    // specify time interval object
    var interval = { "from": from, "to": to, "flags": wialon.item.MReport.intervalFlag.absolute };
    var template = res.getReport(id_templ); // get report template by id
    $("#exec_btn").prop("disabled", true); // disable button (to prevent multiclick while execute)

    res.execReport(template, id_unit, 0, interval, // execute selected report
        function(code, data) { // execReport template
            $("#exec_btn").prop("disabled", false); // enable button
            if(code){ msg(wialon.core.Errors.getErrorText(code)); return; } // exit if error code
            if(!data.getTables().length){ // exit if no tables obtained
                msg("<b>There is no data generated</b>"); return; }
            else showReportResult(data); // show report result
    });
}
```

```
function showReportResult(result){ // show result after report execute
    var tables = result.getTables(); // get report tables
    if (!tables) return; // exit if no tables
    for(var i=0; i < tables.length; i++){ // cycle on tables
        // html contains information about one table
        var html = "<b>"+ tables[i].label +"</b><div class='wrap'><table style='width:100%'>";

        var headers = tables[i].header; // get table headers
        html += "<tr>"; // open header row
        for (var j=0; j<headers.length; j++) // add header
            html += "<th>" + headers[j] + "</th>";
        html += "</tr>"; // close header row
        result.getTableRows(i, 0, tables[i].rows, // get Table rows
            qx.lang.Function.bind( function(html, code, rows) { // getTableRows callback
                if (code) {msg(wialon.core.Errors.getErrorText(code)); return;} // exit if error code
                for(var j in rows) { // cycle on table rows
                    if (typeof rows[j].c == "undefined") continue; // skip empty rows
                    html += "<tr"+(j%2==1?" class='odd' ":"")+">"; // open table row
                    for (var k = 0; k < rows[j].c.length; k++) // add ceils to table
                        html += "<td>" + getTableValue(rows[j].c[k]) + "</td>";
                    html += "</tr>";// close table row
                }
                html += "</table>";
                msg(html +"</div>");
            }, this, html)
        );
    }
}

function getTableValue(data) { // calculate ceil value
    if (typeof data == "object")
        if (typeof data.t == "string") return data.t; else return "";
    else return data;
}

// execute when DOM ready
$(document).ready(function () {
    $("#exec_btn").click( executeReport ); // bind action to button click

    wialon.core.Session.getInstance().initSession("https://hst-api.wialon.com"); // init session
    // For more info about how to generate token check
    // http://sdk.wialon.com/playground/demo/app_auth_token
    wialon.core.Session.getInstance().loginToken("5dce19710a5e26ab8b7b8986cb3c49e58C291791B7F0A7AEB8AFBFCEED7DC03BC48FF5F8", "",
        function (code) { // login callback
            // if error code - print error message
            if (code){ msg(wialon.core.Errors.getErrorText(code)); return; }
            msg("Logged successfully"); init(); // when login suceed then run init() function
    });
});
```

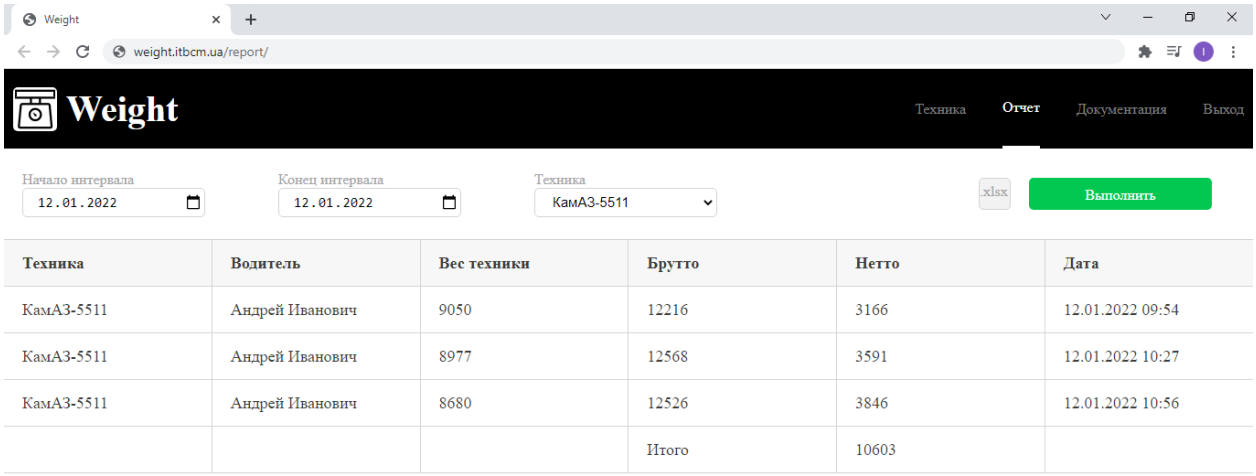
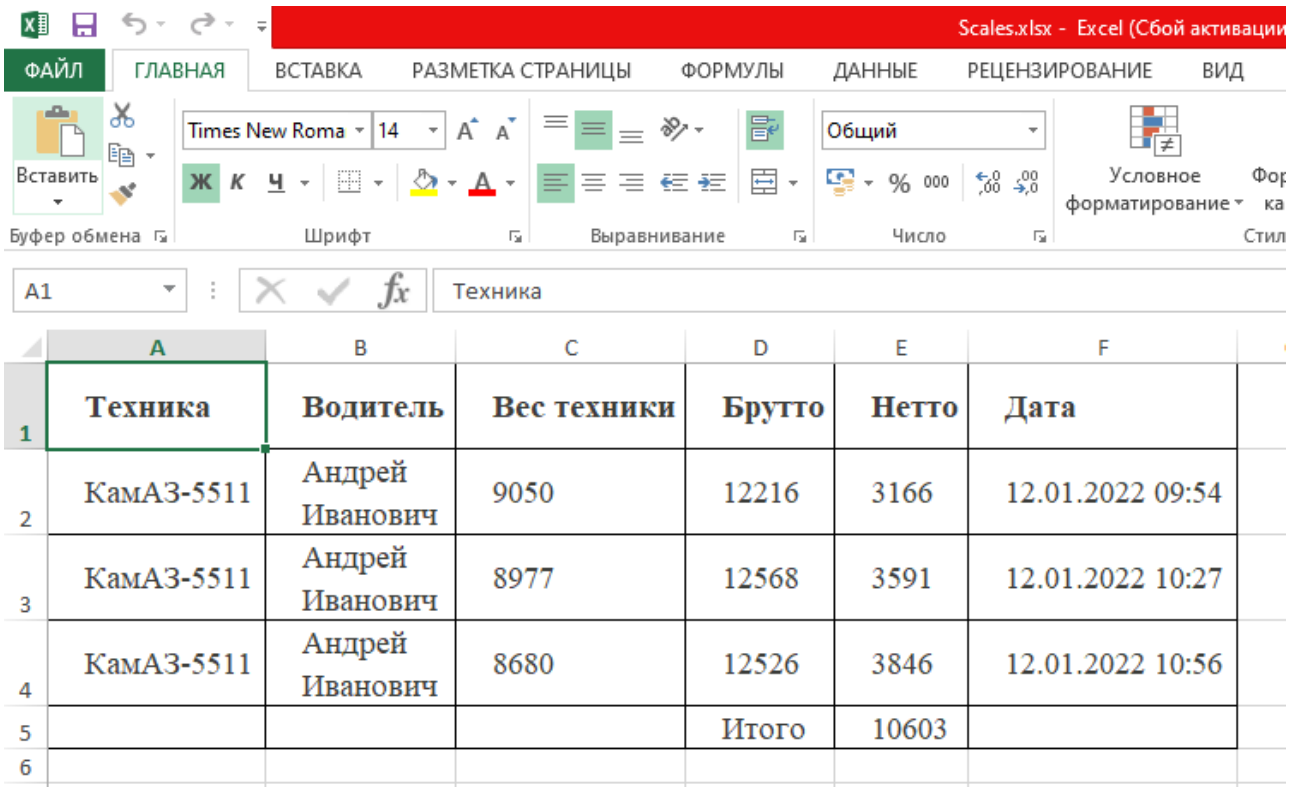


Рис. 4. Звіт зважувань техніки



	A	B	C	D	E	F
1	**Техника**	**Водитель**	**Вес техники**	**Брутто**	**Нетто**	**Дата**
2	КамАЗ-5511	Андрей Иванович	9050	12216	3166	12.01.2022 09:54
3	КамАЗ-5511	Андрей Иванович	8977	12568	3591	12.01.2022 10:27
4	КамАЗ-5511	Андрей Иванович	8680	12526	3846	12.01.2022 10:56
5				Итого	10603	
6						

Рис. 5. Експортований звіт у Excel

2.8 Алгоритм для пришвидшення зважування

Також в ході виконання поставлених завдань було запропоновано алгоритм для пришвидшення зважування, за рахунок переведення системи зважування в роботу динамічного визначення ваги. Алгоритм дозволить виключити хибні значення та за рахунок згладжування буде обраховувати вагу більш точно.

Робота алгоритму:

- спочатку за формулою 1 усереднюємо графік;
- за формулою 2 знаходимо похідну;
- за допомогою похибки виявляємо інтервал усереднення;
- визначаємо шукане значення за формулою 4.

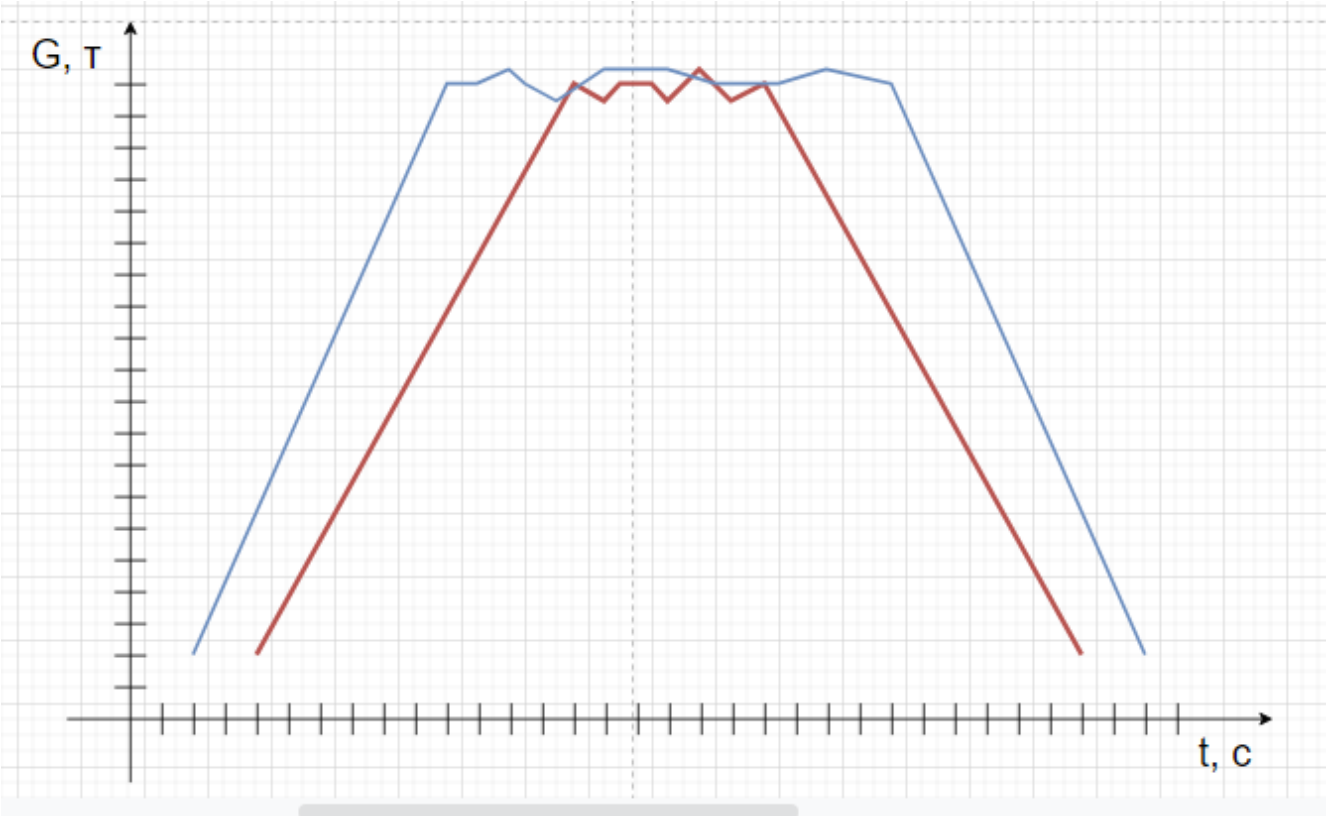


Рис. 7. Графік зважування техніки (синій – техніка повільно заїжджає на ваги; червоний - техніка швидко проїжджає датчики ваги)

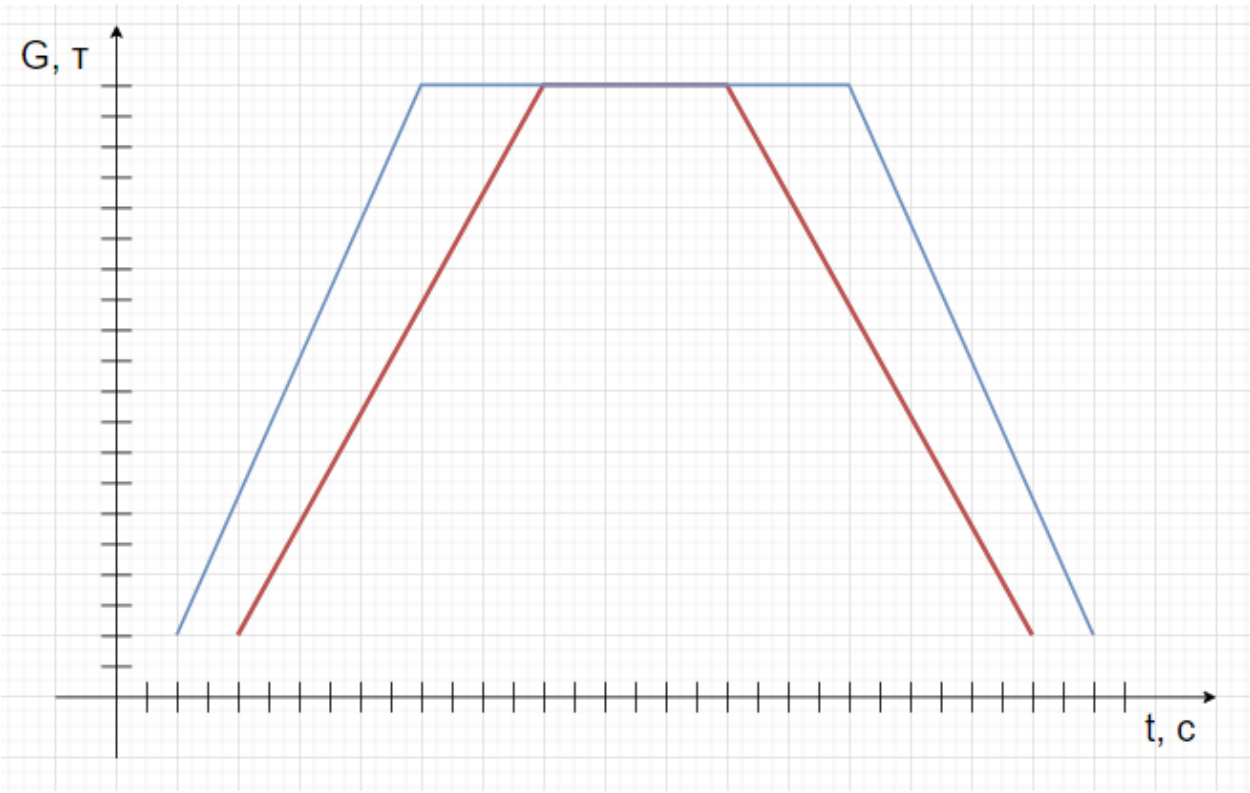


Рис. 6. Графік зважування техніки після застосування алгоритму

$$x = \frac{1}{2j+1}\sum_{j=-3}^{3} x(k_j) \tag{1}$$

$$x` = \frac{x(k_{j+1}) - x(k_{j-1})}{2\Delta t} \tag{2}$$

$$|x`|\Delta t \leq \Delta \tag{3}$$

$$x^* = \frac{1}{N}\sum_{k=1}^{N} x(k) \tag{4}$$

3 РОЗДІЛ

В процесі виконання поставленого завдання було вирішено такі проблеми:

- створення інформаційної системи для роботи з приладами для зважування техніки з сільськогосподарською продукцією;
- розроблено алгоритм для пришвидшення зважування, за рахунок переведення системи зважування в роботу динамічного визначення ваги.

На даний момент для введення алгоритму в роботу потрібно змінити принцип роботи самої системи зважування. Система зважування працює за таким алгоритмом:

- пуста техніка заїжджає на ваги;
- водій прикладає RFID картку до зчитувача;
- дані про вагу надходять до системи;
- техніка їде заповнюватися;
- заповнена техніка заїжджає на ваги;
- водій прикладає RFID картку до зчитувача;
- дані про брутто вагу надходять до системи.

Для роботи алгоритму потрібно замінити принцип роботи системи зважування, на роботу в режимі моніторингу даних. В майбутньому компанія перейде на такий принцип роботи. Тому поставлені завдання виконані повністю.

ВИСНОВКИ

В даній кваліфікаційній роботі:

- проведено аналіз роботи системи зважування;
- проаналізовано дані, які надсилає система зважування до головної програми;
- досліджено потреби компаній, для яких створюється інформаційна система, та складено список майбутніх функцій системи;
- створено програму-додаток;
- розроблено алгоритм для пришвидшення зважування, за рахунок переведення системи зважування в роботу динамічного визначення ваги.

Виконання поставлених задач дозволить зекономити кошти на придбані нового обладнання та збільшить зручність використання системи, за рахунок їх взаємопов’язаної роботи. Також в майбутньому планується оновити сам механізм зважування, для цього було розроблено алгоритм для пришвидшення зважування, за рахунок переведення системи зважування в роботу динамічного визначення ваги.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ловцов Д. А. Информационные системы в профессиональной деятельности : учеб.-метод. комплекс / Д. А. Ловцов, А. В. Зайцев, Е. С. Бурмистрова. – М. : РАП, 2008. – 14 с.

2. Грекул В.И. Проектирование информационных систем: учеб. пособие для студентов вузов по специальностям в обл. информ. технологий / В.И. Грекул, Г.Н. Денищенко, Н.Л. Коровкина. – М.: БИНОМ. Лаборатория знаний, 2008. – 300с.

3. Береза А. М. Основи створення інформаційних систем /Навч. посіб. – К.: КНЕЧ, 1998. – 140 с.

4. Фримен Эрик, Фримен Элизабет. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — П.: «Питер», 2010. — 656 с. — ISBN 978-5-49807-113-8.

5. Эд Титтел, Джефф Ноубл. HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011. — 400 с. — ISBN 978-5-8459-1752-2.

6. Питер Лабберс, Брайан Олберс, Фрэнк Салим. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений = Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development. — М.: «Вильямс», 2011. — 272 с. — ISBN 978-5-8459-1715-7.

7. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010. — 656 с. — ISBN 978-5-8459-1676-1.

8. Фримен Эрик, Фримен Элизабет. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — 1-е изд. — М.: «Питер», 2010. — С. 656. — ISBN 978-5-49807-113-8.

9. Дэвид Сойер Макфарланд. Новая большая книга CSS = CSS: The Missing Manual. — Санкт-Петербург: Питер, 2017. — 720 с. — 1000 экз. — ISBN 978-5-496-02080-0.

10. Эд Титтел, Джефф Ноубл. HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011. — 400 с. — ISBN 978-5-8459-1752-2.

11. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2011. — 656 с. — ISBN 978-5-8459-1676-1.

12. Mayer G. E. T., Awesomeness J. JavaScript: JavaScript Awesomeness Book. CreateSpace Independent Publishing Platform, 2017. 68 с.

13. Bride M. JavaScript. Teach Yourself Books, 2003. 192 с.

14. Gosselin D. JavaScript. 4-те вид. Boston : Thomson / Course Technology, 2008. 700 с.

15. Gosselin D. Javascript. 5-те вид. Boston, Mass : Course Technology, 2011. 899 с.

16. API A. API Polyurethanes Expo 2001. CRC, 2001. 664 с.

17. Api. Polyurethanes Conference 2000: Defining the Future Through Technology. CRC, 2000. 615 с.

ДОДАТКИ

Додаток А. Лістинг index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="assets/css/style.css">
    <title>Варова</title>
</head>
<body>

   <div class="intro">
       <div class="container">
           <div class="intro_inner">
                <img class="logo_img" src="assets/images/1.png" alt="">
                <div class="logo">Варова</div>
                <input class="inpt"><br>
                <input class="inpt"><br>
                <a class="btn" href="machine.html">Авторизовать</a>
            </div>
        </div>
   </div>

</body>
</html>
```

Додаток Б. Лістинг index.css

```
body {
    margin: 0;
}

*,
*:before,
*:after {
    box-sizing: border-box;
}

/* Container */
.container {
    width: 100%;
    margin: 0 auto;
}

/* Intro */
.intro {
    height: 970px;
    background: #000;
    color: #fff;
    display: flex;
    flex-direction: column;
    justify-content: center;
}

.intro_inner {
    text-align: center;
}

.logo_img {
    border-radius: 15px;
}

.logo {
    margin-bottom: 40px;
    font-size: 48px;
}

.inpt {
    margin: 10px;
    height: 30px;
    width: 400px;
}

.btn {
    width: 400px;
    display: inline-block;
    vertical-align: top;
    color: white;
    font-size: 20px;
    background-color: #009000;
    text-decoration: none;
    padding: 14px 40px;
    border-radius: 5px;
    margin-top: 20px;
}

.btn:hover {
    background-color: green;
}
```