DOI 10.34185/1562-9945-4-153-2024-06
УДК 004.45

І.В. Соловей, О.Г. Ворочек

РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ВЕДЕННЯ ЦИФРОВОЇ КАРТКИ ДОМАШНЬОЇ ТВАРИНИ

Анотація. Для вирішення проблеми комплексного ведення цифрової картки домашньої тварини пропонується розробити відповідну програмну систему. Перед розробкою ма ють бути обрані технології для реалізації програмної системи, що включає наступний функціонал: електронний паспорт тварини, ведення активності тварини, даних про вакцинації та захист від втрати завдяки пошуку загубленої тварини за географічною позицією у найближчому радіусі. Для розробки мають бути використані програмне сере довище PyCharm, серверна платформа Django, мови програмування Python, JavaScript, сис тема керування базами даних SQLite.
Ключові слова: програмна система, електронний паспорт, домашні тварини, Django, Python, JavaScript, СКБД, SQLite.

Постановка проблеми. 21 сторіччя – час цифрових технологій, які стрімко розвиваються. Ведення друкованого документообігу – це технологія минулого, і саме час переходити на цифрову документацію. Це зручно і допомагає екології.

У наш час оцифровані копії людських документів вже є нормою (наприклад, державний сервіс Дія). Але, на жаль, такий комплексний підхід відсутній у сфері домашніх тварин. Саме тому було прийнято рішення про створення програмної системи для ведення цифрової картки домашньої тварини.

Перед розробкою програмної системи потрібно визначитися з технологіями, підібрати найкращі варіанти для створення програмної системи, щоб вони відповідали її вимогам. Потрібно зробити аналіз, щоб у майбутньому система могла бути масштабована та доповнена за функціоналом.

Програмна система є багатофункціональною та об'єднує в собі паспорт, медичну картку, ведення активності і захист від втрати з унікальною дошкою оголошень. Ці дані можна використовувати замість паперових, тому що вони захищені і не можуть бути підроблені за допомогою фотомонтажу.

Власник тварини матиме можливість створити паспорт своєї тварини, який буде інтернаціоналізованим незалежно від обраної користувачем мови та матиме захист за допомогою QR коду. Програмна система дозволить вести активність тварини за допомогою вбудованого в нього таймеру – користувачу лише треба буде вказати тип активності та натиснути на кнопку початку чи зупинення. При бажанні користувач зможе переглянути активність за минулі дати. Також сервіс спростить взаємодію власника тварини та ветеринара – користувач повинен буде заповнити інформацію про вакцинацію тварини, та, у разі коректного заповнення, ветеринар зможе підтвердити її за допомогою сканування QR коду. Також унікальною функцією сервісу є дошка оголошень втрачених тварин. Втрата тварини фіксується за місцезнаходженням та відображаються найближчим до нього користувачам.

Аналіз останніх досліджень і публікацій. Наразі, Україна має один з вищих показників кількості котів на душу населення в Європі. Згідно останніх зібраних даних, в Україні налічується приблизно 7.63 млн. домашніх котів [1].

За даними World Population Review, Україна займає друге місце у світі за кількістю котів на душу населення, поступаючись лише США [2].

На жаль, трапляються випадки, коли тварина може втекти з дому або загубитися під час прогулянки. За даними Американської Гуманної Асоціації [3], третина домашніх тварин втрачається в якийсь момент свого життя, і близько 10 млн собак і кішок втрачаються в США щороку.

Наведені дані показують актуальність проблеми та необхідність створення системи, яка би не просто спростила взаємодію з документами улюбленців, а й також допомогла захистити тварин у разі втрати.

Для розробки будь-якої програмної системи потрібно обрати правильні технології. Так як система призначена як для власників, так і для ветеринарів, варто розробляти окремий веб-інтерфейс, а не тільки панель для внутрішнього використання ветеринарними закладами.

Має бути розроблена серверна частина, яка дозволить взаємодіяти з даними та виводити їх у веб-частину. Серед сучасних технологій для розробки серверної частини (або бекенду), Django, Node.js, Ruby on Rails, і ASP.NET Core виділяються своєю продуктивністю та гнучкістю. Node.js пропонує асинхронну, подієорієнтовану архітектуру, ідеальну для обробки великої кількості запитів, що робить його вибором для реалізації реактивних веб-додатків. Ruby on Rails використовує принцип «Convention over Configuration», що сприяє швидкій розробці, а ASP.NET Core забезпечує високу продуктивність та легку інтеграцію

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

з іншими продуктами Microsoft [4]. Натомість, Django надає великий набір готових до використання компонентів і підтримку чистого та ефективного проєктування додатків.

Django вважається однією з найкращих систем для написання бекенду, завдяки своїй архітектурі «batteries-included». Це означає, що в стандартному пакеті Django йдуть усі інструменти необхідні для створення веб-додатків, які можуть включати аутентифікацію користувачів, URL-шаблони, інтерфейс адміністраторської панелі та багато інших можливостей, які звільняють розробника від необхідності «винаходити колесо». Django сприяє швидкій розробці, що ідеально підходить для швидкісного впровадження додатків. Також, Django має велику та активну спільноту, сотню доступних модулів, які можуть бути використані для розширення його можливостей. Ця платформа надзвичайно безпечна, що робить її відмінним вибором для проєктів, які вимагають надійної обробки даних і транзакцій, з автоматичними механізмами захисту від багатьох видів атак [5].

JavaScript є оптимальним вибором для розробки клієнтської частини веб-додатків у поєднанні з Django через свою універсальність і підтримку асинхронних запитів через AJAX, що забезпечує швидке оновлення контенту без перезавантаження сторінки [6]. Інтеграція з фреймворками на кшталт React, Angular або Vue сприяє створенню динамічних SPA (Single Page Applications), значно покращуючи UX (User Experience). Подібна комбінація характеризується високим рівнем масштабованості та безпеки, завдяки чому це ідеально підходить для створення складних веб-додатків.

Мета дослідження. Метою дослідження є створення програмної системи для ведення цифрової картки домашньої тварини. Система повинна включати у себе такі функції як: ведення активностей тварини, додавання медичних даних, електронний паспорт та дошка оголошень про втрати. Система має складатися з серверної (Backend) та клієнтської (Frontend) частини.

Викладення основного матеріалу дослідження. Для створення програмної системи потрібно зрозуміти який саме функціонал вона має включати.

Основні можливості системи розподіляються на власників тварини та працівників ветеринарних медичних закладів.

Власники отримують наступний функціонал:

– електронний паспорт тварини з можливістю підтвердження за допомогою QR коду;

– додавання даних активності тварини з розділенням на години, дні та місяці;

– додавання даних про вакцинацію тварин з інформацією про термін придатності, підтвердження своїм підписом;

– дошка оголошень, на якій можуть відображатися найближчі за місцезнаходженням заявки про втрату з можливістю відмітити свою тварину;

– функція прив'язки тегу нашийнику для покращення взаємодії з медичними закладами та додаткового захисту від втрати.

Ветеринарам, у свою чергу, будуть доступні:

– можливість перегляну дані тварини при скануванні QR коду;

– підтвердження даних про вакцинацію за допомогою сканування QR коду з автоматичним додаванням електронного підпису ветеринару;

– доступ до медичних записів тварини.

Отже, коли ми визначились з кінцевим функціоналом програмної системи, можна приступити до UML проєктування.

UML (Уніфікована мова моделювання) — це стандартизована мова моделювання, що складається з інтегрованого набору діаграм, розроблена для допомоги розробникам систем і програмного забезпечення у визначенні, візуалізації, конструюванні та документуванні артефактів програмних систем, а також для бізнес-моделювання та інших непрограмних систем [7]. Взаємодію користувачів із системою показано на UML діаграмі (див. рис. 1).

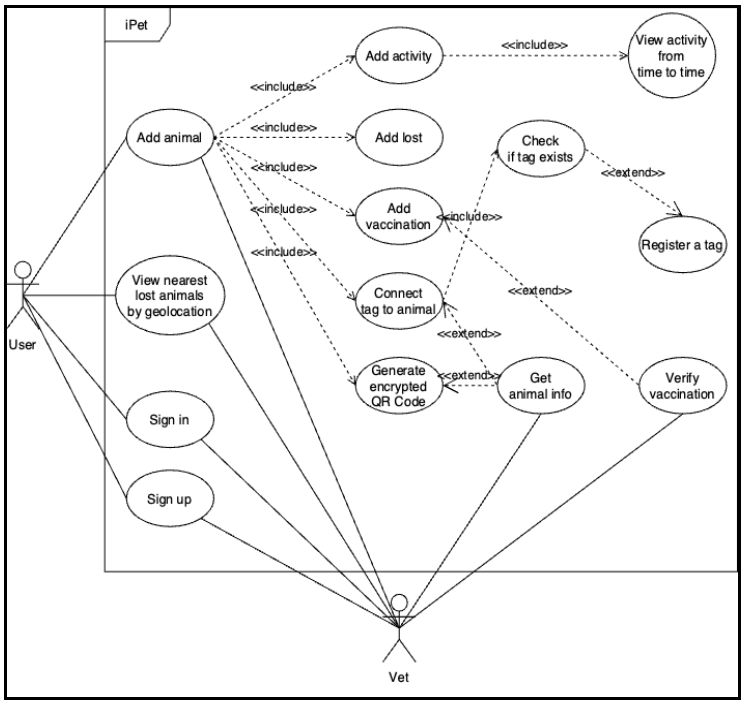

Рисунок 1 – Діаграма прецедентів

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

На діаграмі вказані відносини між акторами і прецедентами у програмній системі.

Таким чином, за допомогою мови UML та діаграми прецедентів було візуально відображено взаємодію сутностей програмної системи, що загалом надає розуміння певних процесів та дозволяє перейти безпосередньо до розробки програмного забезпечення.

REST API (Representational State Transfer API) – це інтерфейс програмування додатків (API), який відповідає архітектурному стилю REST, що використовується для веб-сервісів. REST API розроблені таким чином, що не мають стану та використовують HTTP-методи (GET, POST, PUT, DELETE і PATCH) для роботи з представленнями ресурсів [8].

Програмна система має включати в себе:

– серверну частину. Завдяки технології REST API надається доступ до бази даних іншим компонентам за допомогою API запитів, що дозволяють отримувати, надсилати, редагувати та видаляти інформацію, авторизуватися завдяки хешованому токену користувача;

– клієнтську частину. Дозволяє користуватися системою з будь-якого сучасного веб-браузеру, взаємодіє із сервером за допомогою API запитів – має можливість завантажувати, оновлювати та видаляти дані з сервера та надсилати нові, є відокремленою від серверної частини.

Для розробки серверної частини обрано Python фреймворк Django REST – потужний і гнучкий набір інструментів для створення веб-API. Це відкрита бібліотека, яка надає готовий до використання набір інструментів для побудови API, що включає серіалізацію, яка підтримує джерела даних як з ORM (Object-Relational Mapping), так і без нього, політики аутентифікації, і можливість налаштування аж до рівня даних [9].

Бази даних (БД) є основою переважної більшості застосування сучасних програмних засобів [10] Система використовує базу даних SQLite, структуру якої наведено на ER-діаграмі (див. рис. 2).

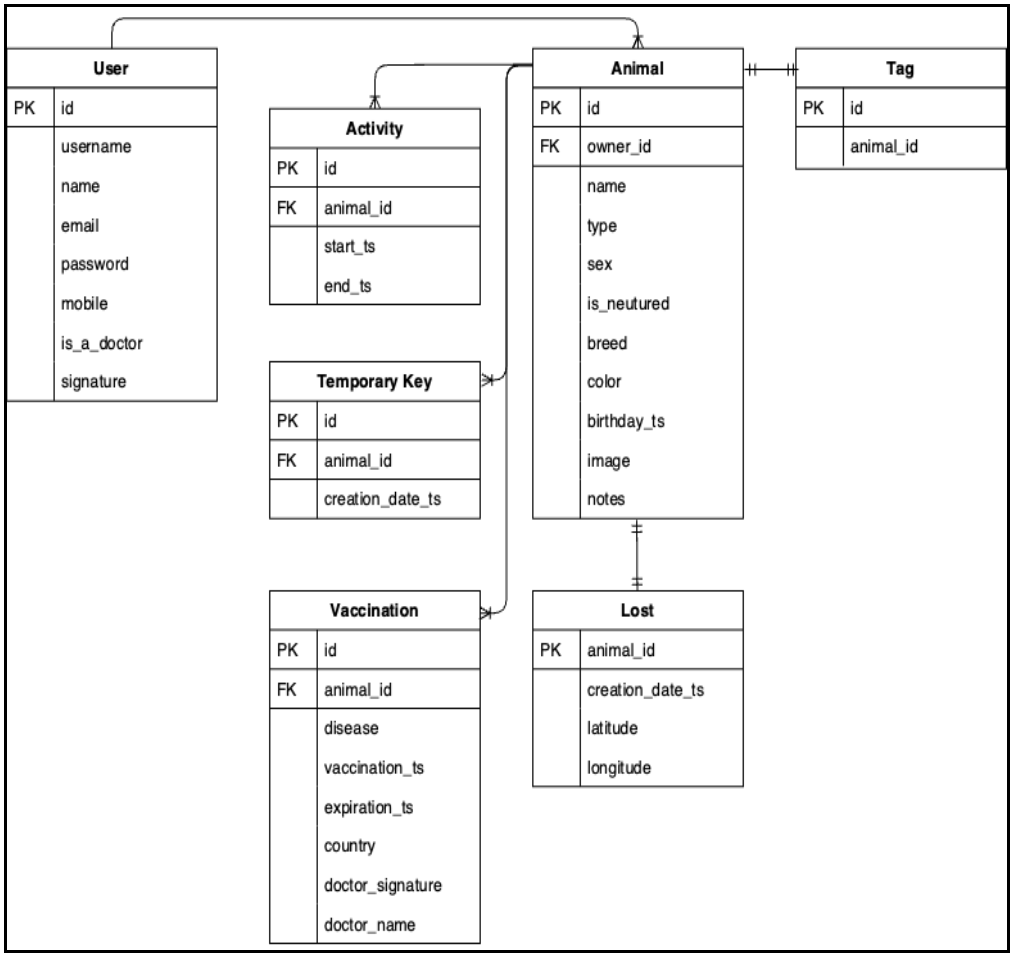

Рисунок 2 – Архітектура Бази Даних

Структура бази даних представлена 7 таблицями:

– User. Таблиця користувачів із персональними даними (ім’я, мобільний та підпис) та інформацією для аутентифікації;

– Animal. Таблиця з даними про тварину. Містить такі дані: ім’я тварини, тип (собака / кіт), стать, чи стерилізована, порода, окрас, день народження, зображення, нотатки;

– Activity. Таблиця, яка містить дані про активність тварини в обмежений проміжок часу;

– Vaccination. Таблиця з даними про вакцинацію тварини. Містить поля: хвороба, дати вакцинації та терміну придатності, країна.

– Lost. Таблиця, де фіксується втрата з вказанням геопозиції власника;

– Temporary Key. Таблиця, яка містить згенерований ключ QR коду, що використовується для підтвердження паспорту тварини;

– Tag. Таблиця, яка містить унікальний тег нашийника тварини.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

У якості патерну проєктування було обрано MVC (Model-View-Controller). Model-View-Controller (MVC) — це архітектурний шаблон, що використовується у розробці ПЗ для забезпечення чистого розділення відповідальності між представленням даних користувачу (View), управлінням даними (Model) та бізнес-логікою, яка взаємодіє з вводом користувача для підтримки даних (Controller). Цей шаблон допомагає організувати код таким чином, що відокремлює інтерфейс користувача, дані та логіку, яка контролює взаємодії, одне від одного, тим самим полегшуючи розробникам управління складністю та підвищення підтримуваності [11].

Для розробки клієнтської (Frontend) частини програмної системи у якості мови програмування було обрано JavaScript та фреймворк jQuery, а у якості мови розмітки – HTML та CSS.

Інтерфейс (з англ. interface - взаємодія) - сукупність засобів, що забезпечують взаємодію пристроїв обчислювальної системи та програм, а також їх взаємодію з людиною. Інтерфейс користувача - комплекс програмних і апаратних засобів, що забезпечують взаємодію користувача з системою [12]. Він повинен відповідати сучасним нормам UI/UX, щоб бути зручним для кінцевого користувача.

Для розгортання використовується вебсервер Apache. Клієнтська частина взаємодіє з серверною (Django REST Framework) завдяки REST API. Вона зберігає авторизаційний токен, завдяки якому може отримати доступ до запитів.

При проєктуванні системи було створено окремі директорії для HTML сторінок, JavaScript коду та CSS стилів, щоб зробити роботу з проєктом зручніше.

Модуль Аутентифікації клієнтської частини має компоненти Реєстрації та Авторизації, а Користувача – Моїх Тварин (My Animals), Втрачених Тварин (Lost Animals), Вакцинації (Vaccination) та Налаштувань (Settings). Компоненти забезпечують надання даних та відображення їх користувачу. Вони мають схожу структуру, де <component_name>.html відповідає за відображення сторінки, <component_name>.css – за стилі, а <component_name>.js – за взаємодію з даними.

Висновки. Результатом дослідження є створена програмна системи для ведення цифрової картки домашньої тварини. Було чітко прописано функціонал, розроблено базу даних та структуру відношень між користувачами, обрано технології для розробки як серверної, так і веб-частини.

ЛІТЕРАТУРА

1. Pet Ownership by Country (Dogs and Cats). Mappr. URL: https://www.mappr.co/thematic-maps/world-pet-ownership/.
2. Pet Ownership Statistics by Country 2024. World Population Review. URL: https://worldpopulationreview.com/country-rankings/pet-ownership-statistics-by-country.
3. US Missing Pet Epidemic and Euthanasia Statistics: Facts/Figures. Peeva. URL: https://peeva.co/missing-pet-epidemic-facts-and-figures.
4. Comparing Backend Frameworks: Node.js, Django, Ruby on Rails, and Dot.Net. habitualCS. URL: https://habitualcs.io/exploring-different-backend-frameworks-node-js-django-ruby-on-rails-and-dot-net-etc/.
5. Django – Overview. URL: https://www.djangoproject.com/start/overview/
6. Souza D. Using React with Django to create an app: Tutorial. LogRocket. URL: https://blog.logrocket.com/using-react-django-create-app-tutorial/.
7. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition) [Text] / Martin Fowler. – New York : Addison-Wesley, 2004. – 208 p.
8. Fielding R. Architectural Styles and the Design of Network-based Software Architectures [Text] / Roy Thomas Fielding. – Irvine : University of California, 2000. – 180 p.
9. Gastón C. Hillar. Django RESTful Web Services: The easiest way to build Python RESTful APIs and web services with Django [Text] / Gastón C. Hillar. – Birmingham : Packt Publishing, 2018. – 306 p.
10. Mazurova O., Naboka A., Shirokopetleva M. Research of acid transaction implementation methods for distributed databases using replication technology // Сучасний стан наукових досліджень та технологій в промисловості : науковий журнал. – 2021. – № 2(16). – С. 19–31.
11. Fowler, M. Patterns of Enterprise Application Architecture [Text] / Martin Fowler. – Boston : Addison-Wesley, 2002. – 533 p.
12. Ареф'єв О. О. Інтерфейс користувача: особливості розробки інтерфейсу користувача для ігрових застосунків // Радіоелектроніка та молодь у XXI столітті: зб. матеріалів 25-го Міжнародного молодіжного форуму. Т. 6. – Харків: ХНУРЕ, 2021. – С. 3.

REFERENCES

1. Pet Ownership by Country (Dogs and Cats). Mappr. URL: https://www.mappr.co/thematic-maps/world-pet-ownership/.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

2. Pet Ownership Statistics by Country 2024. World Population Review. URL: https://worldpopulationreview.com/country-rankings/pet-ownership-statistics-by-country.
3. US Missing Pet Epidemic and Euthanasia Statistics: Facts/Figures. Peeva. URL: https://peeva.co/missing-pet-epidemic-facts-and-figures.
4. Comparing Backend Frameworks: Node.js, Django, Ruby on Rails, and Dot.Net. habitualCS. URL: https://habitualcs.io/exploring-different-backend-frameworks-node-js-django-ruby-on-rails-and-dot-net-etc/.
5. Django – Overview. URL: https://www.djangoproject.com/start/overview/
6. Souza D. Using React with Django to create an app: Tutorial. LogRocket. URL: https://blog.logrocket.com/using-react-django-create-app-tutorial/.
7. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition) [Text] / Martin Fowler. – New York : Addison-Wesley, 2004. – 208 p.
8. Fielding R. Architectural Styles and the Design of Network-based Software Architectures [Text] / Roy Thomas Fielding. – Irvine : University of California, 2000. – 180 p.
9. Gastón C. Hillar. Django RESTful Web Services: The easiest way to build Python RESTful APIs and web services with Django [Text] / Gastón C. Hillar. – Birmingham : Packt Publishing, 2018. – 306 p.
10. Mazurova O., Naboka A., Shirokopetleva M. Research of acid transaction implementation methods for distributed databases using replication technology // Suchasnyy stan naukovykh doslidzhen ta tekhnolohiy v promyslovosti : naukovyy zhurnal. – 2021. – № 2(16). – P. 19–31.
11. Fowler, M. Patterns of Enterprise Application Architecture [Text] / Martin Fowler. – Boston : Addison-Wesley, 2002. – 533 p.
12. Arefyev O.O. Interfeys korystuvacha: osoblyvosti rozrobky interfeysu korystuvacha dlya ihrovykh zastosunkiv // Radioelektronika ta molod u KhKhI stolitti: zb. materialiv 25-ho Mizhnarodnoho molodizhnoho forumu. Vol. 6. – Kharkiv: KhNURE, 2021. – P. 3.

Received 29.04.2024.
Accepted 02.05.2024.

Development of the Software System for Managing a Digital Pet ID

Recent Research and Publications Analysis: The shift towards digital documentation, such as Ukraine's «Diia» for human records, has not yet been fully embraced in pet care, creating a significant service gap. Ukraine, with a high number of household pets, especially approximately 7.63 million cats, highlights the need for a comprehensive digi-

tal pet management system. Given the frequent cases of pets getting lost—with low recovery rates—a robust digital system is essential for improving these figures and enhancing pet safety.

When designing software, it is important to choose the right technologies for both the web interface and the backend. Django is recommended for the backend because of its «batteries-included» architecture, which provides a comprehensive set of ready-to-use tools that facilitate rapid development and ensure a high level of security. For the interface, it is recommended to use JavaScript integrated with frameworks such as React, Angular or Vue to create dynamic applications that improve the user experience with asynchronous requests, allowing the content of the page to be updated without reloading the page. This combination not only provides scalability and security, but also effectively meets the complex needs of web applications.

Purpose of the Study. This study aims to develop a software system that facilitates the management of digital pet IDs, which will integrate medical records, vaccination histories, and detailed activity logs. This integration aims to streamline pet care, making it more efficient and significantly more convenient for pet owners.

Main Material Presentation. The proposed system's architecture will include:

– backend. Utilizing Django REST Framework for creating scalable, secure web APIs that handle data operations efficiently;

– frontend. Employing JavaScript, HTML, and CSS to provide a responsive and interactive user experience.

Key Features:

– digital passports for pets, verifiable via QR codes;

- detailed activity logs that track and display pet movements and behaviors;

– comprehensive vaccination records accessible by both pet owners and veterinarians;

– a lost pet bulletin board that uses geographical data to notify users of nearby lost or found pets.

Conclusions.Technologies have been chosen and a software system has been developed for effective management of digital pet IDs, integrating key functions for comprehensive data management of pets. It utilizes modern technologies to ensure reliable data security, high scalability, and enhanced user interaction, making it a key achievement in the management of digital pet IDs.

Keywords: software system, electronic passport, pets, Django, Python, JavaScript, DBMS, SQLite.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Ворочек Ольга Григорівна - к.т.н., доцент кафедри ПІ харківського Національного Університету Радіоелектроніки.

Соловей Ілля Владиславович - здобувач вищої освіти факультету комп'ютерних наук Харківського Національного Університету Радіоелектроніки.

Vorochek Olga Hryhorivna - Associate Professor of the Department of Software Engineering, Candidate of Science (Technology) at Kharkiv National University of Radio Electronics.

Solovei Illia Vladyslavovych - higher education seeker at the Faculty of Computer Science of Kharkiv National University of Radio Electronics.