НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.415.2

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмний метод оптимізації міжміських пасажирських перевезень з використанням нейронних мереж»

Виконав:
студент VI курсу, групи КП-71мп
Миколайчик Володимир Вікторович _____

Науковий керівник:
Доцент кафедри ПЗКС, к.т.н.,
Олещенко Л.М. _____

Рецензент: доцент кафедри автоматики та управління
в технічних системах ФІОТ, к.т.н., доцент
Полторак В.П. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення» («Програмне забезпечення комп'ютерних та інформаційно-пошукових систем»)

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____ 2017 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Миколайчику Володимиру Вікторовичу

1. Тема дисертації «Програмний метод оптимізації міжміських пасажирських перевезень з використанням нейронних мереж», науковий керівник дисертації: доцент кафедри ПЗКС, к.т.н., Олещенко Любов Михайлівна, затверджені наказом по університету від «15» листопада 2018 р. № 4173-с

2. Термін подання студентом дисертації «14» грудня 2018 р.

3. Об'єкт дослідження: мережа міжміських пасажирських перевезень.

4. Предмет дослідження: програмні методи обробки даних про пасажирів з використанням нейронних мереж

5. Перелік завдань, які потрібно розробити:
   - провести аналіз існуючих рішень;
   - провести розробку архітектури нейронної мережі;
   - розробити принципи виконання прогнозування пасажиропотоку;
   - запропонувати метод оптимізації міжміських перевезень;
   - розробити програмне забезпечення, що реалізує запропонований метод оптимізації міжміських перевезень;
   - провести дослідження за допомогою розробленого програмного методу.

6. Орієнтовний перелік публікацій:
   - XI наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» (ПМК-2018-2);
   - X науково-технічна конференції «Комп'ютерні системи і мережні технології 2017» (м. Київ);
   - «Програмне забезпечення для прогнозування пасажиропотоку та організації рухомого складу АТП» в міжнародному науковому журналі "Інтернаука". — 2017. — №11.

7. Дата видачі завдання «15» грудня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.02.2018	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.03.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	16.04.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	14.05.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	17.09.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2018.	02.10.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу;	28.10.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	10.11.2018	

Студент В.В. Миколайчик

Науковий керівник дисертації Л.М. Олещенко

РЕФЕРАТ

Актуальність теми. В умовах зростання цін на паливо виникає необхідність у використанні інформаційних технологій, спрямованих на оптимізацію роботи автотранспортних підприємств (АТП). Виникнення задачі раціонального використання та обслуговування рухомого складу АТП зумовлене збільшенням рухливості населення у регіонах України внаслідок ситуації на Сході та пошуку робочих місць у великих містах.

Існуючі АТП не забезпечують своєчасного і якісного обслуговування пасажирів в Україні. В моменти максимумів пасажиропотоку, коли утворюються черги пасажирів, внаслідок відсутності технологій обробки і передачі даних про пасажирів, погіршується якість їх обслуговування.

На даний час відсутні технології для автоматизованого прийняття рішень щодо оптимального розподілу транспортних засобів відносно можливого значення пасажиропотоку.

Об'єктом дослідження є мережа міжміських пасажирських перевезень.

Предметом дослідження є програмні методи обробки даних про пасажирів з використанням нейронних мереж.

Мета дослідження: удосконалити метод прогнозування пасажиропотоку з використанням нейронної мережі, оптимізувати розподіл рухомого складу залежно від отриманих значень пасажиропотоку, створити програмну систему для управлінням пасажирськими перевезеннями на міжміському маршруті.

Методи дослідження. Для розробки методу оптимізації міжміських перевезень використано рекурентні нейронні мережі та методи математичного програмування.

Наукова новизна роботи полягає в наступному:

1. Розроблено метод прогнозування пасажиропотоку на основі рекурентної нейронної мережі, який дозволяє отримувати прогнози з точністю 98% за рахунок аналізу додаткових факторів, які впливають на величину пасажиропотоку.
2. Розроблено автоматизовану систему для оптимального розподілу транспортних засобів на основі уточнених прогнозних значень пасажиропотоку.

Практична цінність отриманих в роботі результатів полягає в тому, що запропонований метод оптимізації міжміських перевезень дозволяє отримувати точніші значення пасажиропотоку і за рахунок цього більш раціонально використовувати рухомий склад, що дозволяє зменшувати витрати АТП.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (м. Київ), представлені на X науково-технічній конференції «Комп'ютерні системи і мережні технології 2017» (м. Київ), опублікована стаття «Програмне забезпечення для прогнозування пасажиропотоку та організації рухомого складу АТП» в міжнародному науковому журналі "Інтернаука". – 2017. – №11.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, п'ятьох розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів.

У першому розділі розглянуто класифікацію систем обробки даних про пасажирів для автотранспортних підприємств, особливості їх програмної

реалізації, проаналізовано їх переваги та недоліки, розглянуто існуючі рішення.

У другому розділі розглянуто загальну концепцію виконання прогнозування. Запропоновано метод оптимізації та загальну архітектуру нейронної мережі.

У третьому розділі наведено загальний опис методу, описані вимоги користувача, функціональні вимоги, можливості системи; наведена архітектура системи. Описані основні класи розробленої системи та інтерфейс користувача.

У четвертому розділі розглянуті отримані результати дослідження. Проведено прогнозування пасажиропотоку різними методами з використанням нейронних мереж та порівняно їх результати.

У п’ятому розділі було розглянуто спосіб комерціалізації проведеного дослідження. В запропонованій бізнес моделі описано проблему предметної області, виділені зацікавлені сторони, комерційне рішення та його основні характеристики, конкурентні переваги, клієнти та сегменти ринку, унікальна ціннісна пропозиція, здійснено аналіз доходів та витрат.

У висновках проаналізовано отримані результати роботи.

У додатках наведено копія презентації, лістинги основних класів розробленого методу, схеми архітектури системи.

Робота виконана на 74 аркушах, містить 2 додатки та посилання на список використаних літературних джерел з 40 найменувань. У роботі наведено 33 рисунка та 6 таблиць.

Ключові слова: рекурентні нейронні мережі, метод прогнозування, пасажиропотік, програмне забезпечення, оптимізація, автотранспортне підприємство.

ABSTRACT

Theme urgency. In the context of rising fuel prices, there is a need for the use of information technology aimed at optimizing the work of motor transport enterprises (MTE). The emergence of the problem of rational use and maintenance of the MTE rolling stock is due to an increase in the mobility of the population in the regions of Ukraine due to the situation in the East and job search in large cities.

Existing MTE do not provide timely and qualitative service to passengers in Ukraine. At moments of maximum passenger traffic, when queues of passengers are formed, due to the lack of processing technologies and the transfer of passenger data, the quality of their servicing deteriorates.

At present, there are no technologies for automated decision-making on the optimal distribution of vehicles relative to the possible value of passenger traffic.

The object of research is a network of long-distance passenger transportation.

The subject of the research is the program methods of processing data on passengers using neural networks.

The purpose of the research: to improve the method of forecasting the flow of passengers using the neural network, optimize the distribution of rolling stock depending on the received values of passenger traffic, to create a software system for managing passenger traffic on the long-distance route.

Research methods. Recurrent neural networks and methods of mathematical programming are used to develop the method of optimization of long-distance traffic.

The scientific novelty of the work is as follows:

1. The method of forecasting of passenger traffic on the basis of a recurrent neural network is developed, which allows obtaining more accurate

forecasts in 98% due to the analysis of additional factors that affect the amount of passenger traffic.

2. An automated system for optimal distribution of vehicles has been developed on the basis of refined forecasting values of passenger traffic, which takes into account the economic and technical parameters of vehicles and passenger service time.

Practical value of the results obtained in the work is that the proposed method of optimization of long-distance traffic allows you to obtain more accurate values of passenger traffic and, therefore, more efficiently use the rolling stock, which reduces the cost of MTE.

Approbation. The main provisions and results of the work were presented and discussed at the scientific conference of masters and postgraduates "Applied Mathematics and Computer", PMK-2018 (Kyiv), presented at the Xth Scientific and Technical Conference "Computer Systems and Network Technologies 2017" (Kyiv), the article "Software for forecasting the traffic flow and organization of rolling stock MTE" was published in the international scientific journal "Interna-science". - 2017 - No. 11.

Structure and content of the thesis. The master's dissertation consists of an introduction, five sections, conclusions and appendices.

The introduction provides a general description of the work, an assessment of the current state of the problem is carried out, the relevance of the direction is substantiated researches, the purpose and tasks of researches are formulated, the scientific novelty of the results obtained and the practical value of the work are shown, information on the approbation of the results is given.

In the first section the classification of passenger data processing systems for motor transport enterprises, their principles of work, their advantages and disadvantages are analyzed, existing solutions are considered.

The second section discusses the general concept of performing forecasting. The method of optimization and the general architecture of the neural network is proposed.

The third section gives a general description of the method, describes the requirements of the user, functional requirements, system capabilities; The architecture of the system is given. The main classes of the developed system and the user interface are described.

The fourth section examined the results of the study. It was carried out performing forecasting by different methods using neural networks and comparing their results.

In the fifth section the way of commercialization of the conducted research was considered. The proposed business model describes the problem of the subject area, the identified stakeholders, the commercial solution and its main characteristics, competitive advantages, customers and market segments, a unique value proposition, an analysis of income and expenditure.

The conclusions are analyzed the results of work.

The appendixes contain a copy of the presentation, lists of the main classes of the developed method, the scheme of the architecture of the system.

The work is performed on 74 sheets, contains 2 attachments and a link to the list of used literary sources of 40 titles. The paper presents 33 figures and 6 tables.

Key words: recurrent neural networks, forecasting method, passenger traffic, software, optimization, motor transport enterprise.

РЕФЕРАТ

Актуальность темы. В условиях роста цен на топливо возникает необходимость использования информационных технологий, направленных на оптимизацию работы автотранспортных предприятий (АТП). Возникновение задачи рационального использования и обслуживания подвижного состава АТП обусловлено увеличением подвижности населения в регионах Украины вследствие ситуации на Востоке и поиска рабочих мест в крупных городах.

Существующие АТП не обеспечивают своевременного и качественного обслуживания пассажиров в Украине. В моменты максимумов пассажиропотока, когда образуются очереди пассажиров, из-за отсутствия технологий обработки и передачи данных о пассажирах, ухудшается качество их обслуживания.

В настоящее время отсутствуют технологии для автоматизированного принятия решений относительно оптимального распределения транспортных средств относительно возможного значения пассажиропотока.

Объектом исследования является сеть междугородных пассажирских перевозок.

Предметом исследования являются программные методы обработки данных о пассажирах с использованием нейронных сетей.

Цель исследования: усовершенствовать метод прогнозирования пассажиропотока с использованием нейронной сети, оптимизировать распределение подвижного состава в зависимости от полученных значений пассажиропотока, создать программную систему для управлением пассажирскими перевозками на междугородном маршруте.

Методы исследования. Для разработки метода оптимизации междугородных перевозок использовано рекуррентные нейронные сети и методы математического программирования.

Научная новизна работы заключается в следующем:

1. Разработан метод прогнозирования пассажиропотока на основе рекуррентной нейронной сети, который позволяет получать прогнозы с точностью 98% за счет анализа дополнительных факторов, влияющих на величину пассажиропотока.
2. Разработана автоматизированная система для оптимального распределения транспортных средств на основе уточненных прогнозных значений пассажиропотока, которая учитывает экономико-технические показатели транспортных средств и время обслуживания пассажиров.

Практическая ценность полученных в работе результатов заключается в том, что предложенный метод оптимизации междугородных перевозок позволяет получать более точные значения пассажиропотока и за счет этого более рационально использовать подвижной состав, позволяющий уменьшать расходы АТП.

Апробация работы. Основные положения и результаты работы были представлены и обсуждались на научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2018 (г.. Киев), представленные на X научно-технической конференции «Компьютерные системы и сетевые технологии 2017» (м. Киев), опубликована статья «Программное обеспечение для прогнозирования пассажиропотока и организации подвижного состава АТП» в международном научном журнале "Интернаука". - 2017. - №11.

Структура и объем работы. Магистерская диссертация состоит из введения, пяти глав, заключения и приложений.

Во введении дано общая характеристика работы, выполнена оценка современного состояния проблемы, обоснована актуальность направления

исследований, сформулированы цели и задачи исследований, показано научную новизну полученных результатов и практическую ценность работы, приведены сведения об апробации результатов.

В первой главе рассмотрена классификация систем обработки данных о пассажирах для автотранспортных предприятий, их принципы работы, проанализированы их преимущества и недостатки, рассмотрены существующие решения.

Во втором разделе рассмотрена общая концепция выполнения прогнозирования. Предложен метод оптимизации и общую архитектуру нейронной сети.

В третьем разделе приведены общее описание метода, описаны требования пользователя, функциональные требования, возможности системы; приведена архитектура системы. Описаны основные классы разработанной системы и интерфейс пользователя.

В четвертом разделе были рассмотрены полученные результаты исследования. Проведено прогнозирование пассажиропотока различными методами с использованием нейронных сетей, наведено сравнение их результатов.

В пятом разделе были рассмотрены способ коммерциализации проведенного исследования. В предложенной бизнес модели описано проблему предметной области, выделенные заинтересованные стороны, коммерческое решение и его основные характеристики, конкурентные преимущества, клиенты и сегменты рынка, уникальная ценностная предложение, осуществлен анализ доходов и расходов.

В выводах проанализированы полученные результаты работы.

В приложениях приведены копия презентации, листинги основных классов разработанного метода, схемы архитектуры системы.

Работа выполнена на 74 листах, содержит 2 приложения и ссылки на список использованных литературных источников из 40 наименований. В работе приведены 33 рисунка и 6 таблиц.

Ключевые слова: рекурентные нейронные сети, метод прогнозирования, пассажиропоток, программное обеспечение, оптимизация, автотранспортное предприятие.

ЗМІСТ

СПИСОК ТЕРМІНІВ

АТП – автотранспортне підприємство.

ДКЧП – довга короткочасна пам'ять.

RNN – recurrent neural networks.

LSTM – long short-term memory.

ARIMA – узагальнена модель авто регресійної моделі ковзної середньої.

AR(p) – авторегресивна модель порядку p.

ВСТУП

В умовах кризи та непередбачуваного зростання цін на паливо виникає необхідність у створенні програмного забезпечення, спрямованого на оптимізацію міжміських пасажирських перевезень автотранспортними підприємствами. Постійне збільшення попиту на автобусні перевезення у міжміському сполученні вимагає прийняття рішень щодо заміни застарілого обліку в журналах з використанням удосконаленої технології аналізу статистичних даних при плануванні розкладу, а також прийняття рішень з питань організації та обслуговування рухомого складу. Відповідно до статистики, мегаполіси є привабливими для залучення трудових ресурсів з менш економічно розвинених прилеглих територій. На даний час пасажиропотік не достатньо аналізується та враховується при плануванні рухомого складу автотранспортними підприємствами.

Існуючі технічні рішення для автотранспортних підприємств не забезпечують раціонального використання транспортних засобів для міжміського перевезення пасажирів та створення ефективного розкладу рухомого складу для пасажирських перевезень. Виникає задача створення програмного методу, який буде дозволяти опрацьовувати дані про навантаженість пасажиропотоку в реальному часі.

Використовуючи подібне програмне забезпечення, диспетчер зможе збирати дані про кількість пасажирів в кожен конкретний момент часу, зберігати його в базі, а згодом використовувати для виконання прогнозування пасажиропотоку на майбутнє. Таким чином, з’являється можливість точніше складати графіки перевезень і, відповідно, моделювати необхідну кількість одиниць рухомого складу та кількість водіїв на обраний час згідно прогнозного значення числа пасажирів. Також можливим буде зберігати й іншу корисну інформацію про маршрути, поїздки і контролювати кількість одиниць рухомого складу [1].

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

У даному розділі розглянуто програмні системи, які використовуються для організації роботи автотранспортних підприємств у міжміському сполученні. Виділено невирішені задачі, пов’язані з обробкою даних про пасажирів та організацією структури рухомого складу.

1.1. Класифікація програмних систем для управління діяльністю автотранспортних підприємств

В Україні організація пасажирських автобусних перевезень в основному відбувається на основі складання розкладу руху автобусів. Розклад є в пункті відбуття і призначення. Автобуси виїжджають чітко за розкладом, вони можуть виходити у рейс інколи напівпорожніми. Такі явища є збитковими для АТП. Через нестабільність потоку людей дуже важко реалізувати раціональний розклад, щоб одночасно задовольнити інтереси користувачів та інтереси АТП. Прогноз та статистика в такому випадку абсолютна відсутня.

1.1.1. Програмні платформи для управління діяльністю АТП

За допомогою «1С» (рис. 1.1.) можна налаштувати систему управління для конкретного підприємства. Через персональний комп’ютер поїздки будуть записуватися до бази, де можливим буде перегляд статистики.

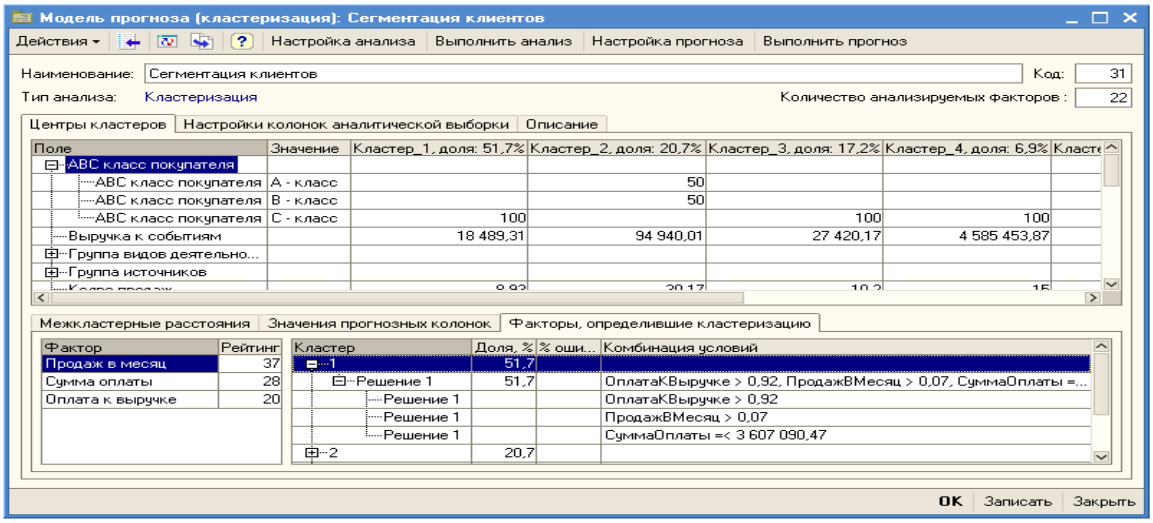


Рис. 1.1. Приклад форми 1С

Можна сказати, що для великих компаній це найзручніший варіант, проте є велика й кількість недоліків.

Переваги:

- простота замовлення системи;
- повна організація автопарку.

Недоліки:

- надзвичайно велика ціна за систему;
- неможливість законного використання через санкції;
- неможливість прогнозування без участі програміста;
- неможливість роботи через планшет чи мобільний телефон.

Отже, використання такої системи не підходить для АТП, що займаються міжміськими пасажирськими перевезеннями, так як більшість підприємств не зможе зручно використовувати персональні комп'ютери через відсутність місця та можливості як таких.

1.1.2. Управління засобами Excel

На багатьох ресурсах можна знайти інструкції або готові документи формату Microsoft Excel для заповнення та складання звітів. Використовуючи такий спосіб, потрібно буде повністю вручну вводити дані про пасажиропотік і та записи про поїздки. Такий спосіб дуже затратний за

часом і диспетчер повинен запам’ятовувати велику кількість інформації. На жаль, в Україні в малих приватних підприємствах взагалі не ведеться обліку або ж диспетчери змушені записувати все до блокнотів вручну і, відповідно, з такими даними складно робити прогнози.

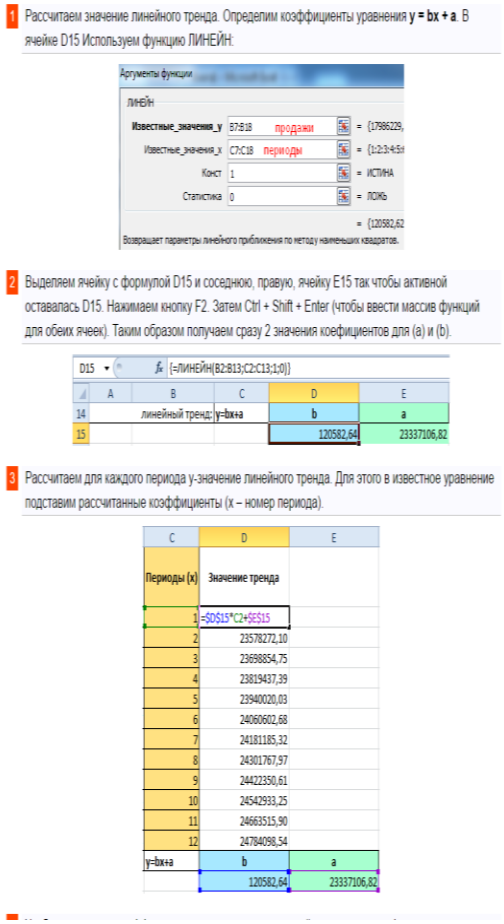


Рис. 1.2. Приклад отримання прогнозу в Microsoft Excel

1.1.3. Ручний запис

Більшість приватних підприємств в Україні досі не використовують жодної системи для зберігання статистичних даних про пасажиропотік. Усі дані записуються вручну до облікових журналів і, відповідно, аналіз і обробка цих даних неможлива без витрат колосального часу. Як водії, так і диспетчери лиш інтуїтивно мають уявлення про кількість пасажирів на заданий час.

1.1.4. Комплексні системи для управління

Серед малого та середнього бізнесу користуються популярністю системи управління на основі «1С», «Бітрікс» та інших, які зображено на рис. 1.3.

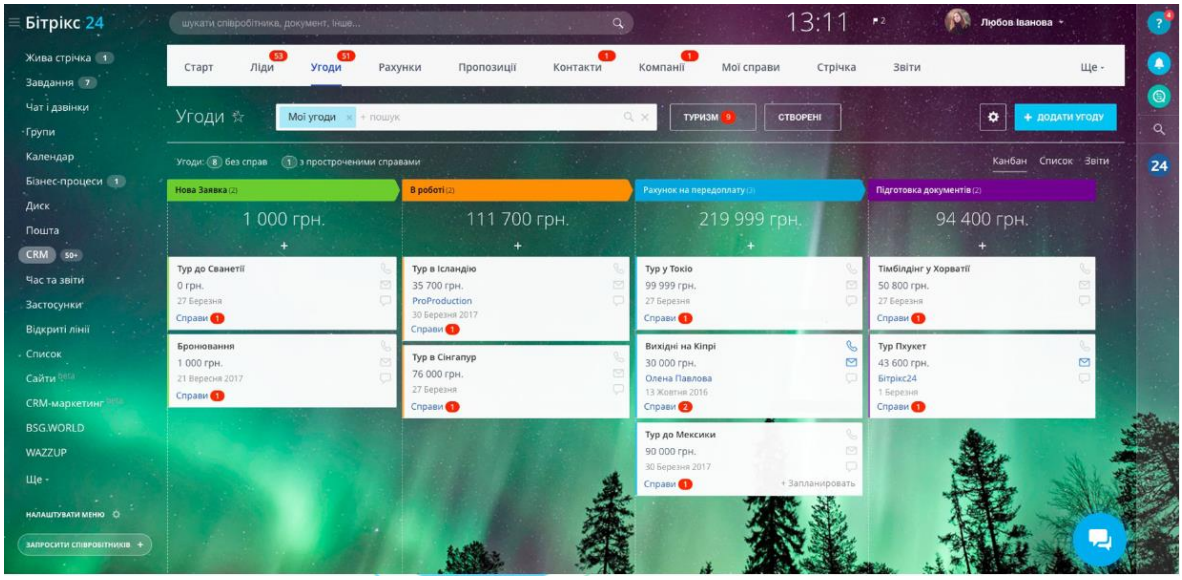


Рис. 1.3. Приклад системи для управління

Системи дозволяють покрити велику кількість проблем в управлінні бізнесом, але дуже складно і майже неможливо зробити таку систему гнучкою і налаштувати саме під бізнес автотранспортних підприємств. В подібних системах неможливо реалізувати та використовувати прогнозування, що відносить їх до незадовільної групи.

1.2. Аналіз існуючих методів прогнозування

Прогноз часових рядів – це важлива область машинного навчання. Це важливо, тому що існує так багато проблем прогнозування, які включають часовий компонент. Однак, компонент часу додає додаткову інформацію, він також ускладнює обробку завдань часового ряду в порівнянні з багатьма іншими завданнями прогнозування.

Основним завданням дисертаційної роботи є дослідження та порівняння існуючих методів прогнозування пасажиропотоку, провадження нових алгоритмів та методів, вибір найкращого методу та

Место для уравнения.реалізація програмного забезпечення для прогнозування пасажиропотоку.

У зв'язку з цим будуть розглянуті найпоширеніші методи для прогнозування часових рядів.

На сьогоднішній день методи машинного навчання та алгоритми глибокого навчання запровадили нові підходи до проблеми прогнозування. Серед методів машинного навчання, які використовують для прогнозування часових рядів, в основному виділяють такі методи, як метод опорних векторів, випадковий ліс. Серед методів глибокого навчання виділяють рекурентну нейронну мережу (RNN), довгу короткострокову пам'ять (LSTM) та їх варіації. Методи глибокого навчання здатні боротися із нелінійністю та часовою складністю прогнозування. Зокрема, LSTM був використаний у багатьох областях програмування, таких як обробка природної мови, розпізнавання рукописного тексту, ідентифікація мови, прогноз часових рядів [4] тощо.

Важливим питанням даного дослідження є порівняння точності традиційних методів прогнозування із точністю методів глибокого навчання. Після дослідження предметної області було виявлено, що на даний момент усі основні існуючі програми для прогнозування пасажиропотоку в України використовують традиційні методи замість методів машинного навчання.

1.2.1. «Наївні» моделі прогнозування

При створенні "наївних" моделей передбачається, що деякий основний період прогнозованого часового ряду краще всього описує майбутнє цього прогнозованого ряду, тому в цих моделях прогноз, як правило, є дуже простою функцією від значень прогнозованої змінної в недалекому минулому.

Найпростішою моделлю є наступна:

$$Y + 1 = Yt, \tag{1.1}$$

що відповідає припущенню, що «завтра буде як сьогодні».

Поза всіляким сумнівом, від такої примітивної моделі не варто чекати великої точності. Вона не тільки не враховує механізми, що визначають прогнозовані дані (цей серйозний недолік взагалі притаманний багатьом статистичним методам прогнозування), але і не захищена від випадкових коливань, вона не враховує сезонні коливання і тенденції. Втім, можна будувати «наївні» моделі дещо по-іншому

$$Yt+1 = Yt + [Yt - Yt-1], \quad (1.2)$$

$$Yt+1 = Yt * [Yt / Yt-1], \quad (1.3)$$

такими способами ми намагаємося пристосувати модель до можливих тенденцій:

$$Yt+1 = Yt-S, \quad (1.4)$$

це спроба врахувати сезонні коливання.

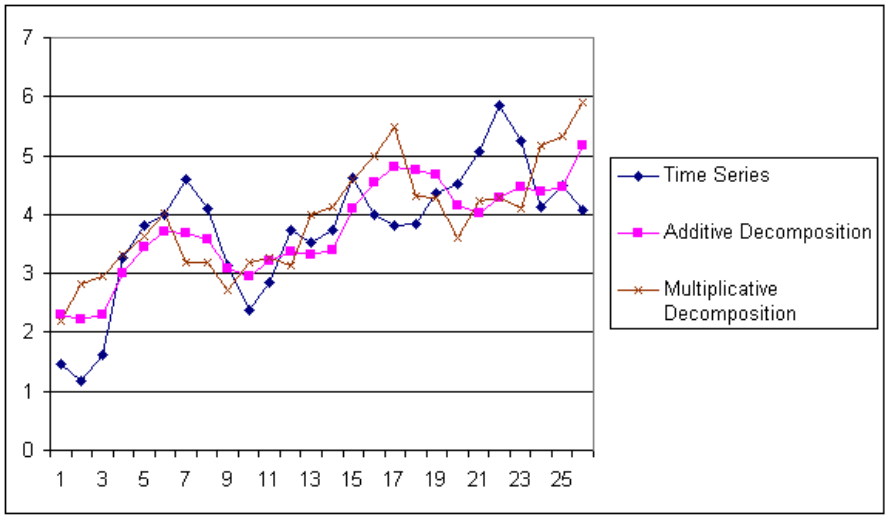


Рис.1.5. Прогнозування найпростішими методами

1.2.2 Регресійні методи прогнозування

Разом з описаними вище методами, заснованими на експоненціальному згладжуванні, вже достатньо довгий час для прогнозування використовуються регресійні методи. Коротко суть алгоритмів такого класу описано у роботі [5]. Існує прогнозована змінна *Y* (залежна змінна) і відібраний заздалегідь комплект змінних, від яких вона залежить, *X1, X2 ..., XN* (незалежні змінні). Природа незалежних змінних

може бути різною. Наприклад, якщо припустити, що Y – рівень попиту на деякий продукт в наступному місяці, то незалежними змінними можуть бути рівень попиту на цей же продукт в минулий і позаминулий місяці, витрати на рекламу, рівень платоспроможності населення, економічна ситуація, діяльність конкурентів тощо. Головне – уміти формалізувати всі зовнішні чинники, від яких може залежати рівень попиту в числовій формі.

У простішому варіанті лінійної регресійної моделі залежність залежної змінної від незалежних має вигляд:

$$Y=\beta 0+\beta 1X1+\beta 2X2+...+\beta NXN+\varepsilon, \quad (1.5)$$

тут $\beta 1,\beta 2,...,\beta N-\beta 1,\beta 2,...,\beta N$ − підібрані коефіцієнти регресії, ε − компонента помилки. Передбачається, що усі помилки незалежні і нормально розподілені. Для побудови регресійних моделей необхідно мати базу даних спостережень приблизно такого вигляду, як показано на рисунку 1.6.

	Змінні				
	Незалежні				Залежна
№	*X1*	*X2*	...	*XN*	*Y*
1	x_11	x_12	...	x_1N	Y_1
2	x_21	x_22	...	x_2N	Y_2
...	...	...	...	...	...
m	x_M1	x_M2	...	x_MN	Y_m

Рис. 1.6. База даних спостережень

За допомогою таблиці значень минулих спостережень можна підібрати (наприклад, методом найменших квадратів) коефіцієнти регресії, побудувавши тим самим модель.

При роботі з регресією треба дотримуватися певної обережності і обов'язково перевірити на адекватність знайдені моделі. Існують різні способи такої перевірки. Обов'язковим є статистичний аналіз залишків, тест Дарбіна-Уотсона. Корисно, як і у випадку з нейронними мережами, мати незалежний набір даних, на яких можна перевірити якість роботи моделі.

1.2.3 Метод Бокса-Дженкінса (ARIMA)

В середині 90-х років минулого століття був розроблений принципово новий і достатньо могутній клас алгоритмів для прогнозування часових рядів. Велику частину роботи по дослідженню методології і перевірці моделей була проведена двома статистиками, Г.Е. Боксом і Г.М. Дженкінсом. З тих пір побудова подібних моделей і отримання на їх основі прогнозів іноді називатися методами Бокса-Дженкінса. В це сімейство входить декілька алгоритмів, найвідомішим і використовуваним з них є алгоритм ARIMA. Він вбудований практично в будь-який спеціалізований пакет для прогнозування. У класичному варіанті ARIMA не використовуються незалежні змінні. Моделі спираються тільки на інформацію, що міститься в передісторії прогнозованих рядів, що обмежує можливості алгоритму. В даний час в науковій літературі часто згадуються варіанти моделей ARIMA, що дозволяють враховувати незалежні змінні. У даній доповіді вони розглядатись не будуть, обмежимось тільки загальновідомим класичним варіантом. На відміну від розглянутих раніше методів прогнозування часових рядів, у методології ARIMA не передбачається чіткої моделі для прогнозування. Задається лише загальний клас моделей, що описують часовий ряд і що дозволяють виражати поточне значення змінної через її попередні значення. Потім алгоритм, підлаштовуючи внутрішні параметри, сам обирає найбільш відповідну модель прогнозування. Існує ціла ієрархія моделей Бокса-Дженкінса.

ARIMA – це узагальнена модель авторегресійної моделі ковзної середньої, яка поєднує в собі авто регресивний (AR) процес та процес Ковзного середнього (MA), обробляє і створює складну модель часових рядів.

Модель розбивається на ключові частини (p,d,q) наступним чином:

AP: авторегресія. Регресійна модель, яка використовує залежності між спостереженням і числом відставання спостережень (p).

I: Інтегрування – для перетворення часового ряду в стаціонарний вимірюванням відмінностей спостережень в різний час (*d*).

MA: Ковзного середнього. Підхід, який враховує залежність між спостереженнями та умовою залишкової помилки, коли для спостережень використовується скоригована середня модель (*q*).

Проста форма AR моделі замовлення *p*, тобто AR(*p*), може бути записана у вигляді лінійного процесу, яке задано як:

$$x_t = c + \sum_{i=1}^{p} \emptyset_i x_{t-i} + \varepsilon_t \quad (1.6)$$

Де $\emptyset$ – ваги, що застосовуються до поточного та попередніх значень стохастичного терму в часовому ряді, $\emptyset_i = 1$. Ми припускаємо, що ε є послідовністю гаусівського білого шуму із середнім, що дорівнює нулю і дисперсією $\sum \varepsilon^2$[6]. Ми можемо об'єднати ці дві моделі разом, об'єднавши їх, і отримаємо модель ARMA порядку (*p,q*).

Метод ARIMA здатний мати справу із нестаціонарними даними часових рядів через його «інтегровані» кроки. Фактично «інтегрування» компонентів включає в себе диференціювання часових рядів для перетворення нестаціонарних часових рядів у стаціонарні. Загальна форма моделі ARIMA позначається як ARIMA (*p,d,q*).

З даними сезонних часових рядів, скоріше за все, на модель будуть впливати короткі несезонні параметри. Таким чином, ми повинні оцінити модель ARIMA, що працює з короткостроковими несезонними компонентами. Таким чином, модель можна визначити як:

$$AR(p)+MA(q) \rightarrow ARMA(p,q) \rightarrow ARMA(p,q)(P,Q) \rightarrow ARIMA(p,q,r)(P,Q,R) \quad (1.7)$$

1.2.4 Авторегресивна модель порядку p

Модель AR(p) має вигляд:

$$Y(t)=f0+f1 \cdot Y(t-1)+f2 \cdot Y(t-2)+...+fp \cdot Y(t-p)+E(t) Y(t)=f0+f1 \cdot Y(t-1)+f2 \cdot Y(t-2)+...+fp \cdot Y(t-p)+E(t), \quad (1.8)$$

де *Y(t)* – залежна змінна у момент часу *t, f0,f1,f2...,fpf0,f1,f2...,fp* –

оцінювані параметри, *E(t)* – помилка від впливу змінних, які не враховуються в даній моделі. Задача полягає в тому, щоб визначити *f0,f1,f2...,fpf0,f1,f2...,fp*. Їх можна оцінити різними способами. Найправильніше шукати їх через систему рівнянь Юла-Уолкера, для складання цієї системи буде потрібно розрахувати значення автокореляційної функції або методом найменших квадратів.

1.2.5. Штучна нейронна мережа

Нейронна мережа складається як мінімум з трьох шарів: 1) вхідний шар, 2) приховані шари і 3) вихідний шар. Кількість характеристик в наборі даних визначає розмірність та кількість вузлів у вхідному шарі. Ці вузли з'єднані через посилання, що називаються "синапсами", до вузлів, що знаходяться у прихованому шарі. Синапси посилання передають ваги для кожного вузла вхідного шару. Ваги допомагають приймати рішення, який сигнал або вхід може пройти і який не може. Ваги також показують силу до прихованого шару. Нейронна мережа в основному вчиться, коригуючи ваги для кожного синопсису.

У прихованих шарах вузли застосовують функцію активації (наприклад, сигмоїдну або гіперболічний тангенс) на зважені суми входів для перетворення вхідних даних на вихідні. Вихідний шар генерує вектор вірогідності для різних виходів і вибирає той, який з мінімальною похибкою або втратою, тобто мінімізує відмінності між очікуваними та прогнозованими значеннями, також відомі як вартість, використовуючи функцію SoftMax.

Щоб знайти найбільш оптимальні значення для вектору вагів, використовується метод зворотного поширення помилок. Від вихідного шару до прихованих шарів регулюються вектори вагів за допомогою стохастичного градієнтного спуску.

Для отримання більшої точності процедура навчання вагів повторюється.

Коли функція вартості мінімізується, модель навчається. Ітерації навчання нейронних мереж називаються епохами.

1.2.6 Рекурентні нейронні мережі

Рекурентна нейронна мережа (RNN) – це особливий тип нейронної мережі, де метою є прогнозування наступного кроку в послідовності спостережень щодо попередніх кроків послідовності. Головна ідея RNNs – це використання послідовних спостережень та навчання на попередніх етапах для прогнозування майбутніх тенденцій [7]. В результаті необхідно запам'ятовувати попередні дані для прогнозування даних в майбутньому. У RNN приховані шари діють як внутрішня пам'ять для зберігання інформації, що була зібрана на попередніх етапах обробки послідовностей даних. RNN називаються саме рекурентними, оскільки вони використовують попередні дані для прогнозування наступних даних. Архітектура простої рекурентної мережі, запропонованої Елманом, представлена на рис. 1.7.

Рекурентні нейронні мережі відрізняються від нейронних мереж прямого поширення саме зворотними зв'язками, пов'язаними з минулими рішеннями. Ці зв'язки отримують результати відразу після ведення даних.

Головна проблема класичних RNN полягає в тому, що ці мережі пам'ятають лише кілька попередніх кроків у послідовності і, таким чином, не підходять для запам'ятовування довгих послідовностей даних. Ця складна проблема вирішується за допомогою "рядка пам'яті", що вводиться в мережі – «довга короткочасна пам'ять» (LSTM).

1.2.7. Мережа «довга короткочасна пам'ять»

Довга короткострокова пам'ять (англ. Long short-term memory; LSTM [8]) – різновид архітектури рекурентних нейронних мереж (RNN), запропонована в 1997 році Сеппом Хохрайтером і Юргеном Шмидхубером. Як і більшість рекурентних нейронних мереж (РНМ), мережа «довга короткочасна пам'ять» (ДКЧП) є універсальною в тому сенсі, що за

достатньої кількості вузлів мережі вона може обчислювати будь-що, що може обчислювати звичайний комп'ютер, за умови, що вона має належну матрицю вагових коефіцієнтів, що може розглядатися як її програма. На відміну від традиційних рекурентних нейронних мереж, мережа ДКЧП добре підходить для навчання з досвіду з метою класифікації, обробки або передбачення часових рядів в умовах, коли між важливими подіями існують часові затримки невідомої тривалості.

Відносна нечутливість до довжини прогалин дає ДКЧП перевагу в численних застосуваннях над альтернативними РНМ, прихованими Марківськими моделями та іншими методами навчання послідовностей. Серед інших успіхів, ДКЧП досягла найкращих з відомих результатів у стисненні тексту природною мовою [9], розпізнаванні несегментованого неперервного рукописного тексту [10], і 2009 року виграла змагання з розпізнавання рукописного тексту ICDAR. Мережі ДКЧП також застосовувалися до автоматичного розпізнавання мовлення, і були головною складовою мережі, яка 2003 року досягла рекордного 17.7-відсоткового рівня фонемних похибок на класичному наборі даних природного мовлення TIMIT [11]. Станом на 2016 рік основні технологічні компанії, включаючи з Google, Apple, Microsoft та Baidu, використовують мережі ДКЧП як основні складові нових продуктів.

Мережа ДКЧП є штучною нейронною мережею, яка містить вузли ДКЧП замість, або на додачу, до інших вузлів мережі. Вузол ДКЧП – це вузол рекурентної нейронної мережі, який виділяється запам'ятовуванням значень для довгих, або коротких проміжків часу. Ключем до цієї здатності є те, що він не використовує функції активації в межах своїх рекурентних складових. Таким чином, значення, що зберігається, не розплющується ітеративно з плином часу, і член градієнту не має схильності розмиватися, коли для його тренування застосовується зворотне поширення у часі.

Вузли ДКЧП часто втілюються у «блоках», які містять декілька вузлів ДКЧП. Така конструкція є типовою для «глибоких» багатошарових

нейронних мереж і сприяє реалізаціям на паралельному апаратному забезпеченні.

Блоки ДКЧП містять три або чотири «вентилі» (англ. gates), які вони використовують для керування плином інформації до або з їхньої пам'яті. Ці вентилі реалізовані із застосуванням логістичної функції для обчислення значень між 0 та 1. Для часткового дозволяння або заборони плину інформації до або з цієї пам'яті застосовується множення на це значення. Наприклад, «входовий вентиль» (англ. input gate) керує мірою, до якої нове значення входить до пам'яті. «Забувальний вентиль» (англ. forget gate) керує мірою, до якої значення залишається в пам'яті. А «вентиль виходу» (англ. output gate) керує мірою, до якої значення в пам'яті використовується для обчислення активування виходу блоку. (В деяких утіленнях входовий та забувальний вентилі об'єднують в один. Ідея їхнього об'єднання полягає в тому, що час забувати настає тоді, коли з'являється нове значення, варте запам'ятовування.)

Архітектура рекурентної нейронної мережі зображена на рис. 1.7.

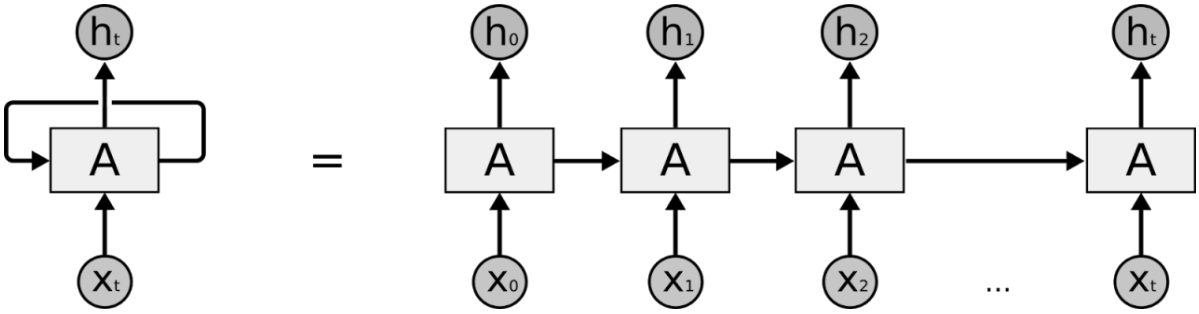


Рис. 1.7. Рекурентна нейронна мережа в розгортці

Дана мережа добре підходить для передбачення часових рядів, адже вона враховує “досвід”, який був раніше. Головною відмінністю LSTM від RNN є наявність вузлів, які виділяються для запам'ятовування значень для довгих проміжків часу [12]. Це відбувається за рахунок того, що такі вузли не використовують функції активації, що не змінює дані.

LSTM має форму ланцюга повторюваних модулів нейронної мережі.

На рис. 1.8. зображена структура LSTM, яка має чотири приховані шари та використовує сигмоїдальну функцію та гіперболічний тангенс в якості функції активації.

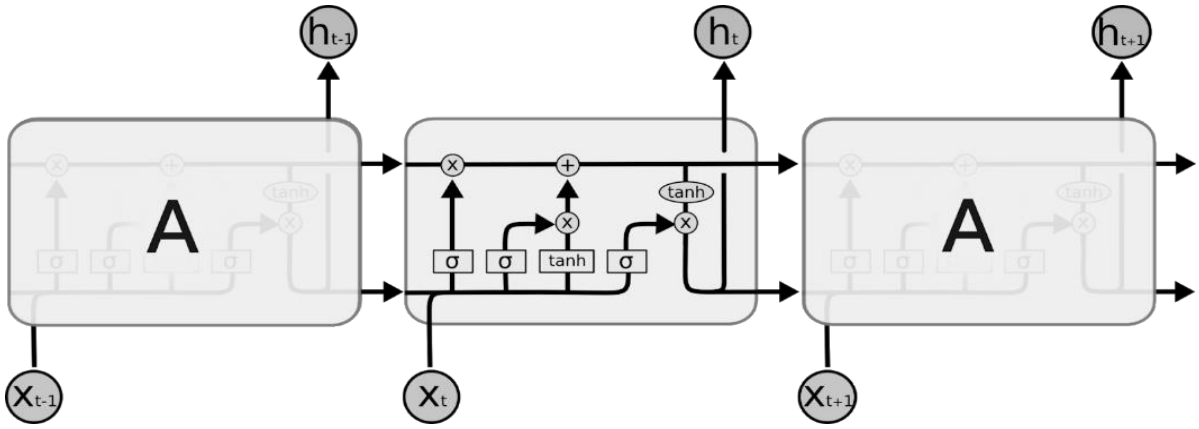


Рис. 1.8. Структура LSTM

Тренування нейронної мережі відбувається за допомогою зворотного поширення в часі, для якого використовується ітеративний градієнтний спуск, що змінює кожний ваговий коефіцієнт пропорційно до його похідної по відношенню до похибки [13].

Традиційна LSTM з вентилями забування (1.8 – 1.12): $c_0 = 0$, $h_0 = 0$,

$$f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (1.9)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (1.10)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (1.11)$$

$$c_t = f_t c_{t-1} + i_t \sigma_c(W_c x_i + U_c h_{t-1} + b_c) \quad (1.12)$$

$$h_t = o_t \sigma_h(c_t) \quad (1.13)$$

Змінні: x_t – вхідний вектор; h_t – вихідний вектор; c_t – вектор станів; W, U та b – матриці параметрів і вектори f_t, i_t та o_t – вектори вентилів: f_t – вентиля забування, ваги запам’ятовування старої інформації, i_t – вхідного вентиля, ваги отримання нової інформації, o_t – вихідного вентиля, кандидат на вихід.

Функції активації: σ_g – на основі сигмоїди, σ_c – на основі гіперболічного тангенсу, σ_h – на основі гіперболічного тангенсу, у даній роботі припускається, що $\sigma_h(x) = x$.

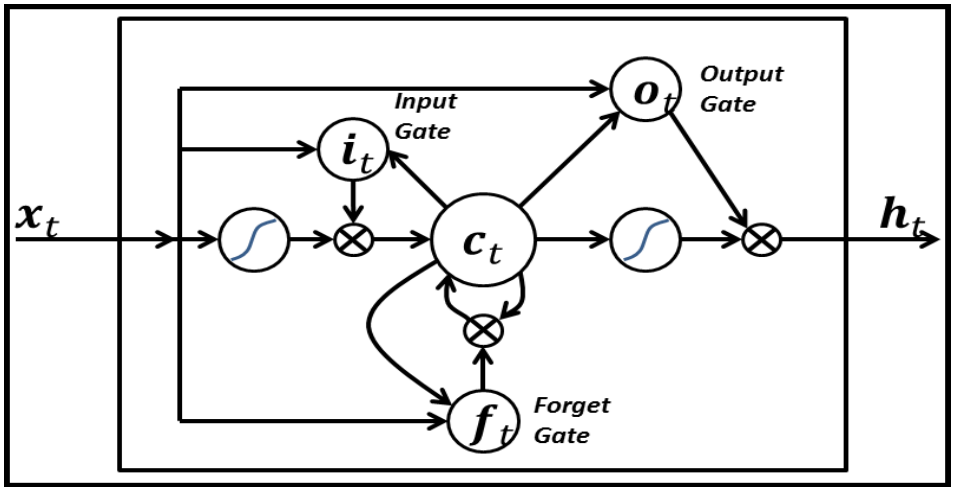


Рис.1.9. Простий LSTM-блок з трьома вентилями: вхідним, вихідним і забування

1.3. Висновок до розділу 1

У даному розділі проведено аналіз відомих програмних систем для управління автотранспортними підприємствами, розглянуті основні методи прогнозування пасажиропотоку. Розглянуто проблеми, що виникають при керуванні автотранспортними підприємствами та способи їх вирішення. Наведено нейронні мережі, які доцільно застосувати для вирішення сформульованої задачі.

При використанні рекурентних нейронних мереж алгоритм виконує менше попередньої обробки даних у порівнянні з іншими методами прогнозування. Це означає, що мережа самостійно знаходить необхідні фільтри, що в традиційних алгоритмах розроблялися вручну. Ця незалежність конструювання ознак від апріорних знань та людських зусиль є великою перевагою.

Також у даному розділі було обгрунтовано вибір та описано рекурентну нейронну мережу LSTM для прогнозування пасажиропотоку.

2. АРХІТЕКТУРА НЕЙРОННОЇ МЕРЕЖІ

Мережа LSTM є періодичною нейронною мережею, яка має клітинні блоки LSTM замість стандартних шарів нейронної мережі. Ці клітини мають різні компоненти, які називаються input gate, the forget gate та the output gate. Графічне зображення комірки LSTM на рис 2.1.

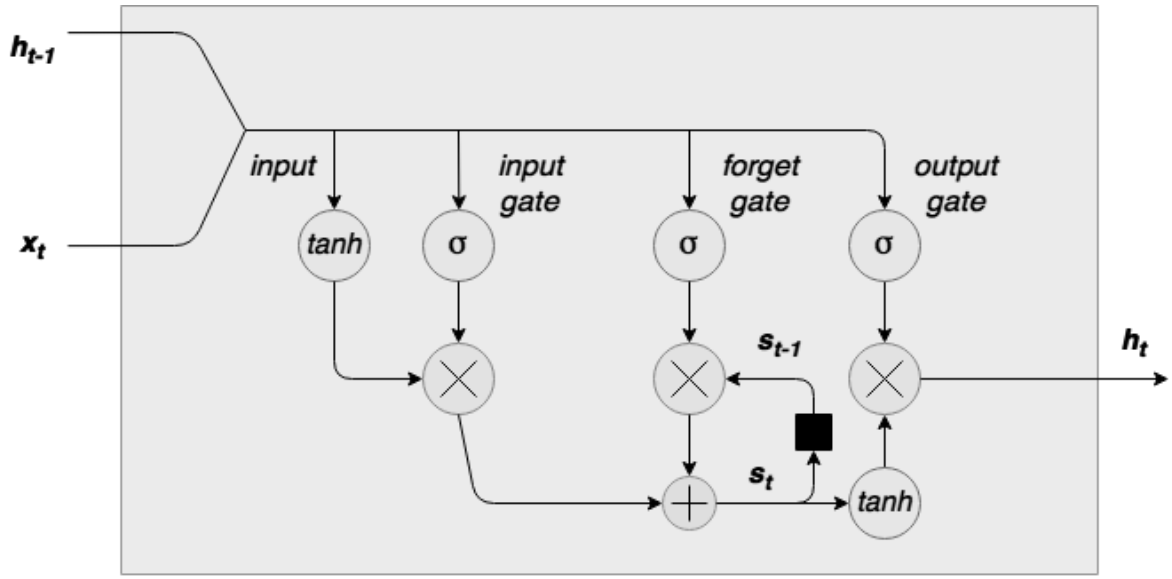


Рис. 2.1. LSTM cell diagram

З лівого боку ми маємо нове значення чи послідовності xt, об'єднане з попереднім виходом з комірки $ht-1$. Перший крок для цього комбінованого введення полягає в тому, щоб він був розібраний через tahn-шар. Другий крок полягає в тому, що цей вхід проходить через вхідні ворота. Вхідні ворота – це шар сигмоїдних активованих вузлів, результати яких множаться на розсічений вхід. Ці сигмоїди вхідних воріт можуть діяти, щоб «знищити» будь-які елементи вхідного вектора, які не потрібні [14]. Функція sigmoid виводить значення між 0 та 1, тому ваги, що з'єднують вхід з цими вузлами, можна навчити виводити значення близькі до нуля, щоб «виключити» певні вхідні значення (або, навпаки, виходи, близькі до 1, «проходять» інші цінності).

Наступним кроком в потоці даних є використання внутрішнього контура за замовчуванням. Клітини LSTM мають внутрішню зміну стану st.

Ця змінна відстає один раз, тобто s_{t-1} додається до вхідних даних, щоб створити ефективний шар повторення. Це операція додавання, а не операція множення, це допомагає зменшити ризик втрати градієнтів. Цей цикл повторення контролюється запам'ятовуючими воротами. Це працює так само, як і вхідні ворота, але допомагає мережі вивчити, які змінні стану слід «запам'ятати» або «забути».

Нарешті, у нас є вихідний рівень tanh функції, вихід якого контролюється вихідними воротами. Ці ворота визначають, які значення фактично допускаються як вихід з комірки h_t.

Вхідний рівень

По-перше, ввід розрівнюється між -1 та 1, використовуючи функцію активації *tanh*. Це може бути виражено таким чином:

$$g=\tanh(b_g+x_tU_g+h_{t-1}V_g), \quad (2.1)$$

де U_g та V_g є вагами для вхідного і попереднього вихідних комірок, відповідно, і b_g є вхідним зміщенням. Показники g не є підвищеними потужностями, а швидше означають, що це значення вхідного ваги та зміщення (на відміну від вхідних воріт, забуті ворота, вихідні ворота тощо).

Даний розподілений ввід потім множиться елементарним шляхом на виході вхідного входу, який, являє собою серію активованих вузлів сигмоїда:

$$i=\sigma(b_i+x_tU_i+h_{t-1}V_i) \quad (2.2)$$

Вихід із забутих воріт виражається як:

$$F=\sigma(b_f+x_tU_f+h_{t-1}V_f) \quad (2.3)$$

Вихід елементарного продукту попереднього стану та забутих воріт виражається як s_{t-1f} закінчення. Вихід із стадії забутих воріт:

$$s_t=s_{t-1f}+g_i \quad (2.4)$$

Таким чином, вихідні ворота, можуть бути показані як:

$$h_t=\tanh(s_t)o \quad (2.5)$$

2.1. LSTM модель

У цьому розділі буде показано, як виглядає повна архітектура LSTM, і відображається архітектура мережі [15]. Це далі висвітлить деякі ідеї, викладені вище, в тому числі вбудований шар і розміри тензора навколо мережі. Пропонована архітектура виглядає наступним чином на рис 2.2.

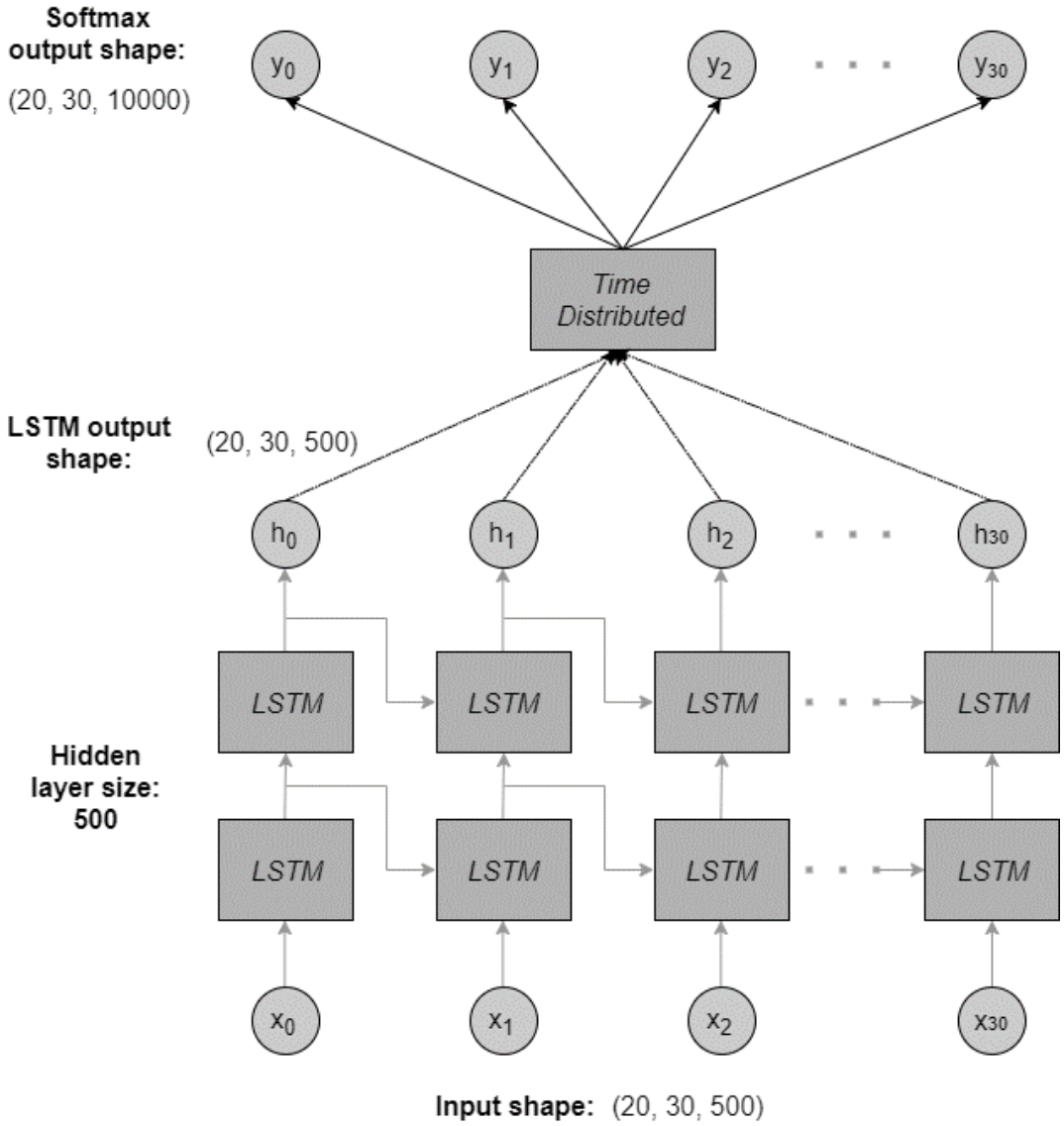


Рис. 2.2. Структура LSTM

Форма введення текстових даних упорядковується за розміром партії, кількістю етапів часу, прихованим розміром. Іншими словами, для кожного вибіркового зразка та кожного слова у кількості етапів часу, для введення

слова вводять вектор вектора вкладання 500 довжин. Ці вектори вбудовування будуть вивчатися як частина загальної моделі навчання. Вхідні дані потім надсилаються в два "складені" шари клітин LSTM (з 500 прихованих розмірів) – на схемі вище, мережа LSTM відображається як розгорнена по всіх етапах часу. Вихідні дані цих розгалужених клітин: розмір партії, кількість кроків часу, прихований розмір.

Вихідні дані потім передаються в шар який називається TimeDistributed, який більш повно пояснюється нижче. Нарешті, вихідний шар має прив'язану до неї активацію softmax. Цей вихід порівнюється з даними підготовки для кожної партії, а потік помилок і градієнта зворотного поширення виконується звідти. Навчальні дані у цьому випадку є входами *x*, які пройшли один час, тобто, на кожному кроці часу модель намагається передбачити саме наступне значення в послідовності [16]. Однак, це відбувається на кожному етапі, отже вихідний шар має таку ж кількість кроків часу, що і вхідний шар.

Щоб отримати прогноз у правильній формі для введення в модель LSTM, для кожного унікального значення в корпусі має бути присвоєний унікальний цілий індекс. Потім корпус повинен бути повторно сформований в порядку, але, а не значення ми маємо цілі ідентифікатори в порядку. Три функції, які роблять це в коді – read_appointment, build_appointment і file_to_ids. Кожне унікальне значення ідентифікується і призначається унікальним цілим числом. Нарешті, вихідний файл перетворюється в список цих унікальних цілих чисел, де кожне значення заміщується новим цілим ідентифікатором. Це дозволяє використовувати дані в нейронній мережі.

Функція load_data, для запуску цих функцій, показана нижче.

Лістинг 1. Функція load_data

```
def load_data ():
   # отримати шляхи даних
```

```
train_path = os.path.join (data_path, "ptb.train.txt")
valid_path = os.path.join (data_path, "ptb.valid.txt")
test_path = os.path.join (data_path, "ptb.test.txt")

# перетворюємо дані у список цілих чисел
appointment_to_id = build_appointment_list(train_path)
train_data = file_to_ appointment_ids (train_path, word_to_id)
valid_data = file_to_ appointment_ids (valid_path, word_to_id)
test_data = file_to_ appointment_ids (test_path, word_to_id)
appointment_list = len (appointment_to_id)
reversed_ appointment_list = dict (zip (appointment_to_id.values (), appointment_to_id.keys ()))

print(train_data[:5])
print(appointment_to_id)
print(appointment_list)
print(" ".join([reversed_appointment_list[x] for x in train_data[:10]]))
return train_data, valid_data, test_data, appointment_list, reversed_appointment_list
```

Щоб викликати цю функцію, ми можемо запустити:

```
train_data, valid_data, test_data, appointment_list, reversed_appointment_list = load_data ()
```

Три виходи цієї функції – дані тренувань, дані перевірки та дані тесту з набору даних, відповідно, але кожен запис відображається у вигляді цілого числа в списку. Деяка інформація друкується під час запуску load_data (), одна з яких є друком (train_data [: 5]). Це видає наступний вивід:

```
[9970, 9971, 9972, 9974, 9975]
```

Тренувальні дані складаються зі списку цілих чисел, як очікувалося.

Далі, вихідний appointment_list – це розмір нашого текстового корпусу. Коли записи входять до навчальних даних, кожен унікальний запис не розглядається. У цьому випадку N = appointment_list = 10 000.

Нарешті, reversed_appointment_list – це словник Python, де ключ є унікальним цілим ідентифікатором запису, а пов'язане значення – це запис у тексті. Це дозволяє нам працювати назад від прогнозованих цілих записів, які будуть прогнозовані нашою моделлю, і переводити їх у реальні дані. Наприклад, наступний код перетворює цілі числа в train_data у запис, який потім друкується: print (''. Join ([reversed_appointment_list [x] для x в train_data [100: 110]])).

2.2. LSTM генератори даних

Під час тренування нейронних мереж ми, як правило, передаємо дані до них невеликими частинами, які називаються міні-партіями або просто «batches». Keras має деякі зручні функції, які можуть автоматично витягати дані навчання з попередньо встановленого об'єкта ітератора чи генератора Python і вводити його в модель. Один з цих функцій Keras називається fit_generator. Першим аргументом fit_generator є функція ітератора Python, яку ми створимо, і вона буде використовуватися для вилучення партій даних під час тренувального процесу. Ця функція в Keras буде обробляти всі дані, вводити в модель, виконувати градієнтні кроки, вести журнал показників, таких як точність і виконання зворотних викликів. У цьому випадку я створив клас генератора, який містить метод, який реалізує таку структуру. Ініціалізація цього класу виглядає так:

Лістинг 2. Фрагмент коду «батч» генератора

```
class KerasBatchGenerator(object):
    def __init__(self, data, num_steps, batch_size, appointment_list, skip_step=5):
        self.data = data
        self.num_steps = num_steps
```

```
        self.batch_size = batch_size
        self. appointment_list = appointment_list
        # this will track the progress of the batches sequentially through the
        # data set - once the data reaches the end of the data set it will reset
        # back to zero
        self.current_idx = 0
        # skip_step is the number of words which will be skipped before the
next
        # batch is skimmed from the data set
        self.skip_step = skip_step
```

Тут об'єкт KerasBatchGenerator приймає дані як перший аргумент. Ці дані можуть бути як тренінгові, перевіряючі, так і тестові дані. Декілька екземплярів одного і того ж класу можуть бути створені та використані на різних етапах циклу розробки нашого машинного навчання: підготовки, перевірки валідації, тестування. Наступний аргумент називається num_steps – це кількість слів, які ми будемо подавати у час, розподілений вхідний шар мережі. Інакше кажучи, це набір слів, з якими модель буде вчитися, щоб передбачити пасажирів. Аргумент batch_size є досить зрозумілим. Нарешті, skip_steps – це кількість записів, які ми хочемо пропустити між зразками навчань у кожній партії. Нарешті, skip_steps – це кількість записів, які потрібно пропустити, перш ніж приймати наступний пакет даних. Якщо в цьому прикладі це skip_steps = num_steps,

Ініціалізація класу – це змінне current_idx, яке ініціалізується при нулі. Ця змінна необхідна для відстеження вилучення даних через повний набір даних. Після того, як повний набір даних був використаний під час навчання, нам потрібно скинути значення current_idx до нуля, щоб використання даних почалося з початку набору даних знову. Іншими словами, це вказівник місця розташування даних.

Лістинг 3. Фрагмент коду fit_generator

```
def generate(self):
    x = np.zeros((self.batch_size, self.num_steps))
    y = np.zeros((self.batch_size, self.num_steps, self.appointment_list))
    while True:
        for i in range(self.batch_size):
            if self.current_idx + self.num_steps >= len(self.data):
                # reset the index back to the start of the data set
                self.current_idx = 0
            x[i, :] = self.data[self.current_idx:self.current_idx + self.num_steps]
            temp_y = self.data[self.current_idx + 1:self.current_idx + self.num_steps + 1]
            # convert all of temp_y into a one hot representation
            y[i, :, :] = to_categorical(temp_y, num_classes=self.vocabulary)
            self.current_idx += self.skip_step
        yield x, y
```

У першій парі рядків створені наші *x* і *y* вихідні масиви. Щоб зрозуміти, що перший параметр – це кількість зразків, які ми вказуємо в пакеті [17]. Другий вимір – це кількість записів, на яких ми будемо бачити наші прогнози. Розмір змінної *y* трохи складніший. По-перше, він має розмір партії як перший розмір, тоді він має кількість етапів часу, як другий, як описано вище.

Тепер на той час True: вихід *x, y* парадигми, що обговорювався раніше для генератора. У першому рядку ми входимо в цикл для розміру batch_size, щоб заповнити всі дані в пакеті. Далі існує умова перевірки того, чи потрібно скинути покажчик current_idx. Якщо поточна точка індексу плюс num_steps більша, ніж довжина набору даних, то покажчик current_idx повинен бути скинутий до нуля, щоб почати знову з набором даних.

Після виконання цієї перевірки вхідні дані використовуються в масиві x. Використовувані показники даних є досить простими для розуміння – це поточний індекс поточного індексу, плюс num_steps – кількість слів [18]. Потім заповнюється тимчасова *y*-змінна, яка практично однаково працює, єдиною відмінністю є те, що відправна точка та кінцева точка використання даних збільшуються на 1.

Тепер, коли створено клас генератора, нам потрібно створити його екземпляри. Як згадувалося раніше, ми можемо налаштувати екземпляри одного і того ж класу відповідно до даних навчання та перевірки. У коді це виглядає наступним чином:

Лістинг 4. Фрагмент коду

```
train_data_generator = KerasBatchGenerator(train_data, num_steps, batch_size, appointment_list,
                                           skip_step=num_steps)
valid_data_generator = KerasBatchGenerator(valid_data, num_steps, batch_size, appointment_list,
                                           skip_step=num_steps)
```

Тепер, коли вхідні дані для нашого LSTM коду налаштовані і готові, потрібно створювати саму мережу LSTM.

2.3. LSTM структура

Послідовна методологія дозволяє легко складати шари у нейронну мережу, не заважаючи на тензиси (та їх форми), що використовуються у моделі. Цей процес виглядає наступним чином:

Лістинг 5. Фрагмент коду

```
model = Sequential()
model.add(Embedding(appointments_list, hidden_size, input_length=num_steps))
model.add(LSTM(hidden_size, return_sequences=True))
```

```
model.add(LSTM(hidden_size, return_sequences=True))
if use_dropout:
   model.add(Dropout(0.5))
model.add(TimeDistributed(Dense(vocabulary)))
model.add(Activation('softmax'))
```

Перший крок передбачає створення моделі з конструктором Sequential (). Перший шар в мережі, як показано на схемі архітектури, є шаром вкладання записів. Це перетворить наші записи (посилання на цілі числа в даних) на значущі вектори вкладу. Цей шар Embedding () приймає кількість пасажирів як перший аргумент, а потім розмір результуючого вкладеного вектора як наступний аргумент. Нарешті, оскільки цей шар є першим шаром у мережі, ми повинні вказати "довжину" входу, тобто кількість кроків у кожному зразку.

Необхідно відстежувати форму тензора у мережі. У цьому випадку вхід до шару вкладу (batch_size, num_steps), а вихід (batch_size, num_steps, hidden_size). У Sequential моделі, завжди підтримується розмір партії як перший розмір. Вона отримує розмір партії від функції підключення (наприклад, fit_generator у цьому випадку), і тому це рідко.

Наступний шар – це перший з наших двох шарів LSTM. Щоб вказати рівень LSTM, спочатку потрібно вказати кількість вузлів у прихованих шарах у комірці LSTM. Наприклад, кількість клітин у кроці забутих воріт, вхідний шар tan і так далі. Наступний аргумент, вказаний у наведеному вище коді, - return_sequences = True argument. Це гарантує, що клітина LSTM повертає всі виходи із розібраної клітини LSTM через час. Якщо цей аргумент не вийшов, клітина LSTM просто забезпечить вихід клітинки LSTM з останнього етапу часу.

Як видно з наведеної вище схеми, є лише один вихід, коли «return_sequences = False - $ h_t $». Однак, коли return_sequences = True, всі розгорнуті виходи із клітин LSTM повертаються *h0 ... ht*. У цьому випадку

потрібне останнє розташування. При тренуванні моделі потрібно порівнювати виведення комірок LSTM на кожному кроці з наступним словом у послідовності, таким чином, ми отримуємо джерела num_steps для виправлення помилок у моделі (через зворотне поширення), а не тільки для кожного зразка.

Тому для обох штатних LSTM шарів ми повертаємо усі послідовності. Форма виводу кожного шару LSTM (batch_size, num_steps, hidden_size).

Наступний шар у нашій мережі LSTM – відсівний шар для запобігання перевизначення. Після цього існує спеціальний шар для використання в періодичних нейронних мережах під назвою TimeDisributed. Ця функція додає незалежний шар для кожного етапу в періодичній моделі. Так, наприклад, якщо у моделі є 10 етапів часу, TimeDistributed шар, який працює на щільному шарі, буде виробляти 10 незалежних шарів щільності, по одному для кожного етапу часу. Активація для цих щільних шарів встановлюється як softmax у фінальному шарі нашої моделі LSTM [19].

Наступний крок у Keras після завершення вашої моделі – запустити команду компіляції на моделі. Це виглядає так:

Лістинг 6. Фрагмент коду

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])

У цій команді необхідно вказати тип втрат, який Keras повинен використовувати для навчання моделі. У цьому випадку ми використовуємо "categorical_crossentropy", яка застосовується у випадках, коли існує безліч класів або категорій, з яких лише один є вірним. Далі в цьому прикладі оптимізатор, який буде використовуватися, – це оптимізатор adam – ефективний оптимізатор «all round» з адаптивним кроком. Нарешті, зазначено метрику – "categorical_accuracy", що може дозволити зрозуміти, наскільки точність під час тренувань поліпшується.

Наступний рядок коду передбачає створення зворотного виклику Keras. Це певні функції, які Keras може додатково викликати, як правило, після завершення навчальної епохи.

Зворотний виклик, який використовується в цьому прикладі, – це зворотний виклик моделі контрольного пункту, він зберігає модель після кожної епохи, що може бути зручним для довготривалих тренувань.

Лістинг 7. Фрагмент коду

```
checkpointer = ModelCheckpoint(filepath=data_path + '/model-{epoch:02d}.hdf5', verbose=1)
```

Кінцевим кроком у навчанні моделі Keras LSTM є виклик вищезазначеної функції fit_generator. Наведена нижче лінія показує, як це зробити:

Лістинг 8. Виклик fit_generator

```
model.fit_generator(train_data_generator.generate(), len(train_data)//(batch_size*num_steps), num_epochs,
                    validation_data=valid_data_generator.generate(),
                    validation_steps=len(valid_data)//(batch_size*num_steps), callbacks=[checkpointer])
```

Перший аргумент fit_generator – це функція генератора, яка була пояснена раніше. Наступним аргументом є кількість ітерацій для кожної навчальної епохи. Значення len (train_data) // (batch_size * num_steps) гарантує, що весь набір даних виконується за допомогою моделі в кожну епоху. Аналогічним чином, викликається генератор для меншого набору даних перевірки, з тим же аргументом для кількості ітерацій для запуску. Наприкінці кожної епохи дані перевірки будуть проходити через модель, і точність буде змінена. Нарешті, зворотний виклик типу контрольної точки, описаний вище, постачається через аргумент зворотних викликів у fit_generator.

Для того, щоб отримати хороші результати, доведеться запускати модель протягом багатьох епох, і модель повинна мати значний рівень

складності. Це займе багато часу на процесорній машині, тому я рекомендую запускати його на машині з хорошим графічним процесором. За відсутності комп'ютера GPU можна створити екземпляр Amazon EC2 [20]. Було запущено модель до 40 епох і я отримав деякі початкові результати. Мої параметри моделі для наведених нижче результатів є такими:

- num_steps=30
- batch_size=20
- hidden_size=500

Після 40 епох, точність набору навчальних даних становила близько 98%, а точність встановлення перевірки становила приблизно 96%.

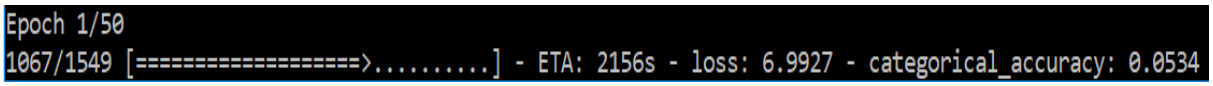


Рис. 2.3. Вигляд вікна тренування

Для того, щоб протестувати навчену модель LSTM, було порівняно прогнозовані результати поїздок із фактичними поїздками у наборі даних для тренування та тестування. Нижче наведений код – це фрагмент способу як це зробити, де порівнюються дані прогнозованої вихідної моделі та набору даних навчання (те ж саме можна зробити з даними test_data).

Лістинг 9. Модель

```
model = load_model(data_path + "\model-40.hdf5")
dummy_iters = 40
example_training_generator = KerasBatchGenerator(train_data, num_steps, 1, vocabulary,
                                                  skip_step=1)
print("Training data:")
for i in range(dummy_iters):
    dummy = next(example_training_generator.generate())
```

```
num_predict = 10
true_print_out = "Actual appointment: "
pred_print_out = "Predicted appointments: "
for i in range(num_predict):
    data = next(example_training_generator.generate())
    prediction = model.predict(data[0])
    predict_appointment = np.argmax(prediction[:, num_steps-1, :])
    true_print_out += reversed_appoinment_list[train_data[num_steps + dummy_iters + i]] + " "
    pred_print_out += reversed_appointment_list[predict_appointment] + " "
print(true_print_out)
print(pred_print_out)
```

У наведеному вище коді спочатку модель перезавантажується з навчених даних (у наведеному вище прикладі це контрольна точка з 40-ї епохи навчання). Потім створюється ще один клас KerasBatchGenerator, як це було описано раніше. У цьому випадку використовується партія довжини 1, щоб порівнювалося лише один елемент num_steps даних. Потім створюється цикл вилучення фіктивних даних з генератора – для того, щоб контролювати, де в наборі даних вказуються порівняльні пропозиції. Другий цикл, від 0 до num_predict.

З даних генератора вилучається партія даних, яка передається методу model.predict (). Це повертає кількість чисел стосовно передбачуваних поїздок.

Нижче наведено на рис. 2.4. результати порівняння фактичних та прогнозованих поїздок після 10 епох навчання на наборі навчальних даних:

```
Actual appointments: 10 16 25 30 50 17 19
Predicted appoitnments: 10 14 23 31 50 35 19
```

Рис. 2.4. Результати прогнозування після 10 епох навчання

Розглянемо порівняння результатів після 40 епох тренувань (рис. 2.5) на наборі навчальних даних:

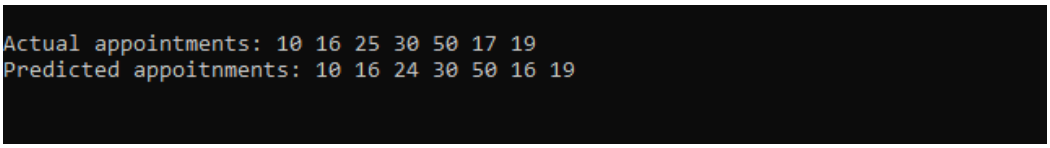


Рис. 2.5. Результати після 40 епох тренування

Ми отримали непогану відповідність між фактичними та передбачуваними поїздками у наборі тренувань.

Отже, розроблена мережа LSTM може добре працювати на реальних великих наборах даних і далі не потрібно використовувати додаткові тренінги.

2.4. Висновок до розділу 2

У даному розділі запропоновано метод покращення прогнозу пасажиропотоку та загальну архітектуру нейронної мережі.

Розглянуто загальну концепцію виконання прогнозування пасажиропотоку на основі статистичних даних за попередній період з використанням нейронної мережі LSTM.

Для тестування навченої моделі LSTM порівняно прогнозовані результати поїздок із фактичними поїздками у наборі даних про тренування та тестування та підтверджено ефективність використання даної моделі.

3. РОЗРОБКА ПРОГРАМНОГО МЕТОДУ ОПТИМІЗАЦІЇ МІЖМІСЬКИХ ПЕРЕВЕЗЕНЬ

Для підвищення якості обслуговування пасажирів та конкурентоспроможності АТП виникає потреба у створенні зворотного зв'язку із пасажирами.

Крім основної мети – перевезення пасажирів, забезпечення комфортної подорожі для пасажирів, для АТП також є ключовим питання отримання максимального прибутку. Забезпечення комфортним перевезення та ефективне планування автобусів має важливе значення для будь-якого АТП. Традиційні підходи базуються на фіксованих графіках.

Використання запропонованого програмного методу для обробки даних про міжміські пасажирські перевезення в режимі реального часу забезпечить нові можливості та підходи, керовані даними, для задоволення попиту пасажирів.

3.1. Вимоги до системи

3.1.1. Вимоги користувача

При розробці даної системи передбачалося, що кінцевий користувач (диспетчер АТП) має такі можливості:

- додавати, редагувати та видаляти записи про поїздки;
- додавати, редагувати інформацію про водіїв;
- переглядати та редагувати розклад;
- прогнозувати кількість пасажирів на необхідний час, дату та маршрут;
- моделювати найзручніший та найбільш вигідний рухомий склад.

3.1.2. Функціональні вимоги

1. Система має надавати результат прогнозування у відповідному вікні.

2. Система повинна надавати можливість вводити необхідні дані за допомогою полів у відповідному вікні.
3. Користувач повинен мати можливість вибирати необхідну дату, час та маршрут за допомогою селекторів у відповідному вікні.
4. Дані для навчання повинні зчитуватися з файлу типу csv.
5. У вікні розкладу система має показувати всі заплановані та виконані поїздки.
6. При натисканні на порожнє вікно має відбуватися створення поїздки та можливість прогнозувати кількість пасажирів на даний час.

3.1.3. Опис можливостей системи

Для того, щоб більш докладно розповісти про можливості, що надає система, були розроблена діаграма прецедентів.

Діаграма прецедентів – це граф, що складається з множини акторів, прецедентів (варіантів використання), відношень між акторами та прецедентами [21].

Розглянемо діаграму прецедентів наведену на рис. 3.1. Користувач може виконувати наступні дії: задавати вхідні дані для роботи програми, що, в свою чергу, включає в себе такі прецеденти:

- авторизація в системі;
- додавання запису про поїздку;
- прогнозування пасажиропотоку на обрану дату та час;
- редагування записів про поїздку;
- видалення непотрібних записів;
- перегляд розкладу рухомого складу автопідприємства;
- перегляд розкладу за обраним водієм окремо.

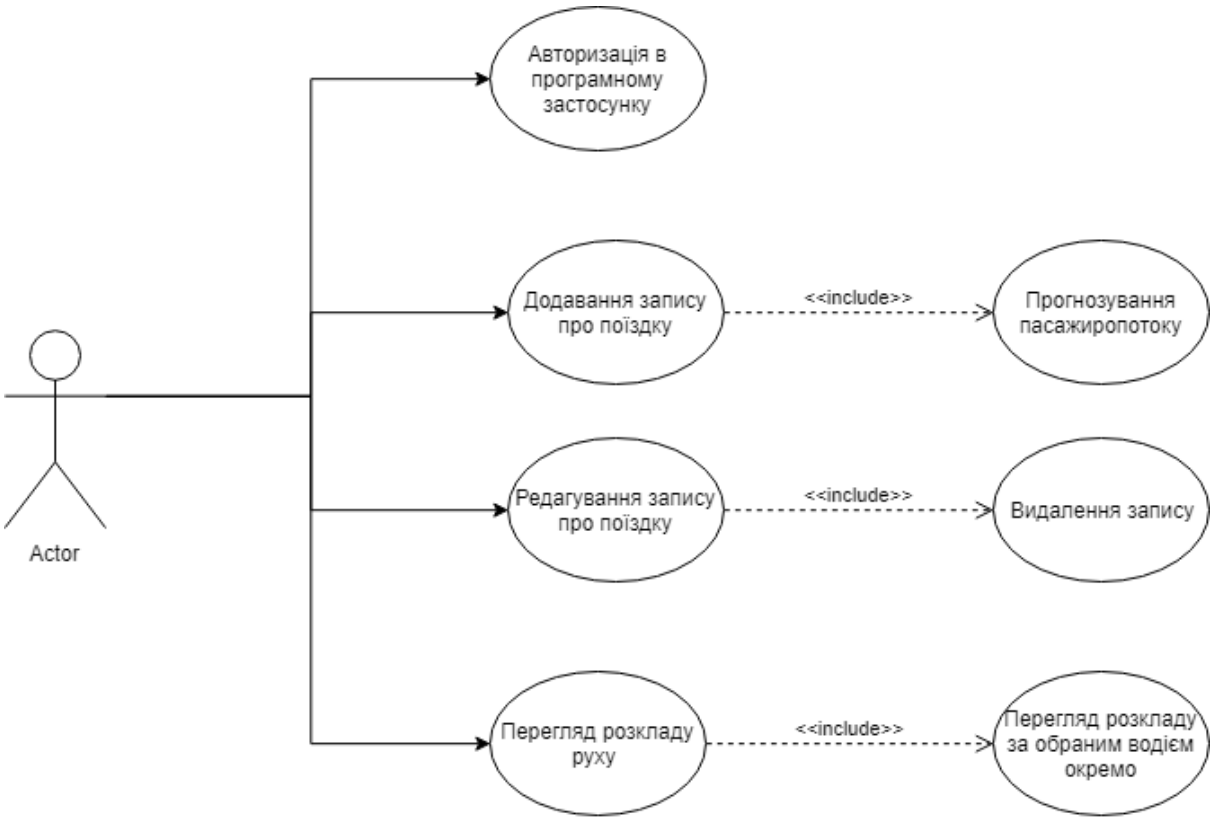


Рис. 3.1. Діаграма прецедентів

3.1.4. Архітектура системи програмного забезпечення

Архітектура системи наведена на рис. 3.2. В архітектурі системи можна виокремити наступні компоненти:

1. сервер для роботи з записами, базою даних та нейронною мережею;
2. інтерфейс користувача – компонент, яким користується користувач для створення поїздок та прогнозування;
3. нейронна мережа – компонент методу, який викликається з серверу для прогнозування пасажиропотоку.

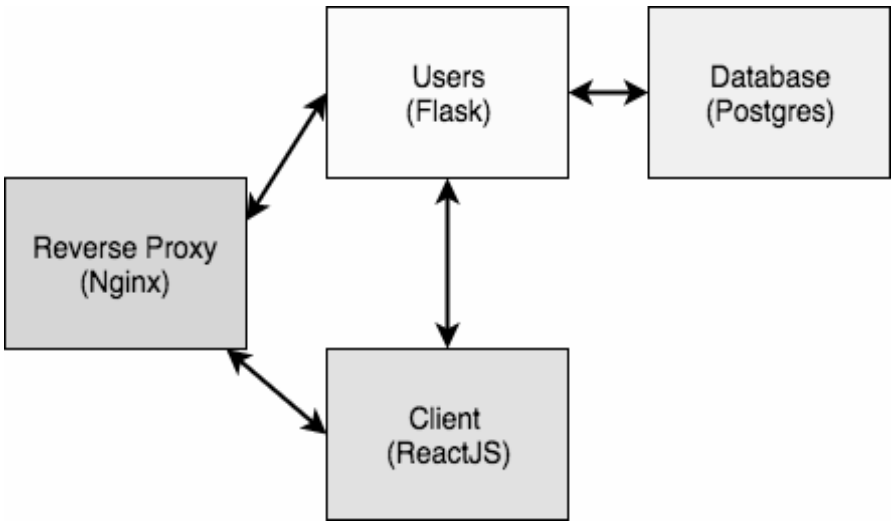


Рис. 3.2. Архітектура системи

3.2. Опис програмних засобів

3.2.1. Мова програмування

Для створення програмного забезпечення було використано мову програмування Python 3.6.

Python – це інтерпретована, високорівнева мова програмування. Python має динамічний тип системи та автоматичне керування пам'яттю. Він підтримує декілька парадигм програмування, включаючи об'єктно-орієнтовану, імперативу, функціональну та процедурну, має велику стандартну бібліотеку.

З часом величезна спільнота навколо цієї відкритої мови створила досить багато інструментів для ефективної роботи з Python. В останні роки було створено цілий ряд інструментів спеціально для наукових досліджень. В результаті аналіз даних з Python став досить простим і ефективним.

3.2.2. Середовище розробки

В якості середовища роробки було обрано Jupyter Lab. JupyterLab дозволяє працювати з документами та діями, такими як Jupyter Notebook, текстовими редакторами, терміналами та спеціальними компонентами, гнучким, інтегрованим та розширюваним способом. Jupyter Lab дозволяє

організувати декілька документів та заходів поряд з робочою зоною за допомогою вкладок. Документи та дії інтегруються один з одним, надаючи нові робочі процеси для інтерактивних обчислень.

Jupyter Lab також пропонує уніфіковану модель для перегляду та обробки форматів даних. Jupyter Lab розуміє багато форматів файлів (зображення, CSV, JSON, Markdown, PDF, Vega, Vega-Lite та ін.), а також може відображати великий вихід ядра в цих форматах.

Для навігації по користувацькому інтерфейсу Jupyter Lab пропонує налаштовувані комбінації клавіш та можливість використовувати ключові карти з vim, emacs та Sublime Text у текстовому редакторі.

Розширення Jupyter Lab можуть налаштовувати або покращувати будь-яку частину Jupyter Lab, включаючи нові теми, редактори файлів та спеціальні компоненти.

3.2.3. Pandas

Pandas – це інструмент маніпулювання високим рівнем даних, розроблений Уесом Маккінні. Вона побудована на пакеті Numpy, і її ключова структура даних називається DataFrame. DataFrames дозволяють зберігати та управляти табличними даними у рядках спостережень та стовпціях змінних.

3.2.4. Numpy

NumPy – це універсальний пакет обробки масивів, розроблений для ефективного маніпулювання великими багатомірними масивами довільних записів, не жертвуючи надто великою швидкістю для малих багатомірних масивів. NumPy побудований на базі цифрових кодів і додає функції, представлені numarray, а також розширеною C-API і можливістю створення масивів довільного типу, що також робить NumPy придатним для взаємодії з універсальними додатками бази даних.

3.2.5. Scikit-learn

Scikit-learn – безкоштовна бібліотека машинного навчання для мови програмування Python. Вона має різні алгоритми класифікації, регресії та

кластеризації, включаючи метод опорних векторів, випадкові ліси, градієнтні підсилювачі, *k*-засоби та DBSCAN, і призначені для взаємодії з чисельними та науковими бібліотеками Python NumPy та SciPy.

3.3. Інтерфейс користувача

3.3.1. Опис вікна входу

На рис 3.3. наведено вікно входу, логіну до програми.

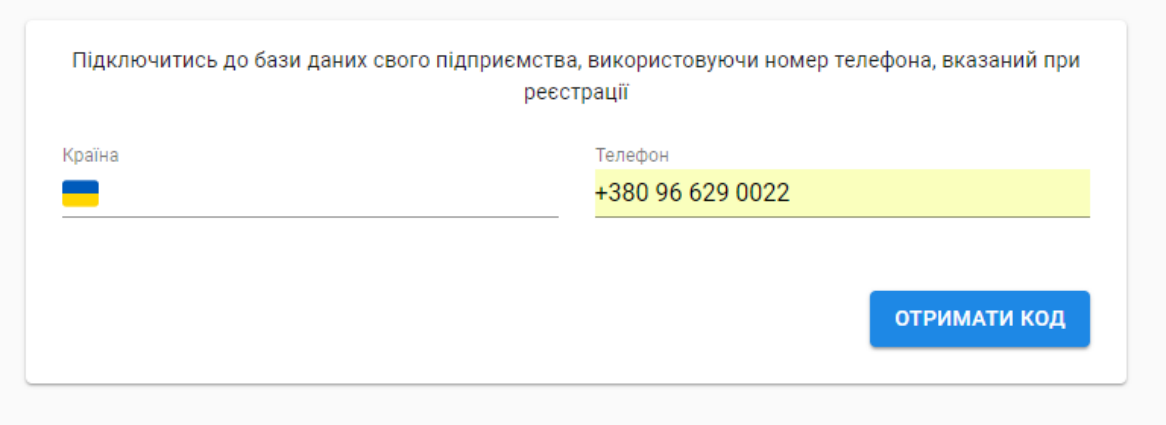


Рис. 3.3. Вікно логіну

Для того, щоб користувач міг ввійти, він повинен ввести зареєстрований в базі номер та отримати код підтвердження в СМС, як показано на рис. 3.4.

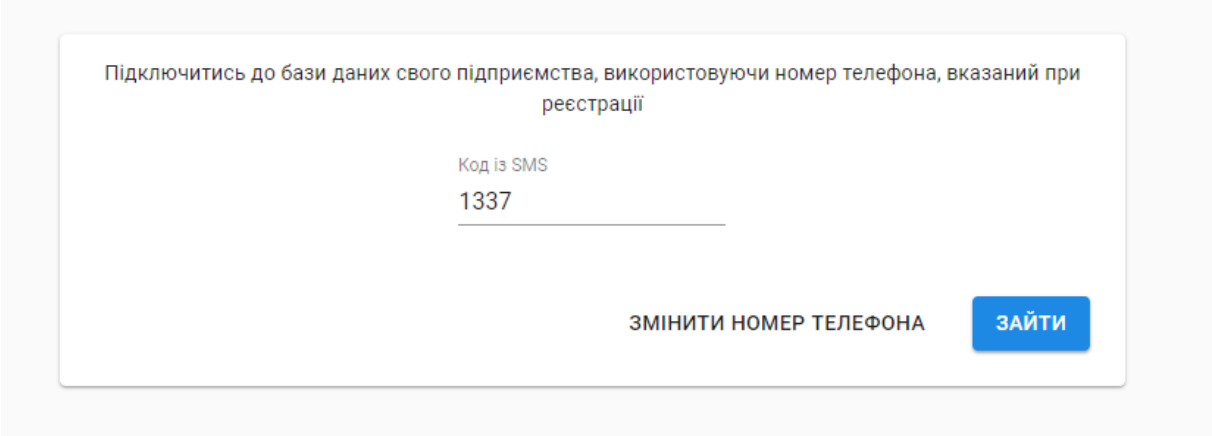


Рис. 3.4. Друге вікно логіну

Після цього користувач зможе ввійти до своєї програми. Двохетапна аутентифікація запобігає від крадіжок, та забезпечує безпеку даних.

3.3.2. Опис панелі навігації

Потрапивши до програми, перед користувачем з’явиться головне вікно та розкладу та панелі навігації, як показано рис. 3.5.

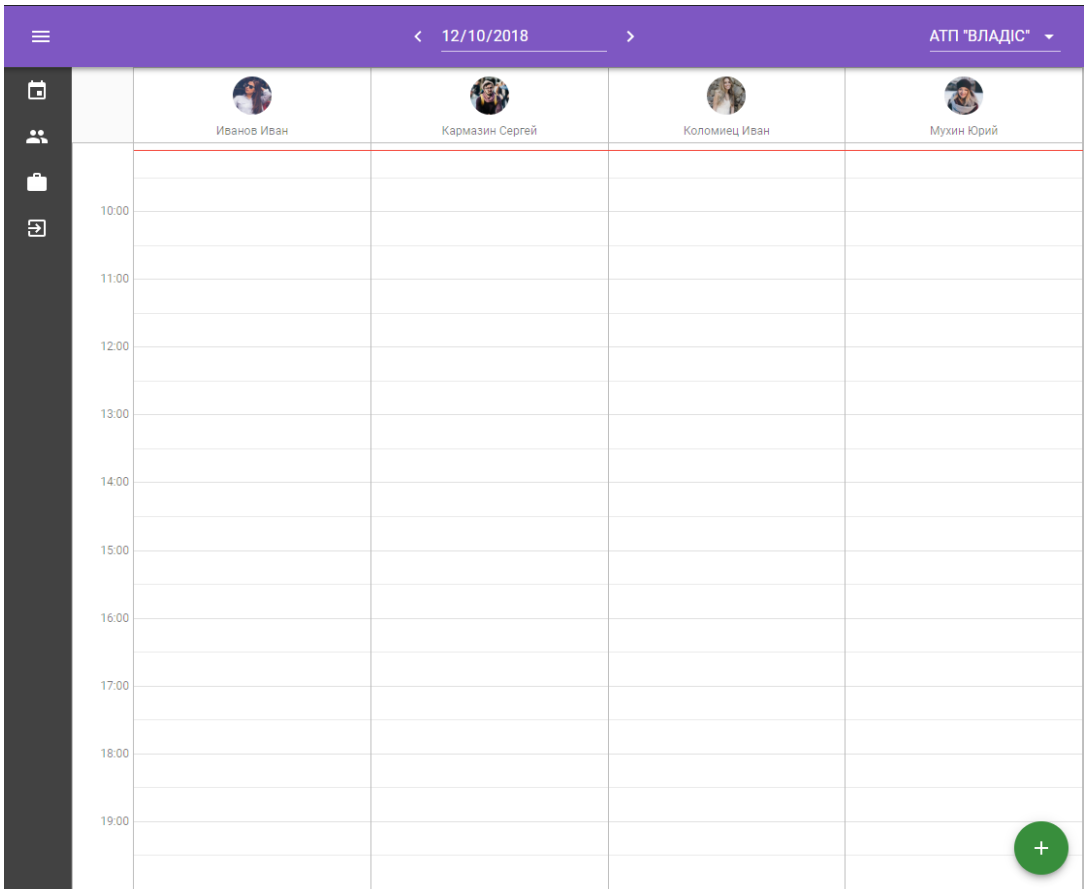


Рис. 3.5 Головне вікно з кнопками навігації

Зліва користувач може швидко переходити між вікнами Розкладу, Водіїв, Поїздок, відповідно. Зверху для адміністратора підприємства є зручна панель навігації за днями. Також зверху справа є селектор для вибору різних підприємств, це дуже зручно, якщо уповноважена особа відразу керує декількома закладами та може швидко між ними перемикатись.

3.3.3. Опис головного вікна розкладу

Вікно наведена на рис. 3.6.

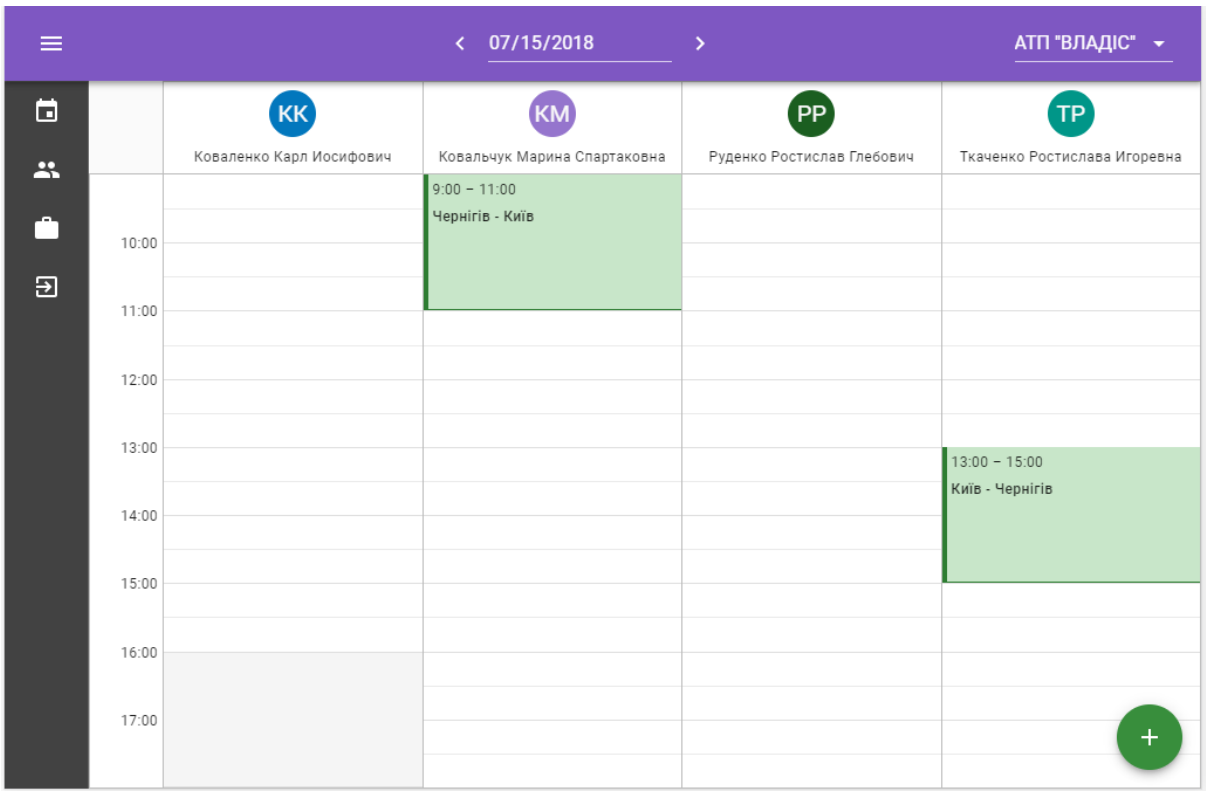


Рис. 3.6. Заповнений розклад

На даному вікні користувач може:

1. переглядати інформацію по розкладу автопарку;
2. створювати нові записи в обраному часі;
3. редагувати необхідні записи про поїздки;

При натисканні на кнопок меню користувач зможе перейти на вікно «Водії», «Запис» та «Прогнозування». Вікно запису показано на рис. 3.7.

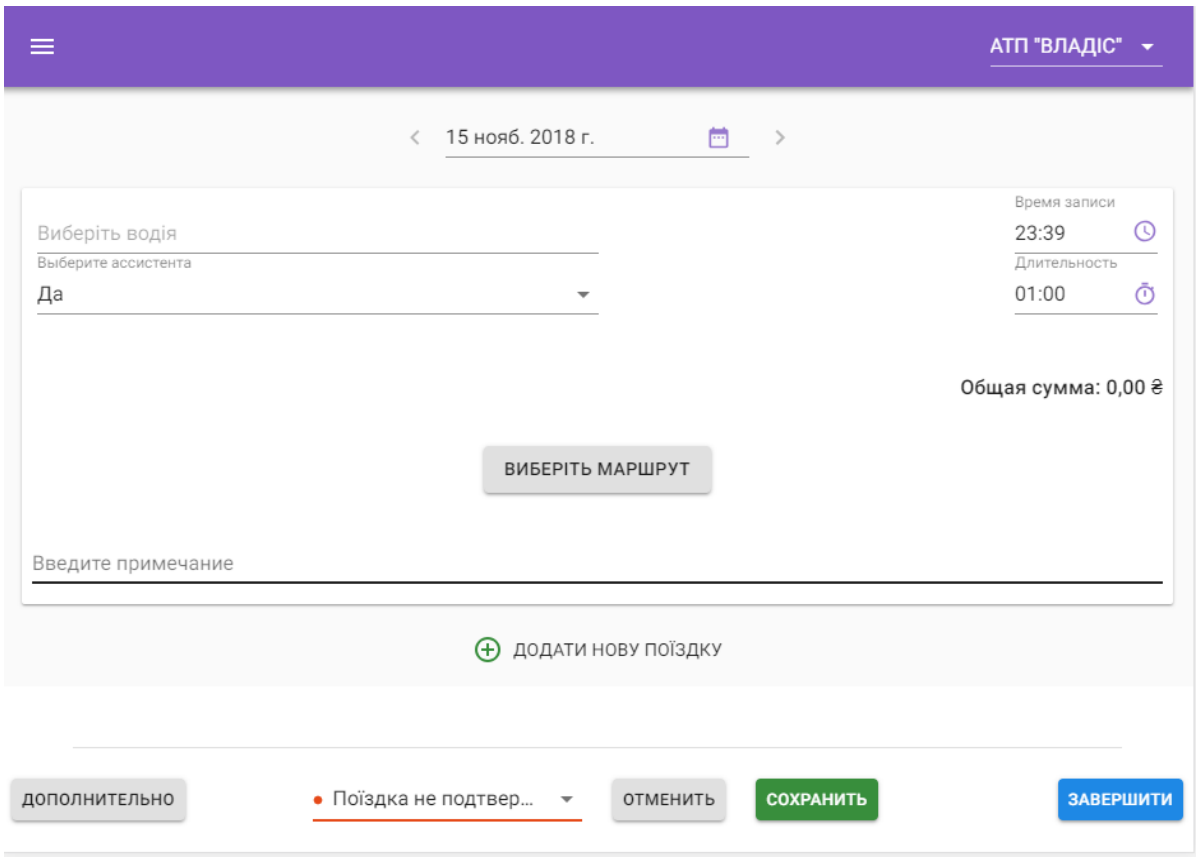


Рис. 3.7. Вікно запису

Користувач може вибрати маршрут, водія та час та створити необхідний запис про заплановану поїздку або відкрити існуючий запис та підтвердити його.

3.2.2. Опис вікна прогнозування

Вікно прогнозування зображено на рис. 3.8. Тут показується:

- обрана дата;
- кількість пасажирів;
- час поїздки;
- обраний маршрут.

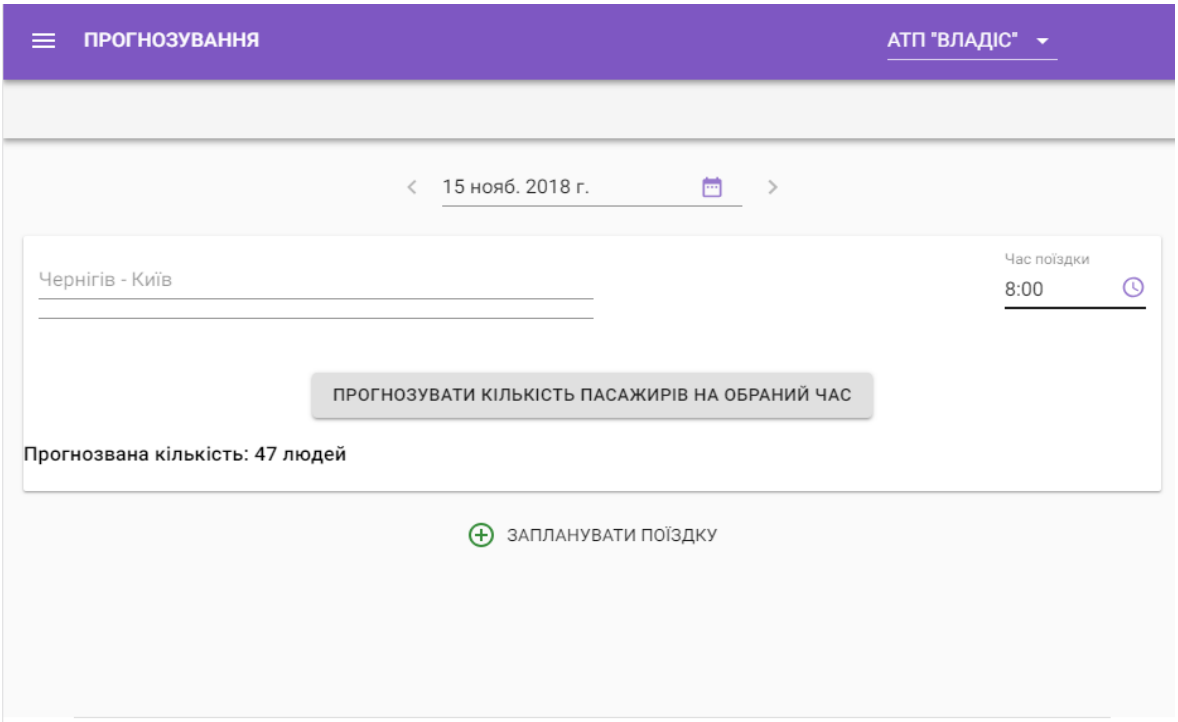


Рис. 3.8. Вигляд вікна прогнозування

3.3. Висновок до розділу 3

В даному розділі описаний процес розробки програмного забезпечення для оптимізації міжміських перевезень на основі отриманих прогнозних значень пасажиропотоку з використанням рекурентної нейронної мережі для прогнозування пасажиропотоку.

Описані вимоги користувача, функціональні вимоги та можливості розробленої системи.

Наведено архітектуру, на основі якої виконувалась розробка програмного забезпечення, інтерфейс користувача, а також описані основні класи розробленого програмного забезпечення. Проаналізровано сучасні мови програмування, середовища розробки та бібліотеки. Для реалізаії було обрано найбільш пристосовані для поставленої задачі засоби.

4. АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1. Дослідження прогнозних значень пасажиропотоку

Дослідження проводились на основі даних про перевезених пасажирів на маршрутах між містами Чернігів-Київ та між Черніговом та районними центрами Чернігівської області (Мена, Ріпки, Щорс, Городня). Результати прогнозування відображено на рис. 4.1.-4.5.

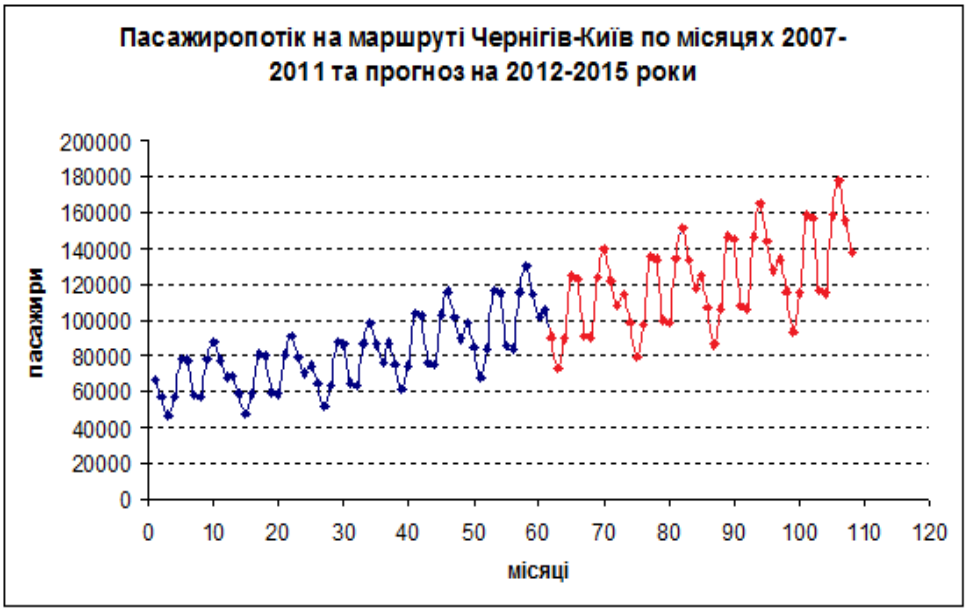


Рис. 4.1. Прогноз пасажиропотоку на маршруті «Чернігів-Київ»

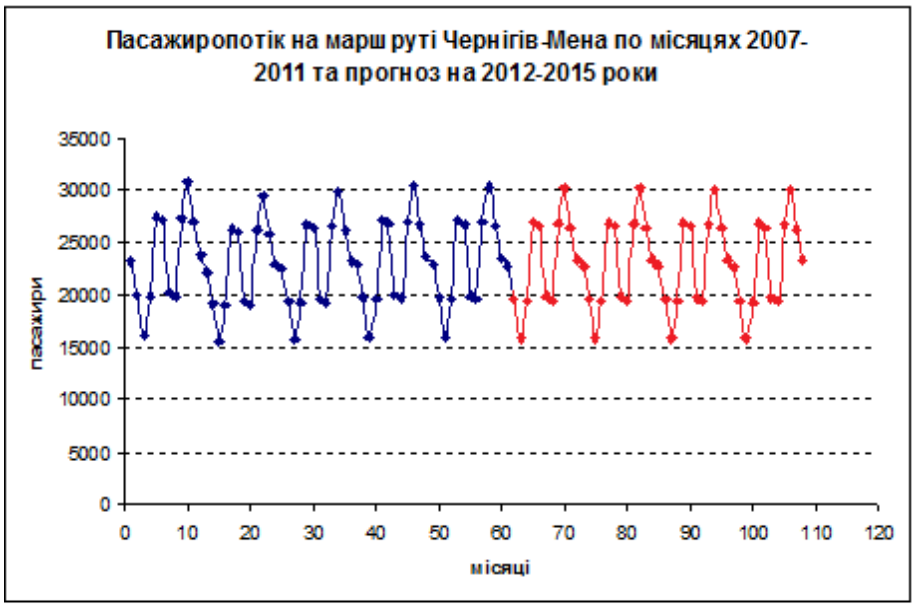


Рис. 4.2. Прогноз пасажиропотоку на маршруті «Чернігів-Мена»

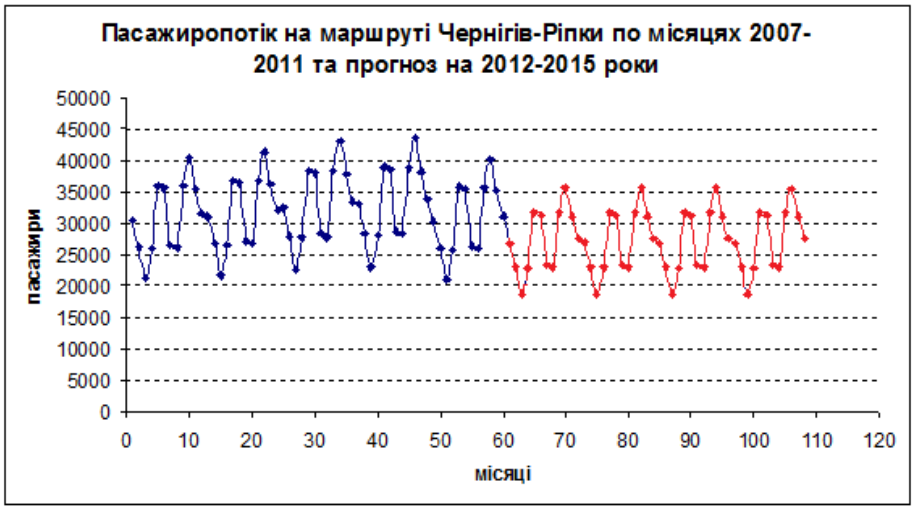


Рис. 4.3. Прогноз пасажиропотоку на маршруті «Чернігів-Ріпки»

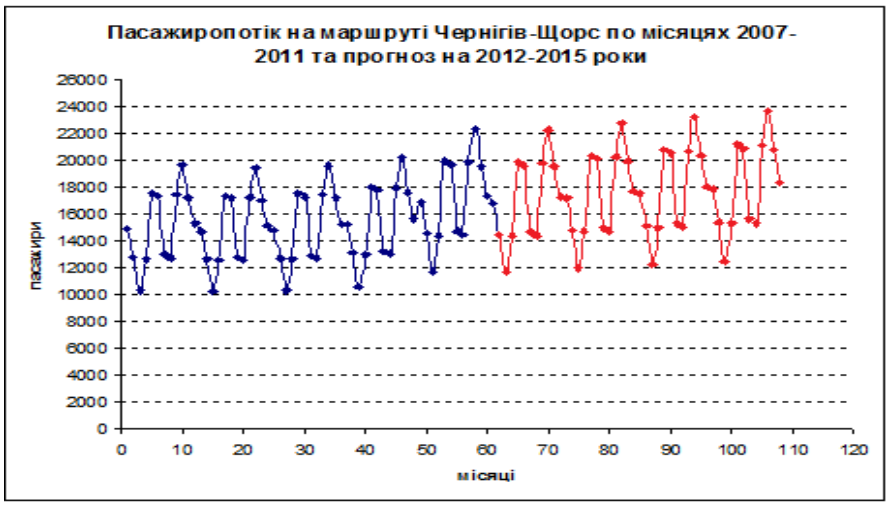


Рис. 4.4. Прогноз пасажиропотоку на маршруті «Чернігів-Щорс»

Рис. 4.5. Прогноз пасажиропотоку на маршруті «Чернігів-Городня»

Порівняємо методи прогнозування Бокса-Дженкінса, авторегресивної моделі порядку *p* та рекурентної мережі LSTM для прогнозування пасажиропотоку між містами Чернігів та Київ на 2012-2015 роки. Порівнявши прогнозні значення, отримані даними методами з фактичними даними, отримано такі показники точності прогнозу:

Таблиця 1. Показники точності прогнозу різними методами

Метод	Точність прогнозу
Бокса-Дженкісна	0,961
Авторегресивної меделі порядку *p*	0,933
Рекурентна нейронна мережа	0,930
Довга короткочасна пам’ять (LSTM)	0,983

Відповідно в стовпці «Точність прогнозу» 1 це максимум, 100%, 0 – абсолютна розбіжність в результаті.

При використанні LSTM згідно запропонованої моделі точність прогнозу кількості пасажирів у майбутньому збільшується на декілька відсотків, що дає можливість перевізникові зменшити витрати на паливо та покращити обслуговування для пасажирів завдяки кращому розкладу.

4.2. Висновок до розділу 4

У даному розділі наведено результати порівняння методів прогнозування Бокса-Дженкінса, авторегресивної моделі порядку p та рекурентної нейронної мережі LSTM для прогнозування пасажиропотоку між містами Чернігів та Київ на 2012-2015 роки.

При дослідженні результатів порівняння фактичних та прогнозованих поїздок після 10 та 40 епох навчання на наборі навчальних даних для рекурентної нейронної мережі LSTM точність прогнозу досягає 0,983, що є найбільш точним результатом серед розглянутих методів прогнозування.

5. РОЗРОБЛЕННЯ БІЗНЕС МОДЕЛІ

5.1. Опис проблеми та дерево проблем

5.1.1. Дерево проблем

Дерево проблем дисертаційного дослідження зображено на рисунку 5.1.

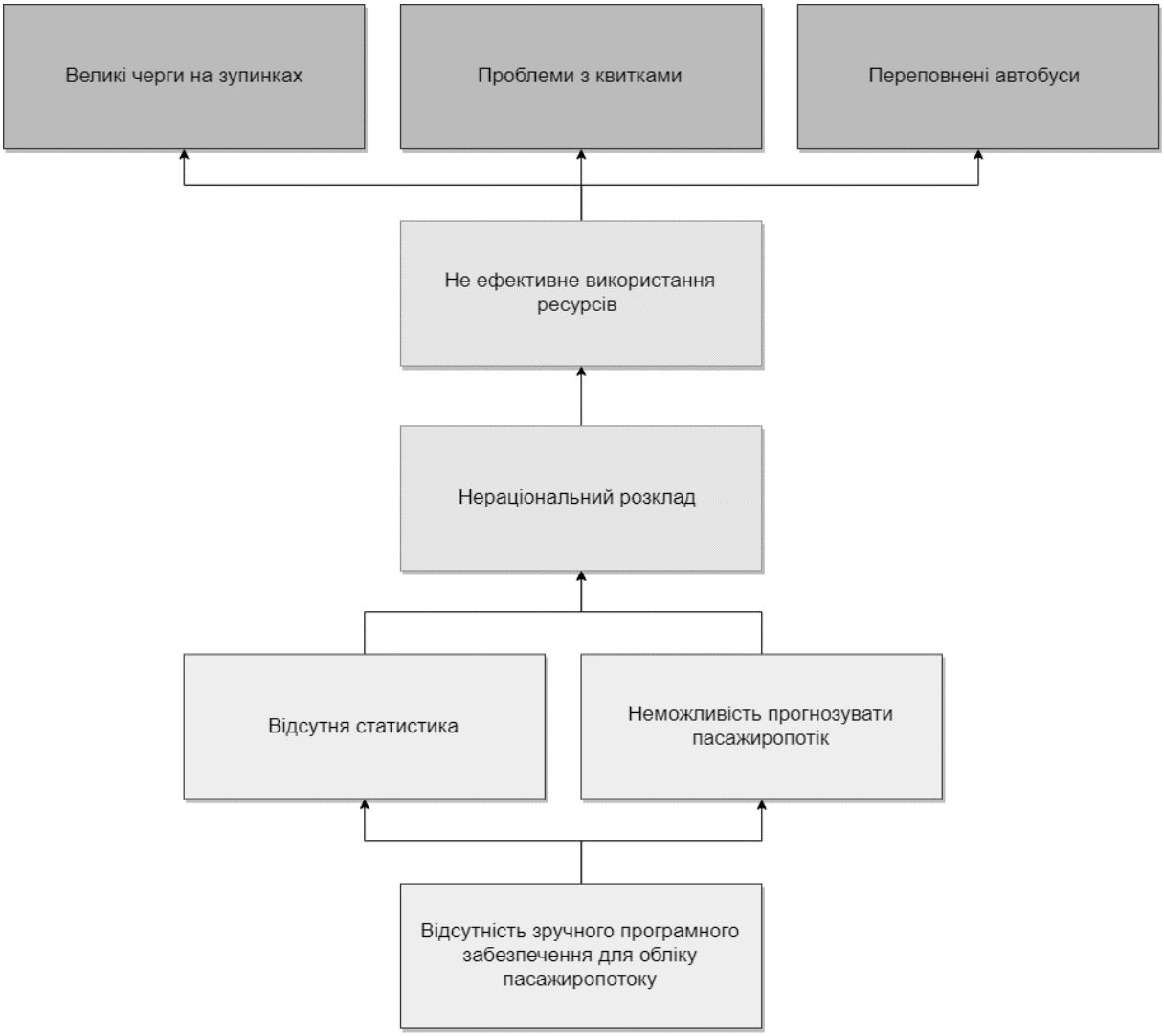


Рис. 5.1. Дерево проблем.

5.2. Аналіз зацікавлених сторін проекту

5.2.1 Зацікавлені сторони проекту

Зацікавлені сторони проекту та оцінка їх важливості та зацікавленості зібрані у таблиці 5.2.1.

Таблиця 5.2.1.

Група зацікавлених осіб	Інтереси групи в проекті	Умови довгого співробітництва з проектом	Важливість	Зацікавленість
Внутрішні зацікавлені сторони проекту				
Розробник проекту	Виконання проекту; Досягнення цільових показників проекту	Подальший розвиток проекту; Приток інвестицій	Висока (10)	Висока (10)
Команда керування проектом	Досягнення цільових показників проекту; Подальший розвиток технологій проекту	Подальший розвиток проекту; Приток інвестицій; Збільшення особистого доходу	Висока (9)	Висока (9)
Інвестори проекту	Отримання зазначених доходів від	Збільшення прибутку від інвестицій;	Висока (9)	Висока (9)

	участі в проекті; Подальший розвиток проекту; Досягнення цілей	Вигідніші умови підтримки проекту		
Внутрішньо – корпоративні зацікавлені сторони проекту				
Менеджмент проекту	Розвиток проекту; Збереження робочих місць;	Приріст робочих місць; Можливість кар’єрного росту; Реклама	Висока (9)	Середня (6)
Акціонери	Приріст доходності; Ріст загальної вартості проекту	Збільшення вартості проекту	Висока (9)	Висока (9)
Побічні співробітники	Кар’єрний ріст	Кар’єрний ріст	Середня (6)	Висока (9)
Зовнішні зацікавлені сторони проекту				
Автотранспортні підприємства, що займаються	Використання продукту проекту для	Розвиток технологіч-ного фону	Нижче середнього (4)	Висока (9)

міжміськими пасажирськими перевезеннями	покращення роботи автотранспорт-ного підприємства; Власний технологічний розвиток			
Наукові дослідницькі центри / заклади	Валідні дані для аналітики	Підтримка продукту	Нижче середнього (4)	Висока (9)
Побічні зацікавлені сторони проекту				
Розробники ПЗ	Використання потокових технологій	Оновлення технологій	Низька (3)	Низька (3)
Споживачі	Зменшення часу на очікування автобуса	Підтримка продукту	Низька (3)	Низька (3)

До зацікавлених сторін проекту відносяться такі групи [22]:

1. Розробники проекту — команда, яка бере на себе відповідальність вирішення проблеми, поліпшення методу для вирішення проблеми та його реалізації.
2. Команда керування проектом — команда, яка бере участь в розподілі зобов'язань, ресурсів та вибору напрямку розвитку проекту, також має зацікавленість у проекті.
3. Інвестори проекту — юридичні особи, які підтримують проект фінансовим забезпеченням та оцінюють і складають план фінансового розвитку проекту.
4. Команда аналітики проекту — команда, яка визначає цілі розробки впровадження їх в бізнес-процес підприємства та визначає технічні завдання для розробки.
5. Дата центри — організації, які використовують продукт для поліпшення роботи свого центру, також надають доступ до сервісу зберігання інформації.
6. Підприємства — організації, які використовують даний продукт для прискорення своєї працездатності та підтримують проект своєю активністю.
7. Наукові дослідницькі центри — центри, які використовують даний продукт для розширення тематики своїх досліджень та підтримують проект своєю активністю.
8. Споживачі — люди, які використовують даний продукт для збільшення швидкості та зручності використання маршрутних таксі та підтримують проект своєю активністю [23].
9. Розробники інших продуктів — люди, які використовують API даного проекту для прискорення роботи інших проектів.

5.3. Опис наукового продукту та технологій

Для реалізації системи прогнозування пасажиропотоку та організації автопарку був обраний необхідний набір засобів для розробки програмного продукту, а також стек технологій програмування**.**

Провівши аналіз існуючих технологій, було з'ясовано, що існує декілька наборів інструментальних засобів, які могли б забезпечити всі необхідні функції для реалізації задачі даної магістерської дисертації. Для реалізації поставленого завдання необхідно було визначити мову програмування, якою буде написана клієнтська та яка найкращим чином взаємодіяла б із сервером, була б максимально зручна для користування.

Проаналізувавши офіційні специфікації розробників та поради спеціалістів в області програмної інженерії щодо можливості взаємодії різних мов програмування між клієнтською і серверною частинами та врахувавши, що серверна частина написана на Java, SQL було вирішено використовувати мову програмування JavaScript з фреймворком React для розробки web-сайту [24], тому що web-проект не прив'язаний до певної платформи і може використовуватись як з настільного ПК, так і мобільного телефону. Набір саме цих засобів реалізації відрізняється хорошою взаємодією та стабільністю [25].

Далі розглянемо детальніше обраний інструментарій.

JavaScript — основна мова програмування для клієнтської частини.

React — відповідає за інтерфейс користувача, має зручну бібліотеку для написання інтерфейсу.

Parcel — відповідає за пакування проекту в завершеному стані, швидкий та зручний у використані.

Babel — відповідає за трансляцію коду для підтримки застарілих версій браузерів.

Redux — відповідає за управління станом додатку. Спрощує та прискорює швидкість написання коду.

React Router — відповідає за управління навігації по додатку.

INTL — відповідає за локалізацію додатку.

Styled Components — відповідає за реалізацію дизайну в проекті.

RXjs — відповідає за управління потоками в додатку.

Prettier — статичний аналізатор для мови програмування.

5.4. Бізнес-рішення. Основні характеристики бізнес-продукту

Розроблені на даний момент засоби для ведення статистики та прогнозування недостатньою мірою відповідають поставленим їм задачам. В Україні організація пасажирських автобусних перевезень в основному відбувається на основі складання розкладу руху автобусів [26]. Розклад є в пункті відбуття і призначення. Автобуси виїжджають чітко за розкладом, вони можуть виходити у рейс інколи напівпорожніми. Такі явища є збитковими для АТП. Через нестабільність потоку людей дуже складно реалізувати раціональний розклад. Прогноз та статистика в такому випадку абсолютна відсутня. Було здійснено пошук сторінок пошукової системи Google за запитами «Прогнозування» та «Моделювання».

Існують компанії, які пропонують налаштувати спеціальну систему під ваш автопарк. Через персональний комп’ютер поїздки будуть записуватися до бази, де можливо буде рахувати статистику [27].

Можна сказати, що для великих компаній це найзручніший варіант, проте є велика й кількість недоліків.

Переваги:

- простота при замовленні;
- повна організація автопарку.

Недоліки:

- надзвичайно висока ціна за систему;
- неможливість законного використання через санкції;
- неможливість прогнозувати без участі програміста;
- неможливо працювати через планшет, чи мобільний телефон.

Отже, використання такої системи не підходить для приватних автотранспортних підприємств, що займаються міжміськими перевезеннями, так як, більшість автопарків не зможе зручно використовувати персональні комп'ютери через відсутність вокзалів як таких.

Більшість приватних АТП в Україні досі не використовують жодної системи для зберігання статистичних даних пасажиропотоку. Всі дані записуються вручну до облікових журналів і відповідно аналіз і обробка цих даних неможлива без витрат колосального часу і як водії так і диспетчери лиш інтуїтивно мають уявлять кількість пасажирів на поїздку [28].

Аналіз систем прогнозування і організації пасажиропотоку показав, що:

- майже всі міжміські компанії автоперевезення не мають обліку та систем прогнозування пасажиропотоку;
- деякі підприємства записують вручну весь потік пасажирів;
- 1С, Excel, Matlab та MathCad не є зручними для таких цілей;
- існуючі системи не є мобільними і робота з ними може виконуватися тільки на стаціонарних персональних комп'ютерах;
- не раціонально підбирається тип автобусів для зменшення затрат;
- можливе моделювання буде економити час диспетчера.

Враховуючи результати аналізу, можна зробити висновок, що на даний момент існує необхідність у створенні сервісу, який би був зручним у використанні і точним у виконанні всіх необхідних функцій [30].

5.5. Конкурентні переваги рішення

Актуальність та значимість прогнозування пасажиропотоку в практиці будь-якого автотранспортного підприємства є очевидною. Комплексна система взаємозв'язаних планів на перспективу, уточнених в тактичних і

оперативних діях та деталізованих по ресурсах, строках і виконавцях – найважливіший інструмент поетапної реалізації цілей підприємства [31].

5.6. Клієнти. Сегменти ринку споживання

Критерієм вибору є спосіб оцінки перспективності входження конкретної фірми в певну частину ринку.

Популярними є наступні критерії:

1. розміри або ємність;
2. поріг входження;
3. можливості ринку;
4. потенційні доходи;
5. конкурентне середовище;
6. ресурси компанії.

Під розміром ринку мається на увазі максимальний торговий оборот в сегменті за певний проміжок часу. Тут потрібно оцінити, чи вигідно виробляти цей обсяг продукції, чи не буде нести фірма втрати, пов'язані з перевиробництвом, або з недоліком товару. Оскільки виробничий потенціал продукції залежить від користувачів даної продукції втрати за виробництво будуть дуже низькими, та загалом будуть нести подальший вигідний розвиток в проекті. Втрати оцінюються затратами на орендування VPS серверів та їх місце розташування.

Для підприємств, особливо невеликих, важливий критерій – простота входження в бажаний сегмент, які потрібно затратити кошти, а також зусилля на подолання різних бюрократичних бар’єрів. В основному це питання, яке буде стосуватися політичного становища в тій чи іншій країні [32].

Ринок може перебувати на різних стадіях розвитку. Залежно від того, коли підприємство входить в його сегмент, є різні можливості для здійснення діяльності. Якщо попит на послугу або товар вже падає, то,

відповідно, можливостей для функціонування набагато менше, ніж, коли вони знаходяться на піку або ж в періоді бурхливого зростання.

Потрібно враховувати прибутковість або рентабельність від функціонування в цьому сегменті.

Наявність великої кількості конкурентів також негативно може позначитися на діяльності підприємства, тому потрібно уважно оцінити конкурентне середовище. Потрібно врахувати, чи будуть фірми в цьому сегменті дійсно конкурентами. В даному сегменті є дуже велика конкуренція яка стосується пошуку інформації, але конкуренти не використовують найефективніший метод пошуку релевантної інформації.

Наявність знань навичок у робітників і керуючого персоналу, а також обладнання враховується при оцінці можливості входу в певну частину ринку. Ринок IT сфери являється одним із самих великих ринків у світі, тому знайти людей, які будуть керувати персоналом та професіональних робітників у діяльності пошуку інформації, не стане перешкодою [33].

Існує три основних стратегії вибору цільових сегментів, які розглядаються в більшості праць різних авторів:

1. Концентрований маркетинг.

2. Диференційований маркетинг.

3. Недиференційований маркетинг.

Суть концентрованого маркетингу полягає в зосередженні компанії на певному сегменті ринку або декількох сегментів для отримання максимального прибутку. Ця стратегія дозволяє максимально ефективно використовувати всі засоби підприємства. Вибір цього напрямку розвитку залежить від кількості напрямків і від їх величини [35].

Диференційований маркетинг передбачає охоплення фірмою всіх галузей торгівлі або надання послуг, але розробкою окремого плану для кожного напрямку. До плюсів цього напрямку можна віднести індивідуальний підхід до більшості клієнтів. До мінусів можна віднести великі витрати на розробку напрямків, рекламу, а також на виробництво.

Негнучким стосовно пристосованості до певних областях ринку є недиференційований маркетинг. Він не враховує потреб різних груп споживачів. Вироблена продукція або послуги, що надаються, мають стандартизований вигляд. Одноманітність скорочує витрати, пов'язані з виробництвом або просуванням товару або послуги. Одночасно з цим не враховуються інтереси пересічного споживача. Розглянувши всі три основні методи стратегій цільових сегментів, було вирішено зупинитися на диференційованому маркетингу [34].

Сегментування ринку підприємства здійснюється в кожному конкретному випадку по-різному. На це впливають критерії і стратегії, описані вище. Ідеальним вважається пристосування під кожного конкретного споживача. Тому найчастіше людей зі схожими потребами відносять до певних груп за показниками:

1. географічним;
2. демографічним;
3. психографічним;
4. поведінковим.

Вибір цільового сегмента завжди відбувається за критеріями, описаним вище, при цьому враховуються можливості фірми. Далі вибирається стратегія і відбувається сегментація ринку.

Конкуренція на ринку виробників товарів і послуг повсякденного попиту досягла таких масштабів, що бути гнучким до змін, це один з основних показників успіху. Визначити, яким буде ваш бізнес завтра, реально вже зараз. Компанії оперують великими масивами даних, які при правильній обробці можуть бути джерелом для пошуку закономірностей і можливих сценаріїв розвитку тих чи інших процесів.

Рисунок 5.2. Аналітика і прогнозування

Мій проект фокусується на прогнозуванні пасажиропотоку та організації рухомого складу автотранспортних підприємств, які займаються міжміськими пасажирськими перевезеннями. Використовуючи мій продукт, АТП зможуть:

- складати якісний прогноз пасажиропотоку для міжміських маршрутів, що дозволить мінімізувати витрати, пов'язані з витратами на заробітну платню та нераціональним використанням палива;
- планувати процес відправки автобусів, забезпечуючи тим самим потрібний обсяг посадочних місць на зупинках і скорочуючи out-of stock;
- розробляти сценарії реалізації промо-активностей в мережах;
- формувати короткострокові прогнози для прийняття оперативних рішень;
- приймати правильні і своєчасні рішення, використовуючи діючі сценарії розвитку тих чи інших бізнес процесів [36].

5.7. Унікальна ціннісна пропозиція

Пропозиція споживчої цінності включає в себе опис проблеми, що є у покупця, рішення, що знімає цю проблему, і цінність цього рішення з точки зору покупця.

Сутність конкуренції полягає в споживчій цінності. Щоб вижити, компанія повинна донести до покупця конкретну пропозицію цінності, націлену на конкретну нішу ринку.

Значення цінності з точки зору покупця: можна запитати у покупця ціну, відповідну цінності, яку він отримує, незалежно від того, скільки витрачено коштів на її створення. Якщо ціна, запитувана за продукт, менша вигоди, одержуваної покупцем з його допомогою, то в очах покупця комерційна пропозиція буде мати хорошу цінність.

Опис унікальних вигод повинен створювати потребу в запропонованому товарі. Цей опис має диференціювати послуги від інших подібних пропозицій. Тобто потенційний покупець, прочитавши пропозицію, повинен відразу зрозуміти, чим ваша пропозиція в вигідну сторону відрізняється від пропозицій конкурентів. Маркетинговий слоган повинен створювати емоційний імпульс, підштовхуючи людей до покупки транспортної послуги.

5.8. Доходи і витрати

Дохід розраховується як загальна сума грошей, отримана в результаті реалізації товарів або послуг, а прибуток — як дохід з відрахуванням витрат на виробництво, придбання і збут товарів або послуг [38].

Витрати підприємства — фінансова категорія, що характеризує в грошовій та матеріальній формах оцінку господарської діяльності (підготовка, організація й здійснення процесів виробництва та реалізації продукції, товарів), фінансової й соціальної діяльності.

Витрати		
Послуга	Ціна($)	Ціна за рік($)
Оренда VPS серверів	63.88	702.68
Оренда сервера бази даних	57.6	691.2
Основні витрати	1549.05	18588.6
Всього	1670.53	19982.48

Таблиця 5.2. Витрати

Для оцінки витрат було обрано валюту usd, також проаналізовано багато ресурсів та введено в одну таблицю див. табл. 5.2.

Оренда VPS серверів або віртуального виділеного серверу: критеріями даної послуги є місце розташування, якість підтримки та вартість серверу. В табл.5.3 наведений список аналізів VPS серверів.

Оренда сервера бази даних: після проведення аналізу було прийнято рішення зупинитися на сервері MongoBD. Критеріями даної послуги є місце розташування, якість підтримки, швидкість отримування даних та вартість серверу.

Основні витрати: до основних витрат належать реклама, витрати на електроенергію, витрати на зарплату.

НАЗВА	ОФІС (РОЗТАШУВАННЯ)	СЕРВЕРИ (РОЗТАШУВАННЯ)	ВІРТУАЛІЗАЦІЯ
Fozzy	США	Нідерланди, США, Москва	OpenVZ
FirstVDS	Москва	Санкт-Петербург, Москва	OpenVZ
TimeWeb	Санкт-Петербург	Санкт-Петербург	OpenVZ, KVM,

Amazon	США	Велика Британія, Ірландія, Франція, Канада, Німеччина, Італія, Іспанія, Нідерланди, Австралія, Бразилія, Японія, Китай, Індія та Мексика	OpenVZ, KVM, Hyper-V
NetAngels	Єкатеринбург	Єкатеринбург, Нідерланди, Німеччина	KVM

Таблиця 5.3. Список підприємств по наданню послуг оренди серверів

Доходи підприємства — це збільшення економічних вигід у вигляді надходження активів або зменшення зобов'язань, внаслідок чого збільшується власний капітал підприємства. Розрахунок відбувається за середньою кількістю підписок кожний місяць 100 користувачів. Було обрано декілька програм доходів, а саме:

1. Програма на підписку — користувач робить підписку на деякий період часу в даному випадку на рік це 150, місяць це 15. Ця підписка має значення, якщо користувачу потрібно більше ніж один пошуковий сервіс. Підрахунки наведено в табл.5.4.
2. Програма оренди рекламного місця — користувач робить на своїй головній сторінці створеного пошукового сервісу, оренду місця для реклами, яку він хоче розмістити. З нього здіймається 10 відсотків від зароблених коштів за рекламу [39]. Враховуючі середню підписку за місяць та дохід від рекламного місця маємо, за рік 480, за місяць 40. Підрахунки наведено в табл.5.4.

3. Програма розміщення власної реклами — користувач хоче мати більш ніж один пошуковий сервіс, але в нього немає можливості взяти програму на підписку. Підрахунки наведено в табл.5.4.

Доходи		
Послуга	Ціна	Ціна за рік
Програма на підписку	15	150
Програма оренди рекламного місця	40	480
Програма розміщення власної реклами	800	9600
Дохід за місяць з впливом середньої кількості підписок	855	
Дохід за рік з впливом середньої кількості підписок	41472	

Таблиця 5.3. Доходи

5.9. Бізнес модель

Перед українською економікою завдання довгострокового розвитку вимагають радикального підвищення ефективності управління на різних рівнях. Це завдання стоїть також перед вітчизняними автотранспортними підприємствами. Необхідність її вирішення актуалізує розробку інструментарію для прогнозування перспектив розвитку та оцінки впливу розробляються стратегій на стійкість фінансового стану підприємств.

Застосування ймовірнісних моделей для прогнозування розвитку підприємства з урахуванням ризиків пов'язане з постановкою цілого ряду складних проблем як загальнотеоретичного, так і методичного характеру, які практично не висвітлені у вітчизняній і зарубіжній спеціальній літературі. Без їх вирішення неможливо широке впровадження в

українських підприємствах сучасних методів фінансового стратегічного управління.

Виділяють необхідні розроблені типові засоби з фінансового моделювання розвитку підприємства, а саме:

- типова багатотрендова фінансова модель, що дозволяє прогнозувати динаміку грошових потоків і оцінювати їх коливання, в тому числі ймовірності їх відхилень від мінімально допустимих значень;
- алгоритми обробки вихідних часових рядів, що забезпечують використання поряд з типовими даними емпіричних розподілів ймовірностей без необхідності їх аналітичного опису, що істотно спрощує впровадження імітаційного методу моделювання в компаніях;
- підхід до структуризації фінансової моделі, заснований на її послідовній деталізації «зверху вниз», при цьому можливий різний ступінь деталізації в залежності від цілей аналізу і наявності вихідної інформації;
- типові засоби автоматизації аналізу бухгалтерських даних; статистичного аналізу часових рядів; побудови графіків і гістограм, в тому числі для інтервалів різної довжини (тиждень, місяць, квартал). У сукупності це дозволяє зробити доступною для менеджерів компаній підготовку вихідних даних для моделі прогнозування;
- особливості оцінки ризиків та інструменти управління розвитком компанії на трьох рівнях (інвестиційного проекту, портфеля проектів, компанії в цілому) з урахуванням безперервності збору, обробки та аналізу даних, що надходять як ззовні, так і формуються всередині компанії;
- розглянутий механізм аналізу чутливості з урахуванням нелінійності; а також підхід до оцінки сумарного ризику

портфеля проектів, заснований, зокрема, на результатах імітаційного моделювання окремих інвестиційних проектів.

Застосування даних підходів до побудови моделей прогнозування та оптимізації діяльності АТП необхідні для оцінки перспектив розвитку і ризику настання неплатоспроможності. Дані підходи можуть використовуватися не лише керівництвом автотранспортного підприємства, але і зовнішніми структурами, в тому числі організаціями (наприклад, в рамках холдингів, держкорпорацій), банками, інвестиційними і страховими компаніями.

ВИСНОВКИ

У даній магістерській дисертації виділено недоліки сучасних інформаційних технологій, які використовуються для організації роботи підприємств пасажирського транспорту у міжміському сполученні. Виділено невирішені задачі, пов’язані з обробкою даних про пасажирів та організацією структури рухомого складу. Об’єктом дослідження було обрано мережу міжміських пасажирських перевезень.

Аналіз предметної області показав, що задачам оптимального управління АТП з використанням ІТ приділено мало уваги. Дуже мало інформації про підвищення якості обслуговування пасажирів при умові мінімізації витрат АТП, планування та обслуговування раціонального розкладу для рухомого складу. Сформульовано проблеми, пов'язані з особливостями функціонування АТП та визначено необхідність розробки інформаційних технологій для оптимального управління автотранспортним підприємством.

Серед багатьох методів для прогнозування часових рядів як даних про пасажиропотоки у міжміському сполученні у якості основного методу був обраний метод машинного навчання – довгострокова короткочасна пам'ять LSTM. Властивістю даної нейронної мережі є довгострокове запам’ятовування даних, що поліпшує точність алгоритму з плином часу. Навчання та тестування обраної нейронної мережі проводилось на реальних статистичних даних про міжміські пасажирські перевезення. При порівнянні обраного методу прогнозування з іншими методами було виявлено, що при використанні LSTM отримуємо більш точний результат за менший час.

Для реалізації даного методу використано сучасні мови програмування, які дозволяють виконувати адаптовані обчислення на великих наборах статистичних даних. Таким чином, розроблене програмне забезпечення може успішно використовуватися в діяльності АТП України.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Миколайчик В.В. «Програмне забезпечення для прогнозування пасажиропотоку та організації рухомого складу АТП» в міжнародному науковому журналі "Інтернаука". — 2017. — №11.
2. Gomathi V. Human Facial Expression Recognition Using MANFIS Model / V. Gomathi, K. Ramar, A. S. Jeevakumar // Proceedings of World Academy of Science Engineering and Technology. — 2009. — 38. — P. 338–342.
3. I. Cohen. Emotion Recognition from Facial Expressions Using Multilevel HMM / I. Cohen, A. Garg, T. S. Huang // Neural Information Processing Systems. — 2000.
4. Jang J.-S. R. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence / J.-S. R. Jang, C.-T. Sun, E. Mizutani. — Prentice Hall, 1997. — 614 p.
5. N. Tsapatsoulis. Mining Associations in text in presence of background knowledge / N. Tsapatsoulis, K. Karpouzis, G. Stamou, F. Piat, S. Kollias // EURASIP Journal on Applied Signal Processing. — 2002. — P. 1021–1038.
6. Douglas David, Peucker Thomas. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature // The Canadian Cartographer. 1973. № 10(2). P. 112-122
7. Ramer Urs. An iterative procedure for the polygonal approximation of plane curves // Computer Graphics and Image Processing. 1972. № 1(3). P.244-256
8. Хайкин С. Нейронные сети: полный курс : пер. с англ. / С. Хайкин. — [2-е вид.]. — М. : Издательский дом «Вильямс». — 2006. — 1104 с.
9. Кононюк А. Ю. *Нейронні мережі*і *генетичні алгоритми* / А. Ю. Кононюк. - К.: *Корнійчук, 2008. – 446* с.
10. Галушкин А. И. Синтез многослойных систем. — М.: Энергия, 1974. – 368 с.
11. Нефьодов Ю.М., Балицька Т.Ю. /Методи оптимізації в прикладах і задачах: Навчальний посібник. – К.: Кондор, 2011. – 324 с.

12. H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber. A System for Robotic Heart Surgery that Learns to Tie Knots Using Recurrent Neural Networks. Advanced Robotics, 22/13–14, 2008 - pp. 1521–1537
13. Хайкин С. Нейронные сети: Полный курс. Пер. с англ. Н. Н. Куссуль, А. Ю. Шелестова. 2-е изд., испр. — М.: Издательский дом Вильямс, 2008 - 1103 с.
14. Guernic P. Python-A data flow-oriented language for neural networrk[Text] / P. Le Guernic, A. Benveniste, P. Bournai, T. Gautier — 1986. — P. 4-5.
15. Brookes S.D. A theory of communicating sequential processes [Text] / S.D. Brookes, C.A. Hoare, A.W. Roscoe — 1984. — P. 560-599.
16. Lee E.A. DRAW: A Recurrent Neural Network [Text] / E.A. Lee and D.C. Messerschmitt — 1987. — P. 24-35.
17. LSTM. [Електронний ресурс]. — Режим доступу: https://wikipedia.org/wiki/LSTM_— Дата доступу : Листопад 2018. — Назва з екрана.
18. LSTM, Form 8-K, Current Report, Filing Date Jan. [Електронний ресурс]. — Режим доступу: *http://edgar.secdatabase.com/2296/74398818000005/filing-main.htm*. — Дата доступу : Листопад 2018. — Назва з екрана.
19. LSTM Inc, Form 10-K, Annual Report. [Електронний ресурс]. — Режим доступу: *http://secdatabase.com/1004/0000743988-17-000046.htm* — Дата доступу : Листопад 2018. — Назва з екрана.
20. Brian Bailey, EE Times. Second generation for FPGA software. [Електронний ресурс]. — Режим доступу: *https://www.eetimes.com/document.asp?doc_id=1315621*. — Дата доступу : Листопад 2018. — Назва з екрана.
21. CERN Scientists Use Virtex-4 FPGAs for Big Bang Research. [Електронний ресурс]. — Режим доступу: http://www.xilinx.com/publications/xcellonline/xcell_65/xc_pdf/p28_31_65_F_XiWild.pdf. — Дата доступу : Листопад 2018. — Назва з екрана.

22. Gated Feedback Recurrent Neural Networks. [Електронний ресурс]. — Режим доступу: http://edgar.secdatabase.com/846/119312512182086/filing-main.htm. — Дата доступу : Листопад 2018. — Назва з екрана.
23. FPGAs Cool Off the Datacenter, Xilinx Heats Up the Race. [Електронний ресурс]. — Режим доступу: https://www.eejournal.*com/article/20141118-datacenter*. — Дата доступу : Листопад 2018. — Назва з екрана.
24. LSTM vs RNN, Calling the Action in the Greatest Semiconductor Rivalry. [Електронний ресурс]. — Режим доступу: https://www.eejourn*al.com/article/20140225-rivalry*. — Дата доступу : Листопад 2018. — Назва з екрана.
25. RNN [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/RNN. — Дата доступу : Листопад 2018. — Назва з екрана.
26. The Vivado Design Suite accelerates programmable systems integration and implementation by up to 4X [Електронний ресурс]. — Режим доступу: https://www.edn.com/electronics-products/other/4375467/The-Vivado-Design-Suite-accelerates-programmable-systems-integration-by-up-to-4X. — Дата доступу : Листопад 2018. — Назва з екрана.
27. WebPACK edition of Xilinx Vivado Design Suite now available [Електронний ресурс]. — Режим доступу: https://www.xilinx.com/products/design-tools/vivado/vivado-webpack.html. — Дата доступу : Листопад 2018. — Назва з екрана.
28. Altera Shipping 28-nm FPGAs [Електронний ресурс]. — Режим доступу: https://www.nasdaq.com/g00/article/altera-shipping-28nm-fpgas-analyst-blog. — Дата доступу : Листопад 2018. — Назва з екрана.
29. Altera's Quartus II design software features Qsys System Integration Tool [Електронний ресурс]. — Режим доступу: https://www.eetimes.com/document.asp?doc_id=1316604. — Дата доступу : Листопад 2018. — Назва з екрана.

30. Latest and greatest Quartus II design software from Altera [Електронний ресурс]. — Режим доступу: https://www.eetimes.com/document.asp?doc_id=1316845. — Дата доступу : Листопад 2018. — Назва з екрана.
31. Neural networks [Електронний ресурс]. — 2010. — Режим доступу : http://xoops.ws/modules/instruction/page.php?id=493. — Дата доступу : Листопад 2018. Назва з екрана.
32. Фленаган, Д. JavaScript. Подробное руководство, 6-е издание [Текст] / Д. Фленаган : пер. с англ. — Санкт-Перетбург : Символ-Плюс, 2012. — С. 21.
33. PixelCrayons [Електронний ресурс]. — 2004. — Режим доступу : http://bootstrap.pixelcrayons.com/. — Дата доступу : Листопад 2018. Назва з екрана.
34. Zurbfoundation [Електронний ресурс]. — 2016. — Режим доступу : http://foundation.zurb.com/. — Дата доступу : Листопад 2018. Назва з екрана.
35. JavaTpoint [Електронний ресурс]. — 2005. — Режим доступу : https://www.javatpoint.com/jsp-tutorial. — Дата доступу : Листопад 2018. Назва з екрана.
36. Діаграма прецедентів [Електронний ресурс]. — Режим доступу : https://uk.wikipedia.org/wiki/Діаграма_прецедентів. — Дата доступу : Листопад 2018. — Назва з екрана.
37. Діаграма послідовності [Електронний ресурс]. — Режим доступу : https://uk.wikipedia.org/wiki/Діаграма_послідовності. — Дата доступу : Листопад 2018. — Назва з екрана.
38. Data transfer object [Електронний ресурс]. — Режим доступу : https://en.wikipedia.org/wiki/Data_transfer_object. — Дата доступу : Листопад 2018. — Назва з екрана.

39. ITKeys.ru [Електронний ресурс]. — 2010. — Режим доступу : http://itkeys.ru/responsive-and-adaptive-design/. — Дата доступу : Листопад 2018. Назва з екрана.
40. Бибо, Б. jQuery. Подробное руководство по продвинутому JavaScript, второе издание [Текст] / Б. Бибо, И. Кац : пер. с англ. — Санкт-Перетбург : Символ-Плюс, 2011. — 624 с.

ДОДАТКИ

Додаток 1

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНИЙ МЕТОД ОПТИМІЗАЦІЇ МІЖМІСЬКИХ ПАСАЖИРСЬКИХ ПЕРЕВЕЗЕНЬ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

Виконав студент:
Миколайчик В.В.

Науковий керівник:
к.т.н. Олещенко Л.М.

Київ – 2018

Наукове завдання

- Удосконалити метод прогнозування пасажиропотоку з використанням нейронної мережі.

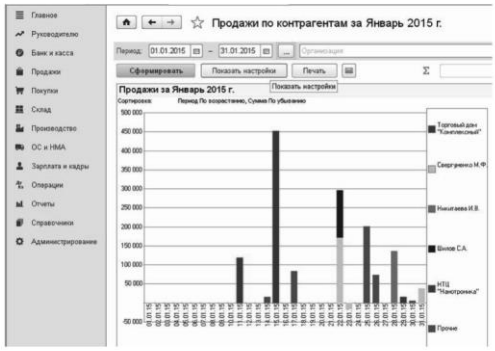
Мета дослідження

- Оптимізувати розподіл рухомого складу залежно від отриманих значень пасажиропотоку.
- Створити програмну систему для управління пасажирськими перевезеннями на міжміському маршруті.

Окремі завдання

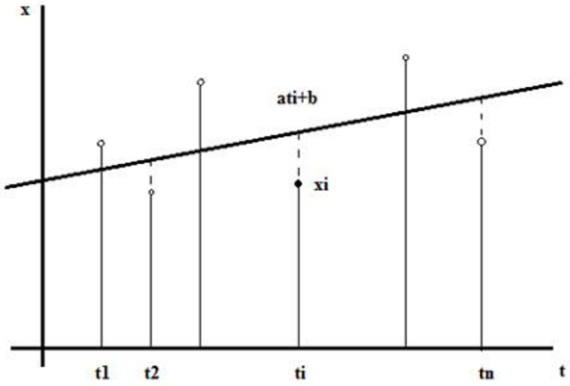
1. Провести аналіз існуючих рішень.
2. Розробити архітектуру нейронної мережі.
3. Розробити програмний метод оптимізації міжміських перевезень на основі нейронної мережі LSTM.
4. Провести аналіз результатів дослідження в порівнянні з існуючими методами.
5. Створити бізнес проект для програмного забезпечення.

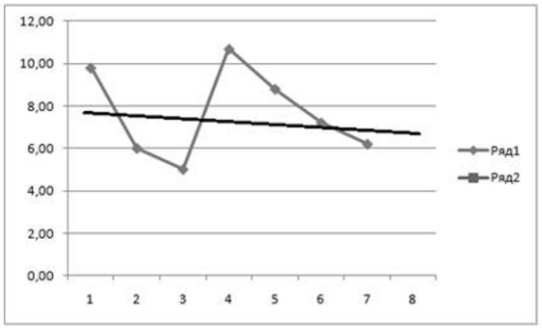
Існуючі рішення



Математичні методи прогнозування

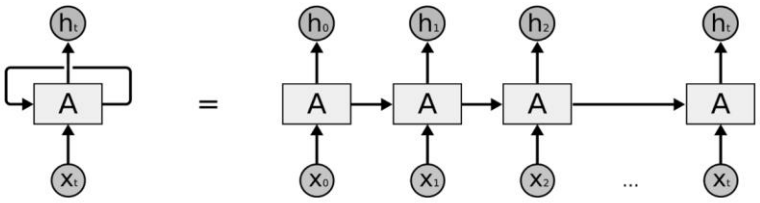
Експоненційного згладжування Холта-Уінтерса



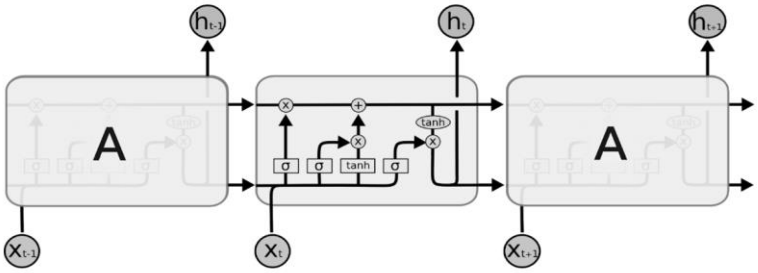


Ковзного середнього

Рекурентна нейронна мережа



LSTM – Long-short time memory



Технології

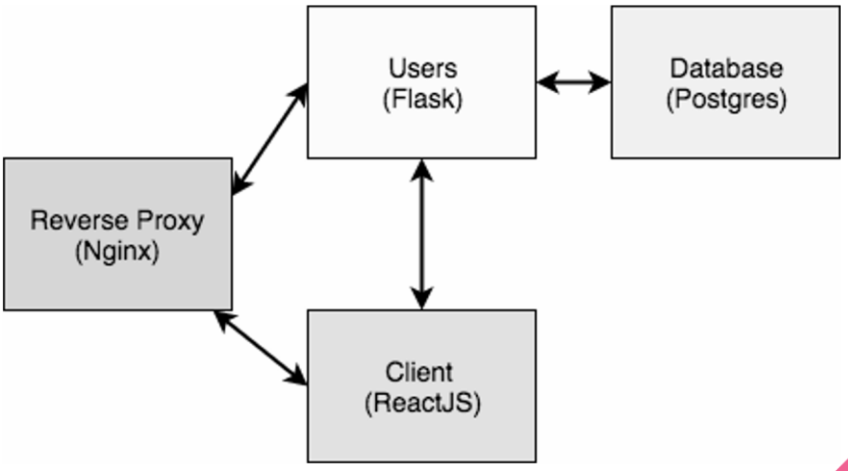
Flask

NGINX

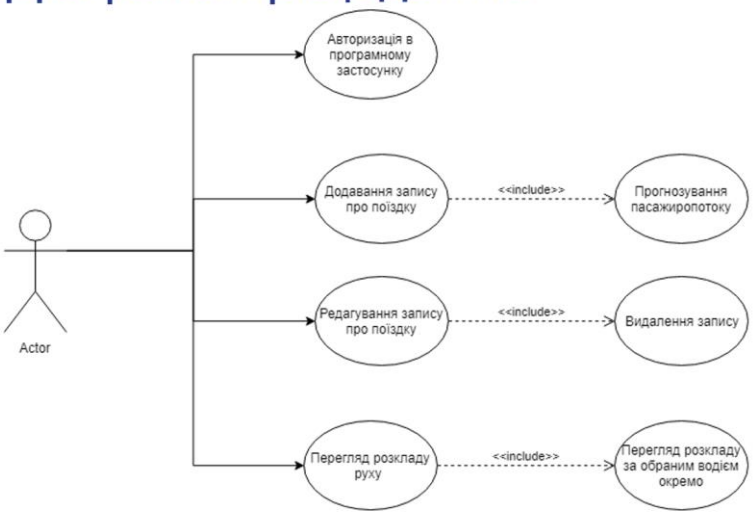
AWS

PostgreSQL

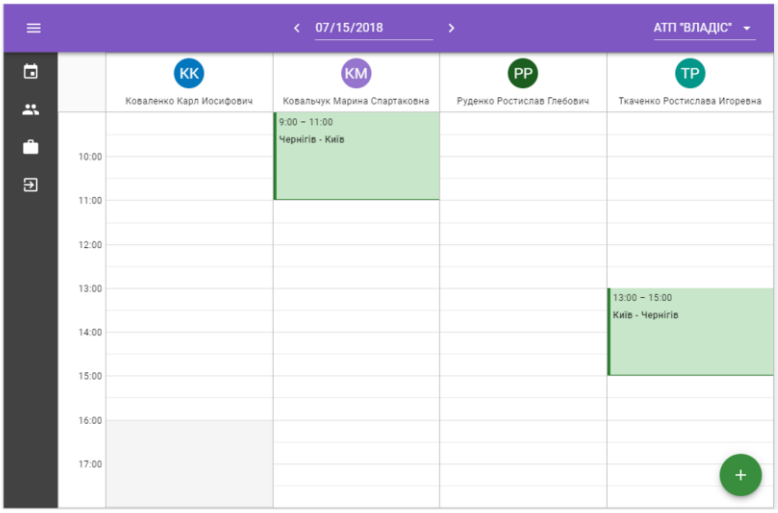
Особливості архітектури програмного комплексу



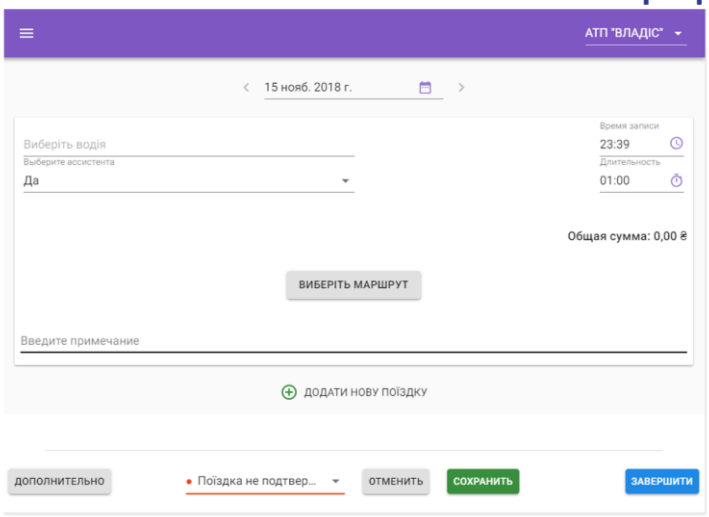
Діаграма прецедентів



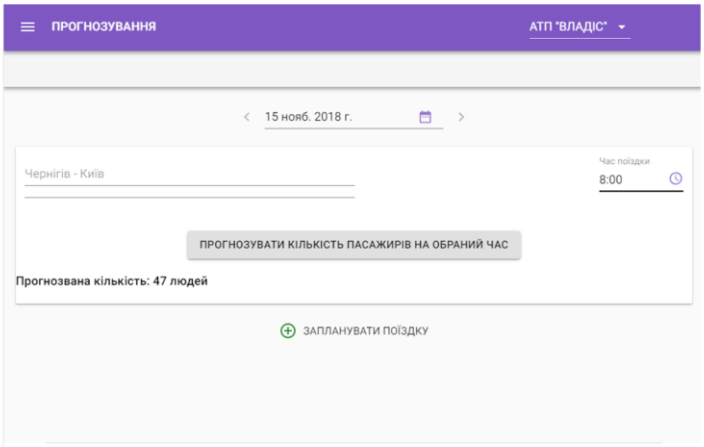
Головне вікно додатку – розклад



Приклад використання програмного забезпечення. Заповнення форми



Приклад використання програмного забезпечення. Результат прогнозування



Оптимізація витрат АТП

- Уточнення прогнозного значення пасажиропотоку.
- На основі уточненого значення пасажиропотоку більш раціональне використання рухомого складу.

Результати досліджень

Метод / критерій	Точність прогнозування	Швидкість навчання	Можливість врахування додаткових факторів	Складність
Математичні методи	92.1%	+	-	+
SRNN	90.5%	+	-	-
ESN	91.8%	-	+	-
LSTM	97.7%	+	+	+

БІЗНЕС-МОДЕЛЬ

<table>
<tr><th>Ключові партнери</th><th>Ключова діяльність</th><th>Цінністна пропозиція</th><th>Відношення з клієнтами</th><th>Сегменти користувачів</th></tr>
<tr><td rowspan="2">1. Логістичні компанії.
3. Інвестори.</td><td>1. Розробка програмного забезпечення для оптимізації міжміських перевезень.</td><td rowspan="2">1. Оптимізація міжміських пасажирських перевезень.
2. Зручний інтерфейс.
3. Легкість використання.</td><td>1. Побудова лояльності</td><td rowspan="2">1. Автотранспортні підприємтва.
2. Приватні міжміські водії</td></tr>
<tr><td>Ключові ресурси
1. Персонал.
2. Серверне обладнання.</td><td>Канали
1. Соціальні мережі.
2. Контекстна реклама.</td></tr>
<tr><th colspan="3">Структура витрат</th><th colspan="2">Джерела доходу</th></tr>
<tr><td colspan="3">1. Фіксовані витрати (оренда приміщення, з/п персоналу)
2. Змінні витрати (придбання оборотних коштів)</td><td colspan="2">1. Дохід від угод.
2. Дохід від реалізації особистих побажань клієнтів.
3. Підписки.</td></tr>
</table>

ФІНАНСОВИЙ ПЛАН СТАРТАПУ

ДОХОДИ (грн)	Довготривалі підписки	Усі функції системи	Обрані функції системи	Розширена технічна підтримка	Всього
Сумарно за рік:	147000	74000	54500	68000	401500

ВИТРАТИ (грн)	Технічна підтримка	Оплата праці	Оренда та комунальні витрати	Реклама	Всього
Сумарно за рік:	10500	155000	60000	47000	272500

ПРИБУТОК (грн) = 129 000

ВИМОГИ ДЛЯ ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Процесор – 1 ГГц.
2. Обсяг оперативної пам’яті – 512 Мб.
3. Будь-яка операційна система
4. Доступ до мережі Інтернет

Наукова новизна

1. Запропонований метод прогнозування пасажиропотоку з використанням нейронних мереж дозволяє зменшити собівартість пасажирських перевезень та покращити якість надання транспортних послуг на досліджуваних маршрутах.
2. Отримані уточнені прогнозні значення на досліджуваних маршрутах дозволяють покращувати роботу АТП, зокрема при плануванні режимів руху та виборі транспортних засобів для перевезення пасажирів.

Висновки

1. Здійснено аналіз існуючих рішень та методів прогнозування пасажиропотоку.
2. Розроблено програмний метод для оптимізації міжміських пасажирських перевезень з використанням LSTM нейронної мережі.
3. Здійснено тестування та порівняння результатів дослідження з існуючими методами.
4. Розроблено програмне забезпечення для прогнозування пасажиропотоку та планування рухомого складу залежно від отриманих прогнозних значень з використанням LSTM нейронної мережі.
5. Розроблено бізнес модель впровадження програмного забезпечення.

Апробація

1. XI наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» (ПМК-2018-2).
2. X науково-технічна конференції «Комп'ютерні системи і мережні технології 2017» (м. Київ).
3. «Програмне забезпечення для прогнозування пасажиропотоку та організації рухомого складу АТП» в міжнародному науковому журналі "Інтернаука". — 2017. — №11.

Результати перевірки тексту дисертації на предмет плагіату

Вступ	Розділ 1	Розділ 2	Розділ 3	Розділ 4	Розділ 5	Висновок	Загалом
98%	79%	90%	90%	92%	93%	91%	87%

ДЯКУЮ ЗА УВАГУ!

Додаток 2

Лістинг програми

Лістинг 1. Bundle.js

```
(window["webpackJsonp"] = window["webpackJsonp"] || []).push([["main"],{
module.exports = ["en","ru","uk"];

/***/ }),

/***/ "../node_modules/@diplom-web/assets/localeData lazy recursive
^\\.\\/.*$":
/*!*************************************************************************
*************!*\
  !*** ../node_modules/@diplom-web/assets/localeData lazy ^\.\/.*$
namespace object ***!

\*************************************************************************
***********/
/*! no static exports found */
/***/ (function(module, exports, __webpack_require__) {

var map = {
        "./en": [
                "../assets/localeData/en.js",
                "locale-data-en"
        ],
        "./en.js": [
                "../assets/localeData/en.js",
                "locale-data-en"
        ],
        "./ru": [
                "../assets/localeData/ru.js",
                "locale-data-ru"
        ],
        "./ru.js": [
                "../assets/localeData/ru.js",
                "locale-data-ru"
        ],
        "./uk": [
                "../assets/localeData/uk.js",
                "locale-data-uk"
        ],
        "./uk.js": [
                "../assets/localeData/uk.js",
                "locale-data-uk"
        ]
};
function webpackAsyncContext(req) {
        var ids = map[req];
        if(!ids) {
                return Promise.resolve().then(function() {
                        var e = new Error("Cannot find module '" + req +
"'");
                        e.code = 'MODULE_NOT_FOUND';
                        throw e;
                });
        }
        return __webpack_require__.e(ids[1]).then(function() {
                var id = ids[0];
                return __webpack_require__.t(id, 7);
        });
}
webpackAsyncContext.keys = function webpackAsyncContextKeys() {
        return Object.keys(map);
};
```

```
webpackAsyncContext.id = "../node_modules/@diplom-web/assets/localeData
lazy recursive ^\\.\\/.*$";
module.exports = webpackAsyncContext;

/***/ }),

/***/ "../node_modules/@diplom-web/assets/messages lazy recursive
^\\.\\/.*$":
/*!*****************************************************************************
***********!*\
  !*** ../node_modules/@diplom-web/assets/messages lazy ^\.\/.*$ namespace
object ***!

\*****************************************************************************
*********/
/*! no static exports found */
/***/ (function(module, exports, __webpack_require__) {

var map = {
        "./en": [
                "../assets/messages/en.json",
                "messages-en"
        ],
        "./en.json": [
                "../assets/messages/en.json",
                "messages-en"
        ],
        "./ru": [
                "../assets/messages/ru.json",
                "messages-ru"
        ],
        "./ru.json": [
                "../assets/messages/ru.json",
                "messages-ru"
        ],
        "./uk": [
                "../assets/messages/uk.json",
                "messages-uk"
        ],
        "./uk.json": [
                "../assets/messages/uk.json",
                "messages-uk"
        ]
};
function webpackAsyncContext(req) {
        var ids = map[req];
        if(!ids) {
                return Promise.resolve().then(function() {
                        var e = new Error("Cannot find module '" + req +
"'");
                        e.code = 'MODULE_NOT_FOUND';
                        throw e;
                });
        }
        return __webpack_require__.e(ids[1]).then(function() {
                var id = ids[0];
                return __webpack_require__.t(id, 3);
        });
}
webpackAsyncContext.keys = function webpackAsyncContextKeys() {
        return Object.keys(map);
};
webpackAsyncContext.id = "../node_modules/@diplom-web/assets/messages lazy
recursive ^\\.\\/.*$";
```

```
module.exports = webpackAsyncContext;

/***/ }),

/***/ "./src/App.jsx":
/*!*********************!*\
  !*** ./src/App.jsx ***!
  \*********************/
/*! exports provided: default */
/***/ (function(module, __webpack_exports__, __webpack_require__) {

"use strict";
__webpack_require__.r(__webpack_exports__);
/* harmony import */ var
C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_modules
_babel_runtime_helpers_esm_slicedToArray__WEBPACK_IMPORTED_MODULE_0__ =
__webpack_require__(/*! ../node_modules/babel-preset-react-
app/node_modules/@babel/runtime/helpers/esm/slicedToArray */
"../node_modules/babel-preset-react-
app/node_modules/@babel/runtime/helpers/esm/slicedToArray.js");
/* harmony import */ var
C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_modules
_babel_runtime_helpers_esm_toConsumableArray__WEBPACK_IMPORTED_MODULE_1__ =
__webpack_require__(/*! ../node_modules/babel-preset-react-
app/node_modules/@babel/runtime/helpers/esm/toConsumableArray */
"../node_modules/babel-preset-react-
app/node_modules/@babel/runtime/helpers/esm/toConsumableArray.js");
/* harmony import */ var _date_io_luxon__WEBPACK_IMPORTED_MODULE_2__ =
__webpack_require__(/*! @date-io/luxon */ "../node_modules/@date-
io/luxon/build/index.esm.js");
/* harmony import */ var _material_ui_core__WEBPACK_IMPORTED_MODULE_3__ =
__webpack_require__(/*! @material-ui/core */ "../node_modules/@material-
ui/core/index.es.js");
/* harmony import */ var _material_ui_styles__WEBPACK_IMPORTED_MODULE_4__ =
__webpack_require__(/*! @material-ui/styles */ "../node_modules/@material-
ui/styles/index.es.js");
/* harmony import */ var jss__WEBPACK_IMPORTED_MODULE_5__ =
__webpack_require__(/*! jss */ "../node_modules/jss/lib/index.js");
/* harmony import */ var jss__WEBPACK_IMPORTED_MODULE_5___default =
/*#__PURE__*/__webpack_require__.n(jss__WEBPACK_IMPORTED_MODULE_5__);
/* harmony import */ var jss_expand__WEBPACK_IMPORTED_MODULE_6__ =
__webpack_require__(/*! jss-expand */ "../node_modules/jss-
expand/lib/index.js");
/* harmony import */ var jss_expand__WEBPACK_IMPORTED_MODULE_6___default =
/*#__PURE__*/__webpack_require__.n(jss_expand__WEBPACK_IMPORTED_MODULE_6__)
;
/* harmony import */ var luxon__WEBPACK_IMPORTED_MODULE_7__ =
__webpack_require__(/*! luxon */ "../node_modules/luxon/build/cjs-
browser/luxon.js");
/* harmony import */ var luxon__WEBPACK_IMPORTED_MODULE_7___default =
/*#__PURE__*/__webpack_require__.n(luxon__WEBPACK_IMPORTED_MODULE_7__);
/* harmony import */ var material_ui_pickers__WEBPACK_IMPORTED_MODULE_8__ =
__webpack_require__(/*! material-ui-pickers */ "../node_modules/material-
ui-pickers/dist/material-ui-pickers.esm.js");
/* harmony import */ var react__WEBPACK_IMPORTED_MODULE_9__ =
__webpack_require__(/*! react */ "../node_modules/react/index.js");
/* harmony import */ var react__WEBPACK_IMPORTED_MODULE_9___default =
/*#__PURE__*/__webpack_require__.n(react__WEBPACK_IMPORTED_MODULE_9__);
/* harmony import */ var react_redux__WEBPACK_IMPORTED_MODULE_10__ =
__webpack_require__(/*! react-redux */ "../node_modules/react-
redux/es/index.js");
/* harmony import */ var react_router_dom__WEBPACK_IMPORTED_MODULE_11__ =
__webpack_require__(/*! react-router-dom */ "../node_modules/react-router-
dom/esm/react-router-dom.js");
```

```
/* harmony import */ var
react_virtualized_styles_css__WEBPACK_IMPORTED_MODULE_12__ =
__webpack_require__(/*! react-virtualized/styles.css */
"../node_modules/react-virtualized/styles.css");
/* harmony import */ var
react_virtualized_styles_css__WEBPACK_IMPORTED_MODULE_12___default =
/*#__PURE__*/__webpack_require__.n(react_virtualized_styles_css__WEBPACK_IM
PORTED_MODULE_12__);
/* harmony import */ var
redux_persist_integration_react__WEBPACK_IMPORTED_MODULE_13__ =
__webpack_require__(/*! redux-persist/integration/react */
"../node_modules/redux-persist/es/integration/react.js");
/* harmony import */ var
_containers_Localized__WEBPACK_IMPORTED_MODULE_14__ =
__webpack_require__(/*! ./containers/Localized */
"./src/containers/Localized.jsx");
/* harmony import */ var _store__WEBPACK_IMPORTED_MODULE_15__ =
__webpack_require__(/*! ./store */ "./src/store.js");
/* harmony import */ var
_utils_getCurrentLocale__WEBPACK_IMPORTED_MODULE_16__ =
__webpack_require__(/*! ./utils/getCurrentLocale */
"./src/utils/getCurrentLocale.js");
/* harmony import */ var _utils_PrivateRoute__WEBPACK_IMPORTED_MODULE_17__
= __webpack_require__(/*! ./utils/PrivateRoute */
"./src/utils/PrivateRoute.js");
/* harmony import */ var _utils_themes__WEBPACK_IMPORTED_MODULE_18__ =
__webpack_require__(/*! ./utils/themes */ "./src/utils/themes.js");


var _jsxFileName = "C:\\Users\\vova\\diplom-web-ng\\core\\src\\App.jsx";
```



```
var jss = Object(jss__WEBPACK_IMPORTED_MODULE_5__["create"])({
  plugins:
Object(C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_
modules_babel_runtime_helpers_esm_toConsumableArray__WEBPACK_IMPORTED_MODUL
E_1__["default"])(Object(_material_ui_styles__WEBPACK_IMPORTED_MODULE_4__["
jssPreset"])().plugins).concat([jss_expand__WEBPACK_IMPORTED_MODULE_6___def
ault()()])
});
luxon__WEBPACK_IMPORTED_MODULE_7__["Settings"].defaultZoneName = "utc"; //
Split routes in chunks and load them instantly

var imports = Object.entries({
  Appointment: Promise.all(/*! import() | route-appointment
*/[__webpack_require__.e(0), __webpack_require__.e(1),
__webpack_require__.e(2), __webpack_require__.e(5),
__webpack_require__.e(3), __webpack_require__.e(6),
__webpack_require__.e("route-
```

```
appointment")]).then(__webpack_require__.bind(null, /*!
./containers/Appointment */ "./src/containers/Appointment.jsx")),
  Vodliteli: Promise.all(/*! import() | route-vodliteli
*/[__webpack_require__.e(0), __webpack_require__.e(1),
__webpack_require__.e(2), __webpack_require__.e(5),
__webpack_require__.e(3), __webpack_require__.e(6),
__webpack_require__.e("route-
vodliteli")]).then(__webpack_require__.bind(null, /*!
./containers/Vodliteli */ "./src/containers/Vodliteli.jsx")),
  Login: Promise.all(/*! import() | route-login
*/[__webpack_require__.e(0), __webpack_require__.e(2),
__webpack_require__.e(4), __webpack_require__.e(8),
__webpack_require__.e("route-login")]).then(__webpack_require__.bind(null,
/*! ./containers/Login */ "./src/containers/Login.jsx")),
  Schedule: Promise.all(/*! import() | route-schedule
*/[__webpack_require__.e(0), __webpack_require__.e(1),
__webpack_require__.e(4), __webpack_require__.e(3),
__webpack_require__.e("route-
schedule")]).then(__webpack_require__.bind(null, /*! ./containers/Schedule
*/ "./src/containers/Schedule.jsx"))
}).reduce(function (acc, _ref) {
  var _ref2 =
Object(C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_
modules_babel_runtime_helpers_esm_slicedToArray__WEBPACK_IMPORTED_MODULE_0_
_["default"])(_ref, 2),
      name = _ref2[0],
      promise = _ref2[1];

  acc[name] = Object(react__WEBPACK_IMPORTED_MODULE_9__["lazy"])(function
() {
    return promise;
  });
  return acc;
}, {});
var locale =
Object(_utils_getCurrentLocale__WEBPACK_IMPORTED_MODULE_16__["default"])().
base;

var Router = function Router() {
  return
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(react_router_dom
__WEBPACK_IMPORTED_MODULE_11__["Switch"], {
    __source: {
      fileName: _jsxFileName,
      lineNumber: 38
    },
    __self: this
  },
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(react_router_dom
__WEBPACK_IMPORTED_MODULE_11__["Route"], {
    path: "/login",
    component: imports.Login,
    __source: {
      fileName: _jsxFileName,
      lineNumber: 39
    },
    __self: this
  }),
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_utils_PrivateRo
ute__WEBPACK_IMPORTED_MODULE_17__["default"], {
    path: "/appointment",
    component: imports.Appointment,
    __source: {
      fileName: _jsxFileName,
```

```
      lineNumber: 40
    },
    __self: this
  }),
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_utils_PrivateRo
ute__WEBPACK_IMPORTED_MODULE_17__["default"], {
    path: "/vodliteli",
    component: imports.Vodliteli,
    __source: {
      fileName: _jsxFileName,
      lineNumber: 41
    },
    __self: this
  }),
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_utils_PrivateRo
ute__WEBPACK_IMPORTED_MODULE_17__["default"], {
    path: "/schedule",
    component: imports.Schedule,
    __source: {
      fileName: _jsxFileName,
      lineNumber: 42
    },
    __self: this
  }),
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(react_router_dom
__WEBPACK_IMPORTED_MODULE_11__["Redirect"], {
    to: "/schedule",
    __source: {
      fileName: _jsxFileName,
      lineNumber: 43
    },
    __self: this
  }));
};

var App = function App() {
  return
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_material_ui_sty
les__WEBPACK_IMPORTED_MODULE_4__["StylesProvider"], {
    jss: jss,
    __source: {
      fileName: _jsxFileName,
      lineNumber: 48
    },
    __self: this
  },
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_material_ui_sty
les__WEBPACK_IMPORTED_MODULE_4__["ThemeProvider"], {
    theme: _utils_themes__WEBPACK_IMPORTED_MODULE_18__["main"],
    __source: {
      fileName: _jsxFileName,
      lineNumber: 49
    },
    __self: this
  },
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(react__WEBPACK_I
MPORTED_MODULE_9__["Suspense"], {
    fallback:
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_material_ui_cor
e__WEBPACK_IMPORTED_MODULE_3__["CircularProgress"], {
      __source: {
        fileName: _jsxFileName,
        lineNumber: 50
      },
```

```
      __self: this
    }),
    __source: {
      fileName: _jsxFileName,
      lineNumber: 50
    },
    __self: this
  },
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(react_redux__WEB
PACK_IMPORTED_MODULE_10__["Provider"], Object.assign({
    store: _store__WEBPACK_IMPORTED_MODULE_15__["store"]
  }, {
    __source: {
      fileName: _jsxFileName,
      lineNumber: 51
    },
    __self: this
  }),
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(redux_persist_in
tegration_react__WEBPACK_IMPORTED_MODULE_13__["PersistGate"],
Object.assign({
    loading:
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_material_ui_cor
e__WEBPACK_IMPORTED_MODULE_3__["CircularProgress"], {
      __source: {
        fileName: _jsxFileName,
        lineNumber: 52
      },
      __self: this
    })
  }, {
    persistor: _store__WEBPACK_IMPORTED_MODULE_15__["persistor"]
  }, {
    __source: {
      fileName: _jsxFileName,
      lineNumber: 52
    },
    __self: this
  }),
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(react_router_dom
__WEBPACK_IMPORTED_MODULE_11__["BrowserRouter"], {
    __source: {
      fileName: _jsxFileName,
      lineNumber: 53
    },
    __self: this
  },
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_containers_Loca
lized__WEBPACK_IMPORTED_MODULE_14__["default"], {
    __source: {
      fileName: _jsxFileName,
      lineNumber: 54
    },
    __self: this
  },
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(material_ui_pick
ers__WEBPACK_IMPORTED_MODULE_8__["MuiPickersUtilsProvider"], {
    utils: _date_io_luxon__WEBPACK_IMPORTED_MODULE_2__["default"],
    locale: locale,
    __source: {
      fileName: _jsxFileName,
      lineNumber: 55
    },
    __self: this
```

```
  },
react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(_material_ui_cor
e__WEBPACK_IMPORTED_MODULE_3__["CssBaseline"], {
    __source: {
      fileName: _jsxFileName,
      lineNumber: 56
    },
    __self: this
  }), react__WEBPACK_IMPORTED_MODULE_9___default.a.createElement(Router, {
    __source: {
      fileName: _jsxFileName,
      lineNumber: 57
    },
    __self: this
  }))))))))));
};

/* harmony default export */ __webpack_exports__["default"] = (App);

/***/ }),

/***/ "./src/actions/authorization.js":
/*!**************************************!*\
  !*** ./src/actions/authorization.js ***!
  \**************************************/
/*! exports provided: getFreshAccessToken, login, logout,
refreshAccessToken, changeLocation */
/***/ (function(module, __webpack_exports__, __webpack_require__) {

"use strict";
__webpack_require__.r(__webpack_exports__);
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"getFreshAccessToken", function() { return getFreshAccessToken; });
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"login", function() { return login; });
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"logout", function() { return logout; });
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"refreshAccessToken", function() { return refreshAccessToken; });
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"changeLocation", function() { return changeLocation; });
/* harmony import */ var
C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_modules
_babel_runtime_regenerator__WEBPACK_IMPORTED_MODULE_0__ =
__webpack_require__(/*! ../node_modules/babel-preset-react-
app/node_modules/@babel/runtime/regenerator */ "../node_modules/babel-
preset-react-app/node_modules/@babel/runtime/regenerator/index.js");
/* harmony import */ var
C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_modules
_babel_runtime_regenerator__WEBPACK_IMPORTED_MODULE_0___default =
/*#__PURE__*/__webpack_require__.n(C_Users_vova_diplom_web_ng_node_modules_
babel_preset_react_app_node_modules_babel_runtime_regenerator__WEBPACK_IMPO
RTED_MODULE_0__);
/* harmony import */ var
C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_modules
_babel_runtime_helpers_esm_asyncToGenerator__WEBPACK_IMPORTED_MODULE_1__ =
__webpack_require__(/*! ../node_modules/babel-preset-react-
app/node_modules/@babel/runtime/helpers/esm/asyncToGenerator */
"../node_modules/babel-preset-react-
app/node_modules/@babel/runtime/helpers/esm/asyncToGenerator.js");
/* harmony import */ var luxon__WEBPACK_IMPORTED_MODULE_2__ =
__webpack_require__(/*! luxon */ "../node_modules/luxon/build/cjs-
browser/luxon.js");
```

```
/* harmony import */ var luxon__WEBPACK_IMPORTED_MODULE_2___default =
/*#__PURE__*/__webpack_require__.n(luxon__WEBPACK_IMPORTED_MODULE_2__);
/* harmony import */ var redux_act__WEBPACK_IMPORTED_MODULE_3__ =
__webpack_require__(/*! redux-act */ "../node_modules/redux-
act/lib/index.js");
/* harmony import */ var redux_act__WEBPACK_IMPORTED_MODULE_3___default =
/*#__PURE__*/__webpack_require__.n(redux_act__WEBPACK_IMPORTED_MODULE_3__);
/* harmony import */ var _utils_callApi__WEBPACK_IMPORTED_MODULE_4__ =
__webpack_require__(/*! ../utils/callApi */ "./src/utils/callApi.js");
/* harmony import */ var _utils_constants__WEBPACK_IMPORTED_MODULE_5__ =
__webpack_require__(/*! ../utils/constants */ "./src/utils/constants.js");
/* harmony import */ var _settings__WEBPACK_IMPORTED_MODULE_6__ =
__webpack_require__(/*! ./settings */ "./src/actions/settings.js");
```

```
var refreshAccessTokenPromise = null;
var getFreshAccessToken = function getFreshAccessToken() {
  return (
    /*#__PURE__*/
    function () {
      var _ref =
Object(C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_
modules_babel_runtime_helpers_esm_asyncToGenerator__WEBPACK_IMPORTED_MODULE
_1__["default"])(
      /*#__PURE__*/

C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_modules
_babel_runtime_regenerator__WEBPACK_IMPORTED_MODULE_0___default.a.mark(func
tion _callee(dispatch, getState) {
        var _getState, expiresAt;

        return
C_Users_vova_diplom_web_ng_node_modules_babel_preset_react_app_node_modules
_babel_runtime_regenerator__WEBPACK_IMPORTED_MODULE_0___default.a.wrap(func
tion _callee$(_context) {
          while (1) {
            switch (_context.prev = _context.next) {
              case 0:
                _getState = getState(), expiresAt =
_getState.authorization.expiresAt;

                if
(!(luxon__WEBPACK_IMPORTED_MODULE_2__["DateTime"].fromISO(expiresAt).diffNo
w("hours").hours < 1)) {
                  _context.next = 6;
                  break;
                }

                if (!refreshAccessTokenPromise) {
                  refreshAccessTokenPromise =
dispatch(refreshAccessToken()).promise;
                }

                _context.next = 5;
                return refreshAccessTokenPromise;

              case 5:
                refreshAccessTokenPromise = null;
```

```
              case 6:
                return _context.abrupt("return",
getState().authorization.accessToken);

              case 7:
              case "end":
                return _context.stop();
            }
          }
        }, _callee, this);
      }));

      return function (_x, _x2) {
        return _ref.apply(this, arguments);
      };
    }()
  );
};
var login =
Object(redux_act__WEBPACK_IMPORTED_MODULE_3__["createAction"])("Login");
var logoutAction =
Object(redux_act__WEBPACK_IMPORTED_MODULE_3__["createAction"])("Logout");
var logout = function logout() {
  return function (dispatch) {
    dispatch(logoutAction());

dispatch(Object(_settings__WEBPACK_IMPORTED_MODULE_6__["clearData"])());
  };
};
logout.toString = logoutAction.toString;
var refreshAccessToken =
Object(_utils_callApi__WEBPACK_IMPORTED_MODULE_4__["default"])({
  accessTokenRequired: false,
  description: "Refresh access token",
  requestTransformer: function requestTransformer(_ref2) {
    var getState = _ref2.getState;
    return {
      url: "auth/refresh",
      params: {
        application_id:
_utils_constants__WEBPACK_IMPORTED_MODULE_5__["APPLICATION_ID"],
        refresh_token: getState().authorization.refreshToken
      }
    };
  },
  responseTransformer: function responseTransformer(_ref3) {
    var _ref3$response$data = _ref3.response.data,
        accessToken = _ref3$response$data.access_token,
        expiresAt = _ref3$response$data.expires_at,
        refreshToken = _ref3$response$data.refresh_token;
    return {
      payload: {
        accessToken: accessToken,
        expiresAt: expiresAt,
        refreshToken: refreshToken
      }
    };
  }
});
var changeLocation =
Object(_utils_callApi__WEBPACK_IMPORTED_MODULE_4__["default"])({
  description: "Change location",
  requestTransformer: function requestTransformer(_ref4) {
    var location = _ref4.payload.location,
```

```
        getState = _ref4.getState;

    var _getState2 = getState(),
        refreshToken = _getState2.authorization.refreshToken,
        locations = _getState2.locations;

    return {
      url: "auth/employee",
      params: {
        new_database: locations[location].database,
        refresh_token: refreshToken
      }
    };
  },
  responseTransformer: function responseTransformer(_ref5) {
    var payload = _ref5.payload;
    return {
      payload: payload
    };
  }
});

/***/ }),

/***/ "./src/actions/vodliteliCategories.js":
/*!********************************************!*\
  !*** ./src/actions/vodliteliCategories.js ***!
  \********************************************/
/*! exports provided: fetchVodliteliCategories */
/***/ (function(module, __webpack_exports__, __webpack_require__) {

"use strict";
__webpack_require__.r(__webpack_exports__);
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"fetchVodliteliCategories", function() { return fetchVodliteliCategories;
});
/* harmony import */ var normalizr__WEBPACK_IMPORTED_MODULE_0__ =
__webpack_require__(/*! normalizr */
"../node_modules/normalizr/dist/normalizr.es.js");
/* harmony import */ var _utils_callApi__WEBPACK_IMPORTED_MODULE_1__ =
__webpack_require__(/*! ../utils/callApi */ "./src/utils/callApi.js");


var vodliteliCategoriesSchema = new
normalizr__WEBPACK_IMPORTED_MODULE_0__["schema"].Entity("vodliteliCategorie
s");
var vodliteliCategoriesListSchema = [vodliteliCategoriesSchema];
var fetchVodliteliCategories =
Object(_utils_callApi__WEBPACK_IMPORTED_MODULE_1__["default"])({
  description: "Fetch vodliteli categories",
  requestTransformer: function requestTransformer() {
    return {
      url: "vodlitelicategories"
    };
  },
  responseTransformer: function responseTransformer(_ref) {
    var normalizedData = _ref.response.normalizedData;
    return {
      payload: normalizedData.entities.vodliteliCategories || {}
    };
  },
  schema: vodliteliCategoriesListSchema
});
```

```
/***/ }),

/***/ "./src/actions/vodliteliHistory.js":
/*!*****************************************!*\
  !*** ./src/actions/vodliteliHistory.js ***!
  \*****************************************/
/*! exports provided: fetchVodliteliHistory */
/***/ (function(module, __webpack_exports__, __webpack_require__) {

"use strict";
__webpack_require__.r(__webpack_exports__);
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"fetchVodliteliHistory", function() { return fetchVodliteliHistory; });
/* harmony import */ var _utils_callApi__WEBPACK_IMPORTED_MODULE_0__ =
__webpack_require__(/*! ../utils/callApi */ "./src/utils/callApi.js");
// import { schema } from "normalizr";
 // const vodliteliHistorySchema = new schema.Entity("vodliteliHistory");
// const vodliteliHistoryListSchema = [vodliteliHistorySchema];

var fetchVodliteliHistory =
Object(_utils_callApi__WEBPACK_IMPORTED_MODULE_0__["default"])({
  description: "Fetch vodliteli history",
  requestTransformer: function requestTransformer(_ref) {
    var id = _ref.payload.id;
    return {
      url: "vodliteli/".concat(id, "/history")
    };
  },
  responseTransformer: function responseTransformer(_ref2) {
    var id = _ref2.request.id,
        data = _ref2.response.data;
    return {
      payload: {
        vodliteli: id,
        history: data.reverse()
      }
    };
  } //   schema: vodliteliHistoryListSchema,

});

/***/ }),

/***/ "./src/actions/vodliteli.js":
/*!**********************************!*\
  !*** ./src/actions/vodliteli.js ***!
  \**********************************/
/*! exports provided: fetchVodliteliList, updateVodliteli, createVodliteli,
deleteVodliteli, updateVodliteliPhoto, resetCreatedVodliteli */
/***/ (function(module, __webpack_exports__, __webpack_require__) {

"use strict";
__webpack_require__.r(__webpack_exports__);
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"fetchVodliteliList", function() { return fetchVodliteliList; });
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"updateVodliteli", function() { return updateVodliteli; });
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"createVodliteli", function() { return createVodliteli; });
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"deleteVodliteli", function() { return deleteVodliteli; });
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"updateVodliteliPhoto", function() { return updateVodliteliPhoto; });
```

```
/* harmony export (binding) */ __webpack_require__.d(__webpack_exports__,
"resetCreatedVodliteli", function() { return resetCreatedVodliteli; });
/* harmony import */ var normalizr__WEBPACK_IMPORTED_MODULE_0__ =
__webpack_require__(/*! normalizr */
"../node_modules/normalizr/dist/normalizr.es.js");
/* harmony import */ var redux_act__WEBPACK_IMPORTED_MODULE_1__ =
__webpack_require__(/*! redux-act */ "../node_modules/redux-
act/lib/index.js");
/* harmony import */ var redux_act__WEBPACK_IMPORTED_MODULE_1___default =
/*#__PURE__*/__webpack_require__.n(redux_act__WEBPACK_IMPORTED_MODULE_1__);
/* harmony import */ var _utils_callApi__WEBPACK_IMPORTED_MODULE_2__ =
__webpack_require__(/*! ../utils/callApi */ "./src/utils/callApi.js");



var vodlitelichema = new
normalizr__WEBPACK_IMPORTED_MODULE_0__["schema"].Entity("vodliteli");
var vodliteliListSchema = [vodlitelichema];
var vodliteliFields = ["additional_fields", "balance", "birthday", "bonus",
"categories_names", "categories", "email", "feedback",
"first_visit_description", "first_visit", "last_visit_description",
"last_visit", "location", "name_parts", "name", "phone", "photo_exists",
"sex"].join();
var fetchVodliteliList =
Object(_utils_callApi__WEBPACK_IMPORTED_MODULE_2__["default"])({
  description: "Fetch vodliteli list",
  requestTransformer: function requestTransformer() {
    return {
      url: "vodliteli",
      params: {
        fields: vodliteliFields
      }
    };
  },
  responseTransformer: function responseTransformer(_ref) {
    var normalizedData = _ref.response.normalizedData;
    return {
      payload: normalizedData.entities.vodliteli || {}
    };
  },
  schema: vodliteliListSchema
});
var updateVodliteli =
Object(_utils_callApi__WEBPACK_IMPORTED_MODULE_2__["default"])({
  description: "Update vodliteli",
  requestTransformer: function requestTransformer(_ref2) {
    var _ref2$payload = _ref2.payload,
        id = _ref2$payload.id,
        data = _ref2$payload.data;
    return {
      url: "vodliteli/".concat(id),
      method: "put",
      data: data,
      params: {
        fields: vodliteliFields,
        info: true
      }
    };
  }
});
var createVodliteli =
Object(_utils_callApi__WEBPACK_IMPORTED_MODULE_2__["default"])({
  description: "Create vodliteli",
  requestTransformer: function requestTransformer(_ref3) {
```