Міністерство освіти і науки України
Ніжинський державний університет імені Миколи Гоголя
ННІ природничо-математичних, медико-біологічних наук та інформаційних технологій
Кафедра інформаційних технологій, фізико-математичних та економічних наук

Освітньо-професійна програма:
Комп'ютерні науки
зі спеціальності 122 Комп'ютерні науки

КВАЛІФІКАЦІЙНА РОБОТА

На здобуття освітнього ступеня магістр

РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ РОБОТИ ЛІКАРЯ РЕАБІЛІТОЛОГА

Студента: Халанського Павла Олеговича

Науковий керівник: Завідувач кафедри інформаційних технологій, фізико-математичних та економічних наук, професор Казачков Іван Васильович

Рецензенти: Нестеренко Олександр
д. т. н., професор, завідувач кафедри інформаційних технологій,
Міжнародний європейський університет

Допущено до захисту _____

Завідувач кафедри інформаційних технологій,
фізико-математичних та економічних наук,
професор Казачков Іван Васильович

Ніжин - 2024

Анотація

Дана магістерська робота представлена на 69 листах, містить в собі 2 ілюстрацій, 5 діаграм, 2 таблиць, ілюстративний матеріал (презентацію).

Тема роботи: Розробка програмного застосунку для роботи лікаря реабілітолога.

Мета роботи: розробка та планування програмного застосунку на основі потреб лікаря у комфортному та коректному веденні своєї роботи щодо пошкоджень і травм пацієнтів, створення зручного та зрозумілого додатку для підвищення якості роботи лікарів та оптимізації їх робочого часу

Актуальність теми: актуальність даної теми достатньо високого рівня через застарілу та недбалу систематизацію роботи, яка існує на даний момент у великої кількості медичних закладів,в тому числі і найбільшої проблеми, такої як паперовий документообіг.

Об’єктом роботи є лікар із своїми професійними даними стосовно пацієнтів, та самі пацієнти, кожен із яких має свій індивідуальний діагноз, план реабілітації, дані та способи і тривалість лікування, призначені лікарем безпосередньо.

У першому розділі будуть представлені відомості з теорії фізичної реабілітації, інформація стосовно роботи фахівців у даному напрямку медицини.

У другому розділі безпосередній метод і матеріал досліджень даної роботи.

У третьому розділі було розписано практичні рішення задачі та сам алгоритм.

По результатам роботи були зроблені висновки щодо розробки програмного застосунку для лікаря реабілітолога.

Ключові слова: лікар, реабілітолог, гаджет, пацієнт, дані, реабілітація, система, лікування, система, оптимізація, комфорт, робота, алгоритм, інтерфейс.

Abstract

This master's thesis is presented on 69 sheets, contains 22 illustrations, 5 diagrams, 2 tables, illustrative material (presentation).

Topic of the work: Development of a software application for a rehabilitation doctor.

Purpose of the work: development and planning of a software application based on the doctor's needs for comfortable and correct conduct of his work regarding injuries and traumas of patients, creation of a convenient and understandable application for improving the quality of work of doctors and optimization of their working time

Relevance of the topic: the relevance of this topic is of a sufficiently high level due to the outdated and careless systematization of work, which currently exists in a large number of medical institutions, including the biggest problem, such as paper document flow.

The object of the work is a doctor with his professional data regarding patients, and the patients themselves, each of whom has his own individual diagnosis, rehabilitation plan, data and methods and duration of treatment, prescribed by the doctor directly.

The first section will present information on the theory of physical rehabilitation, information about the work of specialists in this area of medicine.

The second section presents the direct method and research material of this work.

The third section describes practical solutions to the problem and the algorithm itself.

Based on the results of the work, conclusions were drawn regarding the development of a software application for a rehabilitation doctor.

Keywords: doctor, rehabilitation doctor, gadget, patient, data, rehabilitation, system, treatment, system, optimization, comfort, work, algorithm, interface.

ЗМІСТ

ВСТУП

Сучасний світ стрімко розвивається, технологічний процес досягає величещного успіху у багатьох сферах нашого життя, роблячи його простішим і більш комфортним. Це стосується як і повсякденних речей, розваг, техніки в побуті так і найбільших масштабів станцій, енергетики і медицини. Систему роботи будь-якої галузі хочуть звести до максимального зменшення затрати людської роботи, ресурсу та витраченого часу, оскільки все це під час втоми людини може привести до людського фактору, такого як втома, втрата уваги та пильності, прості помилки, що у дуже важливих сферах може привести до катастрофічних наслідків і негативних результатів. Але, попри величезний прогрес, є сфери, у які автоматизація ніяким чином не замінить людину. У таких випадках за допомогою технологій людство прагне не замінити людину, а полегшити її роботу, надаючи найбільш комфортні умови для її подальшої праці, ставлячи саму людину «керівником» технологій.

Актуальність: максимальне зменшення документообігу, створення за стосунку для ведення, обміну, коригування інформації між лікарем і пацієнтом та лікарями різного роду всередині лікарні для зручності. Створення комфортного порядку роботи лікаря, щоб зменшити кількість затраченого часу та сил на паперові записи від руки, їх заповнення, постійне коригування заново та фізичну діяльність, тобто перенос документів до архіву, ведення щоденників, історій та індивідуального плану пацієнта.

РОЗДІЛ 1
ЛІТЕРАТУРНИЙ ОГЛЯД

1.1 Фізична реабілітація

Процес фізичної реабілітації – це сукупність професійних аспектів клінічної освіти і практики, собою являє окремий самодостатній напрямок. Для можливості працювати у даній сфері, людина повинна здобути спеціальну кваліфікацію у призначених для цього освітніх закладах. Право користуватись даною професією може бути надано фахівцю після успішного завершення навчального закладу, саме після цього людина може влаштуватись незалежним спеціалістом[2].

Лікар реабілітолог – це спеціаліст, кінцевою метою якого є відновлення всіх функцій людини до нормального стану. Фахівець в даній області не лікує травми безпосередньо, це напрям роботи інших лікарів, реабілітолог, в свою чергу бере під контроль лише процес відновлення. Він повинен стабілізувати навантаження на пацієнта, призначати та спостерігати за спеціальними процедурами, тримати під увагою весь процес відновлення.

1.2 Фахівці з фізичної реабілітації

Реабілітолог виконує ключову роль у процесі відновлення здоров’я пацієнтів, зокрема його основним завданням є допомога у відновленні функцій організму, які були втрачені внаслідок травм, захворювань або інших проблем зі здоров’ям. Його робота полягає не тільки в тому, щоб полегшити симптоми захворювання, але й максимально сприяти їхньому усуненню, створюючи умови для якісного покращення фізичного стану пацієнта.

Для досягнення цього реабілітолог має володіти не лише фундаментальною професійною освітою, але й спеціалізованими знаннями в галузі фізичної реабілітації. Це включає методи, які через цілеспрямований вплив на м’язову систему активізують роботу найважливіших систем організму, таких як нервова, кровоносна чи імунна, забезпечуючи гармонійне функціонування всього тіла. Підготовка спеціаліста також включає практичні навички застосування сучасних засобів та технологій реабілітації, які допомагають пацієнтові ефективніше долати наслідки хвороби.

Таким чином, реабілітолог не лише сприяє фізичному відновленню, але й виступає провідником до повернення людини до звичного життя, забезпечуючи підтримку та мотивацію на кожному етапі лікування.

Ефективне застосування може бути можливим лише за умови строгого дозування вправ фізичного навантаження, чіткого виконання лікарських настанов та, власне, за активної участі пацієнта у процесі лікування.

Фахівці з фізичної реабілітації проводять обстеження своїх пацієнтів з метою виявити певні дисфункції та визначення рухового потенціалу. Також вони розробляють і виконують особистий план реабілітації, який можливий у співпраці з іншими колегами (лікарями, психологами, тренерами, тощо) та, безпосередньо, самим своїм пацієнтом.[2]

Фахівці фізичної реабілітації мають відповідну кваліфікацію щодо виконання професійних обов'язків:

- проведення повного комплексного обстеження або оцінки потреб свого певного пацієнта або потреб певної групи клієнтів, також встановлення чіткого діагнозу, визначення планів, заходів надання рекомендацій та консультації в рамках своєї компетенції та призначення того, коли саме пацієнтам відвідувати лікаря наступний раз, або ж рекомендацій звернутися до додаткових медичних фахівців
- впровадження планів та програми лікування, що була складена фахівцем в області фізичної реабілітації, прогноз щодо очікуваних

результатів, будь-якого втручання в процес лікування, надання рекомендацій

Фахівці з фізичної реабілітації мають широкий спектр можливостей для професійної діяльності, працюючи в різноманітних установах. Зокрема, вони можуть надавати свої послуги в медичних центрах, де зосереджені на реабілітації пацієнтів після травм або хвороб, а також у навчально-реабілітаційних закладах, які забезпечують допомогу дітям і дорослим із особливими потребами.

Їхній внесок важливий у закладах освіти, де фізична реабілітація сприяє інтеграції учнів із проблемами здоров'я. Крім того, фахівці працюють у закладах соціального захисту населення, забезпечуючи підтримку людям, які перебувають у складних життєвих обставинах, а також у спортивно-тренувальних комплексах, де допомагають спортсменам підтримувати оптимальну форму та відновлюватися після навантажень. Їхня участь цінується і в спортивних командах, де вони відіграють важливу роль у профілактиці травм та реабілітації гравців.

Згідно з чинним законодавством, спеціалісти з фізичної реабілітації можуть обіймати різні посади залежно від свого профілю та кваліфікації. Серед основних посад можна виділити такі, як «спеціаліст фізичної реабілітації», який забезпечує широкий спектр реабілітаційних послуг, «медична сестра (інструктор) з лікувальної фізкультури», що спеціалізується на впровадженні програм фізичних вправ для пацієнтів, та «медична сестра з масажу», яка надає послуги масажу як частину лікувального процесу.

Таким чином, фахівці цієї сфери мають змогу працювати в різноманітних середовищах, що дозволяє їм використовувати свої знання та навички для покращення фізичного стану різних категорій населення.[2]

1.3 Використання гаджетів

Сучасні технології стрімко розвиваються, постійно вдосконалюючись і відкриваючи нові можливості для людей у різних сферах життя. На сьогоднішній день існує широкий вибір інноваційних гаджетів, які стали невід'ємною частиною повсякденного життя та активно використовуються для вирішення різноманітних завдань. Від смарт-годинників, що стежать за фізичною активністю, до високотехнологічних пристроїв для домашньої автоматизації – сучасні технології впливають практично на всі аспекти нашого буття.

Особливу увагу заслуговує застосування таких пристроїв у медицині та сфері реабілітації. Інноваційні гаджети, розроблені спеціально для цих напрямків, сприяють значному спрощенню та оптимізації процесу лікування і відновлення здоров'я пацієнтів. Наприклад, смарт-браслети можуть моніторити пульс, рівень насичення крові киснем або рівень фізичної активності пацієнта, надаючи лікарям необхідну інформацію для оцінки стану хворого.

Інші пристрої, такі як віртуальні реабілітаційні системи, дозволяють пацієнтам виконувати спеціальні вправи у віртуальному середовищі під наглядом системи, яка оцінює правильність їх виконання. Такі інструменти значно полегшують контроль і корекцію реабілітаційного процесу, підвищуючи його ефективність. Крім того, використання гаджетів дозволяє скоротити час відновлення і підвищити комфорт пацієнта, роблячи процес лікування менш стресовим і більш технологічно адаптованим до сучасних реалій.

Отже, сучасні гаджети стають важливим помічником не лише в повсякденному житті, а й у медичній сфері, пропонуючи нові інструменти для підвищення якості лікування та реабілітації пацієнтів.

1.4 Фітнес-трекер

Фітнес-трекер – це багатофункціональний наручний гаджет, який значно перевершує можливості звичайного годинника. Він поєднує сучасний дизайн, практичність і широкий набір інструментів для моніторингу фізичної активності та стану здоров’я. Завдяки легкій вазі й ергономічній конструкції трекер комфортно носити щодня, навіть під час фізичної активності чи сну.

Цей пристрій стає незамінним помічником не лише у повсякденному житті, але й у спеціалізованих сферах, зокрема в медицині та реабілітації.

Однією з основних переваг фітнес-трекера є його здатність надавати користувачеві і спеціалісту точну інформацію про стан організму в режимі реального часу. На відміну від звичайного годинника, його функціонал включає безліч корисних можливостей, які роблять його справжнім технологічним партнером для тих, хто займається своїм здоров’ям. У процесі реабілітації фітнес-трекер стає ефективним інструментом, допомагаючи пацієнтам і лікарям досягати поставлених цілей.

Унікальні можливості фітнес-трекера для реабілітації:

1. Контроль фізичної активності. Трекер допомагає фіксувати кількість кроків і пройдену дистанцію за день, дозволяючи пацієнтам стежити за своєю активністю та дотримуватися рекомендацій лікаря. Ця функція особливо важлива для поступового збільшення фізичних навантажень, що є ключовим елементом реабілітаційного процесу.
2. Аналіз якості сну. Гаджет відстежує тривалість і якість сну, виявляє відхилення та навіть надає рекомендації для покращення режиму відпочинку. Це допомагає пацієнтам уникати перевтоми та сприяє загальному покращенню їхнього фізичного і психоемоційного стану.
3. Моніторинг фізіологічних показників. Трекер здатен вимірювати частоту серцевих скорочень, рівень активності м’язів і навіть насичення крові киснем. Ці дані особливо важливі для контролю стану пацієнта під час виконання фізичних вправ або у період відпочинку.

Переваги використання фітнес-трекерів у реабілітації:

- Оперативний обмін інформацією. Трекер дозволяє швидко передавати дані між пацієнтом і лікарем. Наприклад, лікар може отримати доступ до актуальних показників здоров'я пацієнта дистанційно, що значно спрощує контроль за дотриманням реабілітаційного плану.
- Миттєвий доступ до даних. У будь-який момент пацієнт або спеціаліст можуть зняти показники для аналізу. Це особливо важливо в умовах динамічного реабілітаційного процесу, де стан пацієнта може швидко змінюватися.
- Довгостроковий аналіз показників. Трекер накопичує дані за певний період часу, дозволяючи лікарю створювати детальні графіки змін стану здоров'я. Ця інформація дає можливість визначити тенденції у відновленні пацієнта, оцінити ефективність реабілітаційних заходів та внести необхідні корективи у план лікування.

Фітнес-трекер не лише сприяє точнішому моніторингу фізичного стану, а й допомагає пацієнтам відчувати більшу залученість у процес власного одужання. Його використання сприяє кращій комунікації між лікарем і пацієнтом, а також підвищує мотивацію до дотримання рекомендацій.

Завдяки таким технологічним помічникам реабілітаційний процес стає простішим, ефективнішим і комфортнішим для всіх учасників.

Висновки з розділу 1

У цьому розділі було детально проаналізовано принципи діяльності лікарів-реабілітологів, які спеціалізуються на відновленні здоров'я пацієнтів після фізичних травм. Важливим аспектом їхньої роботи є необхідність мати високу професійну кваліфікацію, яка дозволяє ефективно оцінювати стан пацієнта, розробляти індивідуальні програми реабілітації та супроводжувати їх реалізацію.

Окрім теоретичних та практичних знань, сучасні реабілітологи мають доступ до новітніх технологій, які роблять їхню роботу ще більш ефективною та зручною. Зокрема, було розглянуто приклад використання фітнес-трекерів – компактних наручних гаджетів, які стають важливим інструментом у реабілітаційному процесі. Такі пристрої забезпечують постійний моніторинг стану пацієнта, дозволяють фіксувати ключові фізіологічні показники та надавати лікарю необхідну інформацію в режимі реального часу.

Використання фітнес-трекерів сприяє не лише підвищенню точності та швидкості оцінки стану пацієнта, але й налагодженню більш тісного зв’язку між лікарем і пацієнтом. Пацієнти отримують можливість брати активну участь у процесі реабілітації, відстежуючи власний прогрес і дотримуючись рекомендацій лікаря. З іншого боку, лікарі можуть адаптувати реабілітаційні програми відповідно до отриманих даних, забезпечуючи індивідуальний підхід до кожного пацієнта.

Таким чином, поєднання професійних знань лікарів-реабілітологів із сучасними технологіями, такими як фітнес-трекери, значно розширює можливості реабілітаційної медицини, роблячи процес лікування більш комфортним, ефективним і орієнтованим на потреби пацієнтів.

РОЗДІЛ 2
АНАЛІТИЧНА ЧАСТИНА

2.1 Java

Java — це об’єктно-орієнтована мова програмування, яка була вперше представлена у 1995 році компанією Sun Microsystems. Її створення стало революційним кроком у сфері розробки програмного забезпечення, адже Java була розроблена з метою забезпечити незалежність програм від платформи. Це стало можливим завдяки концепції "Write Once, Run Anywhere" (WORA) — "Напиши один раз, виконуй де завгодно". Після придбання Sun Microsystems у 2010 році, розвиток Java перейшов до корпорації Oracle, яка продовжує активно підтримувати й вдосконалювати цю технологію.

Основний синтаксис Java ґрунтується на мовах C та C++, що робить її зрозумілою для програмістів, які знайомі з цими мовами. Проте Java суттєво спрощує розробку, оскільки має більш чисту об’єктно-орієнтовану модель і відсутність складних аспектів, властивих C++, таких як множинне успадкування чи необхідність ручного управління пам’яттю. Замість цього Java використовує автоматичне управління пам’яттю через механізм збирача сміття (Garbage Collection), що значно спрощує роботу з ресурсами програми.

Основні характеристики Java:

1. Портативність. Завдяки використанню байт-коду, програми на Java можуть виконуватися на будь-якому пристрої або операційній системі, де встановлена Java Virtual Machine (JVM). Ця особливість зробила Java надзвичайно популярною для розробки програм, які мають працювати на різних платформах, від серверів до мобільних пристроїв.
2. Безпека. Java забезпечує високий рівень безпеки завдяки вбудованим засобам перевірки коду, управління доступом та використанню пісочниці (sandbox) для виконання незнайомих або ненадійних програм.

3. Масштабованість і багатозадачність. Завдяки підтримці багатопоточності (multithreading), Java дозволяє розробляти продуктивні програми, які можуть виконувати декілька завдань одночасно, що особливо корисно для сучасних серверів і багатоядерних процесорів.
4. Розширюваність. Java має широку екосистему бібліотек і фреймворків, таких як Spring, Hibernate, Apache Maven, що значно полегшує розробку програм будь-якої складності.

Використання Java у різних сферах:

1. Мобільні додатки. Java є основною мовою для розробки програм під Android. Використання Java разом із Android SDK дозволяє створювати потужні й ефективні мобільні додатки.
2. Веб-розробка. Java широко використовується для створення динамічних веб-додатків через такі технології, як JavaServer Pages (JSP), Servlets, а також сучасні фреймворки на кшталт Spring і Struts.
3. Корпоративні системи. Завдяки стандарту Java Enterprise Edition (Java EE), ця мова стала вибором №1 для розробки складних бізнес-додатків, таких як системи управління ресурсами підприємства (ERP) чи управління взаємовідносинами з клієнтами (CRM).
4. Наука і дослідження. Java часто використовується для створення симуляцій, обробки великих даних і розробки програмного забезпечення для науки через бібліотеки типу Apache Hadoop або Weka.
5. Інтернет речей (IoT). Завдяки своїй універсальності, Java також застосовується для створення програмного забезпечення для розумних пристроїв, датчиків і контролерів.

Переваги Java для розробників:

- Величезна спільнота. Java має одну з найбільших і найактивніших спільнот розробників, що робить пошук допомоги чи рішень надзвичайно простим.

- Екосистема інструментів. Потужні інструменти для розробки, такі як IntelliJ IDEA, Eclipse, NetBeans, забезпечують ефективність роботи програмістів.
- Стабільність. Незважаючи на вік, Java залишається актуальною завдяки постійним оновленням і введенню нових можливостей, таких як модульність у Java 9 чи функціональне програмування у Java 8.

Завдяки своїй універсальності, стабільності та багатофункціональності, Java стала однією з найпоширеніших мов програмування у світі, яка активно використовується в різних галузях від фінансових систем до розважальних платформ. Її популярність пояснюється не лише технічними перевагами, але й здатністю адаптуватися до змін у світі технологій.

2.2 Переваги та недоліки

Java є однією з найпопулярніших мов програмування у світі завдяки своїм унікальним особливостям, які забезпечують їй відмінність серед інших технологій розробки. Розглянемо детально ключові характеристики та переваги цієї мови.

Особливості Java-технологій

1. Переносимість

Однією з найбільших переваг Java є її портативність. Програми, написані на цій мові, після одноразової трансляції у байт-код можуть виконуватися на будь-якій платформі, де встановлена віртуальна машина Java (JVM). Завдяки цьому розробникам не потрібно створювати окремі версії програми для різних операційних систем, що значно спрощує процес розробки.

Однак функціонування програм обмежується можливостями JVM. На відміну від «рідного» машинного коду, який може повністю використовувати апаратні ресурси комп’ютера, Java-програми працюють через інтерпретатор JVM, що накладає певні обмеження на доступ до фізичної пам’яті та іншого

апаратного забезпечення. Це може дещо знижувати продуктивність, хоча сучасні JVM вже значно оптимізовані для мінімізації цих недоліків.

2. Безпечність і контроль типів

Java спроектована таким чином, щоб мінімізувати можливість виникнення помилок. У мові відсутні механізми, які можуть спричинити нестабільність, наприклад, арифметика покажчиків чи неявні перетворення типів із втратою точності. Завдяки строгому контролю типів і обов’язковому обробленню винятків (exceptions), багато помилок можуть бути виявлені ще на етапі компіляції, що робить програми більш надійними.

Проте суворий контроль і додаткові перевірки можуть негативно впливати на продуктивність програм. Однак це компенсується підвищенням стабільності та передбачуваності роботи програмного забезпечення.

3. Автоматичне управління пам’яттю

У Java використовується механізм автоматичного звільнення пам’яті, відомий як збирач сміття (Garbage Collector). Цей підхід дозволяє розробникам уникати складностей, пов’язаних із ручним управлінням пам’яттю, що є однією з основних причин помилок у багатьох мовах, таких як C++.

Однак у випадках інтенсивної роботи з динамічною пам’яттю збирач сміття може не встигати очищувати невикористовувані ресурси, що іноді призводить до затримок у виконанні програм. Незважаючи на це, автоматичне управління пам’яттю робить Java простішою для освоєння й використання.

4. Використання стандартних бібліотек

Java пропонує багатий набір стандартних бібліотек, які значно спрощують процес розробки. Рішення для багатьох поширених завдань, таких як робота з файлами, мережами чи багатопоточністю, вже реалізовані у вигляді готових об'єктів. Це дозволяє зосередитися на реалізації бізнес-логіки замість вирішення рутинних технічних питань.

Проте для запуску програм на Java необхідно встановити середовище виконання Java (Java Runtime Environment, JRE), яке містить повний набір бібліотек. Якщо потрібна версія JRE відсутня, це може стати перешкодою для роботи програми.

5. Генерація документації

Java має вбудовані механізми для автоматичної генерації документації на основі коментарів у коді, такі як Javadoc. Ця функція полегшує розробникам і командам роботу над великими проєктами, забезпечуючи доступ до добре структурованої документації без додаткових зусиль.

6. Гнучкість у реалізації програм

Java дозволяє створювати програми, які можуть працювати в різних середовищах і з різними способами функціонування. Це робить мову універсальною, придатною для розробки як веб-додатків, так і мобільних, серверних або настільних програм.

Додаткові переваги Java

1. Концепції хорошого програмування

Java підтримує основні принципи об’єктно-орієнтованого програмування (ООП), такі як наслідування, поліморфізм і абстракція, що дозволяє створювати модульні та масштабовані програми. Ці концепції є базовими для багатьох інших мов програмування, тому знання Java полегшує освоєння інших технологій.

2. Стабільність і тривала підтримка

Java відома своєю стабільністю: нові версії мови додаються поступово, що дозволяє уникати різких змін. Розробникам не потрібно постійно адаптувати код до нових стандартів, як це буває в деяких інших мовах. Це особливо важливо для великих корпоративних систем, які потребують надійності й довготривалого функціонування.

3. Широка екосистема бібліотек і фреймворків

Java має багатий вибір бібліотек і фреймворків, які спрощують процес розробки. Наприклад, Spring і Hibernate широко використовуються для

створення веб-додатків і роботи з базами даних. Крім того, підтримка з боку великих компаній, таких як Google і Apache, забезпечує тривалу популярність і актуальність мови.

Отже, Java — це мова програмування, яка поєднує в собі простоту, універсальність і надійність. Її можливості роблять її ідеальним вибором для створення додатків будь-якої складності, а багатий набір бібліотек та інструментів забезпечує високу продуктивність роботи розробників.

Незважаючи на деякі обмеження, Java залишається одним із лідерів у світі програмування завдяки своїй портативності, безпечності й потужній спільноті розробників.[6]

2.3 Клас File

Клас File у Java є одним із ключових компонентів для роботи з файловою системою. Він забезпечує абстрактне представлення імен файлів та директорій, надаючи розробникам універсальний інструмент для роботи з шляхами у різних операційних системах. [1]

Особливості абстрактного шляху

Абстрактний шлях — це системно-незалежне представлення імені файлу або директорії. Воно складається з двох основних компонентів:

1. Системно-залежний префікс. Це може бути:
    - Ідентифікатор диску (наприклад, "C:" у Windows).
    - Символ кореневого каталогу "/" (у Unix-подібних системах).
    - UNC-префікс "\" для мережевих шляхів у Windows.
2. Послідовності імен. Це або пустий набір, або певна множина ідентифікаторів, що представляють директорії чи файли.

Перший елемент шляху (префікс) може бути ім’ям директорії або хосту (для UNC-шляхів). Наступні ідентифікатори вказують на вкладені директорії, а останній — на ім’я файлу чи директорії.

Системно-залежні аспекти шляху

Перетворення абстрактного шляху на рядок символів та навпаки залежить від операційної системи. Наприклад:

- Розділювач ідентифікаторів визначається системним полем file.separator.
- Клас File також надає доступ до розділювачів через статичні поля separator (рядок) і separatorChar (символ).

При перетворенні рядка на абстрактний шлях використовуються стандартні символи-розділювачі (або підтримувані символи, специфічні для файлової системи). Це дозволяє забезпечити коректну роботу з файлами та директоріями на будь-якій платформі.

Абсолютні та відносні шляхи

Шляхи можуть бути:

1. Абсолютними. Вони повністю визначають місцезнаходження файлу чи директорії в файловій системі. Наприклад, у Windows: C:\Users\Documents\file.txt, у Unix: /usr/local/bin.
2. Відносними. Ці шляхи залежать від поточного місцезнаходження (робочої директорії).

Поточна робоча директорія за замовчуванням визначається системним полем user.dir. Вона відповідає тій директорії, з якої була запущена віртуальна машина Java (JVM). Наприклад, якщо JVM була запущена в /home/user, то шлях documents/file.txt буде інтерпретовано як /home/user/documents/file.txt. [1]

Робота з батьківськими елементами шляхів

Для отримання батьківського елемента (предка) абстрактного шляху використовується метод getParent() класу File. Він повертає префікс і всі ідентифікатори шляху, окрім останнього. Наприклад:

- Для шляху /usr/local/bin батьківським елементом буде /usr/local.
- Для файлу /home/user/file.txt метод поверне /home/user.

Абсолютний шлях до директорії завжди є предком для будь-якого об’єкта File, шлях якого починається з цього абсолютного шляху. Наприклад, директорія /usr є предком для /usr/local/bin.

Переваги використання класу File

1. Абстракція: Клас File дозволяє працювати з файлами та директоріями незалежно від операційної системи. Це спрощує перенесення програм між платформами.
2. Гнучкість: Підтримка як абсолютних, так і відносних шляхів дозволяє ефективно організовувати роботу з файлами в різних контекстах.
3. Інтеграція: File легко інтегрується з іншими класами пакету java.io, такими як BufferedReader чи FileWriter, що робить його зручним для обробки файлів.

Отже, клас File у Java є потужним інструментом для роботи з файлами та директоріями. Він забезпечує абстрактну модель шляху, підтримує роботу зі стандартними та системно-залежними розділювачами, а також дозволяє зручно маніпулювати файлами як на рівні операційної системи, так і у програмному коді. Завдяки своїм можливостям, File є незамінним у розробці кросплатформених додатків. [1]

2.4 My SQL

MySQL — одна з найпопулярніших систем управління базами даних (СУБД), яка широко використовується у всьому світі. Вона забезпечує ефективне управління даними, підтримуючи різні операційні системи, і завдяки своїй функціональності стала вибором номер один для багатьох програмних проектів. Разом із MySQL існують й інші СУБД, такі як Oracle, MSSQL, та PostgreSQL, але MySQL часто виділяється своєю безкоштовною ліцензією та крос-платформністю.[3]

Історія та розвиток MySQL

MySQL була створена шведською компанією MySQL AB. Основні етапи її розробки та еволюції:

- Початок розробки припав на 1994 рік, а перший реліз відбувся 23 травня 1995 року.
- Першу версію для Windows випустили 8 січня 1998 року, що зробило платформу доступною для більш широкої аудиторії.
- Далі MySQL активно вдосконалювалася, виходили нові версії:
  - 3.23 у 2001 р.
  - 4.0 у 2003 р.
  - 4.1 у 2004 р.
  - 5.0 у 2005 р.

У 2008 році компанія Sun Microsystems викупила MySQL AB. Згодом, у 2010 році, після придбання Sun Microsystems корпорацією Oracle, MySQL перейшла під її управління, ставши важливою складовою IT-екосистеми Oracle.

Популярність та використання MySQL

MySQL стала особливо популярною серед безкоштовних програмних проектів, де потрібна потужна СУБД з можливостями пошуку. Вона використовується на серверах як під управлінням Windows, так і Linux, що робить її універсальною платформою для розробки веб-додатків, CRM-систем, аналітичних платформ та інших додатків.

Станом на квітень 2009 року MySQL пропонувала два основних варіанти своєї СУБД:

1. MySQL Community Server — версія з відкритим вихідним кодом.
2. MySQL Enterprise Server — комерційна версія з додатковими інструментами підтримки та оптимізації.

Основні можливості MySQL

MySQL надає розробникам та адміністраторам систем низку важливих функцій:

1. Простота встановлення та використання. Встановлення MySQL є інтуїтивно зрозумілим процесом, що не вимагає глибоких технічних знань.
2. Підтримка багатокористувацького доступу. Одночасно з базою даних можуть працювати необмежена кількість користувачів.
3. Обробка великих обсягів даних. Таблиці можуть містити до 50 мільйонів рядків, що робить MySQL потужним інструментом для роботи з великими обсягами інформації.
4. Висока продуктивність. Швидкість обробки запитів є однією з ключових переваг MySQL.
5. Безпека.
   Ефективна система авторизації та підтримка шифрування SSL забезпечують високий рівень захисту даних.
6. Крос-платформність.
   MySQL підтримує основні операційні системи, включаючи Windows, Linux, macOS та інші.

Додаткові функції MySQL

MySQL включає у себе великий набір функцій, які значно спрощують розробку та адміністрування баз даних:

- Збережені процедури та функції, що дозволяють виконувати складні операції безпосередньо в базі даних.
- Тригери для автоматизації реакції на події в базі даних.
- Курсори, які дають можливість обробляти рядки результатів по одному.
- Оновлювані представлення, які дозволяють створювати логічні подання даних.
- Реплікація для синхронізації даних між декількома серверами.
- Повноцінна підтримка Unicode (UTF-8 і UCS2), що дозволяє працювати з багатомовними додатками.

- Кешування запитів для оптимізації продуктивності.
- Сегментування таблиць для підвищення швидкості роботи з великими наборами даних.[7]

Чому обирають MySQL?

1. Безкоштовна ліцензія. Відкритий вихідний код робить MySQL доступним для невеликих команд та великих організацій.
2. Широкий вибір інструментів. MySQL має великий набір утиліт для управління базами даних, як-то Workbench.
3. Підтримка спільноти. Велика база користувачів і спільнот дозволяє швидко знайти рішення для більшості проблем.
4. Стабільність. MySQL є перевіреним інструментом, який використовують такі гіганти, як Google, Facebook та Twitter.

Отже, MySQL — це надійна, потужна та функціональна система управління базами даних, яка підходить як для початківців, так і для досвідчених розробників. Її простота у використанні, розширені можливості та активна підтримка роблять її ідеальним вибором для сучасних додатків.

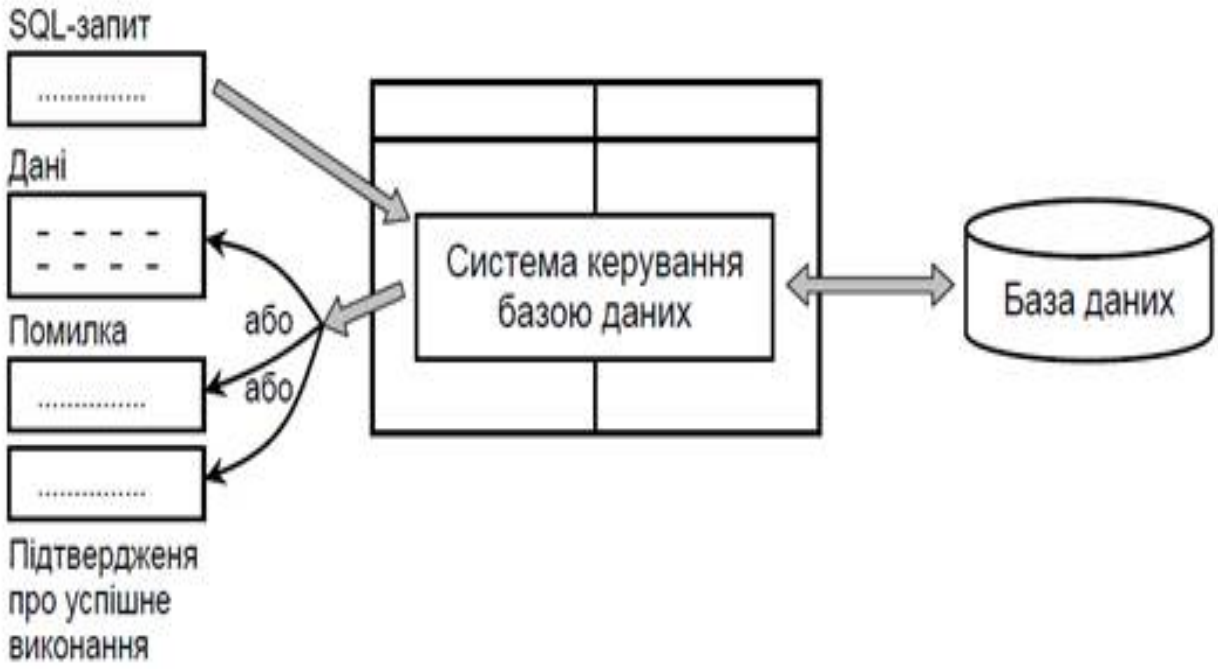


Рис. 2.1 – Використання SQL для роботи з базою даних

Мова SQL (Structured Query Language) — це стандартна мова для взаємодії з реляційними базами даних, яка забезпечує створення, зміну, маніпулювання даними та управління доступом до бази. Вона виникла на основі реляційної моделі даних, розробленої Едгаром Коддом у 1970-х роках, і була вперше реалізована IBM як SEQUEL (Structured English Query Language). Надалі назва трансформувалася в SQL, хоча історично абревіатура може читатися як «сіквел», правильним вважається варіант «ескюель».

Основні етапи розвитку SQL

1. Початки: SQL вперше реалізовано в реляційній СКБД DB2 від IBM. На відміну від традиційних процедурних мов програмування (наприклад, COBOL або C), SQL є непроцедурною мовою. Це означає, що вона описує результат, який потрібно отримати, без деталізації алгоритму його досягнення.
2. Стандартизація:
    - У 1986 році Американський національний інститут стандартів (ANSI) затвердив перший стандарт SQL.
    - У 1989 році ISO підтвердила його.
    - Надалі вийшли оновлення стандарту: SQL92 (SQL2), SQL99 (SQL3), SQL2003, з наступними модифікаціями у 2006 та 2008 роках.
3. Популяризація: IBM розробила теорію реляційних баз даних, але першою з цією технологією на ринок вийшла компанія Oracle. SQL активно впроваджувався у більшість реляційних СКБД, які отримали власні діалекти.

Основні можливості SQL

SQL використовується для:

- Опису структури баз даних (створення таблиць, змінення або видалення їх).
- Маніпулювання даними: вставка, оновлення, видалення та вибірка інформації.

- Упорядкування та фільтрація даних: пошук інформації у кількох таблицях з можливістю впорядкування результатів.
- Підтримки цілісності: опис процедур забезпечення узгодженості даних.
- Контролю доступу: встановлення прав доступу до даних.

Основні діалекти SQL

Через потребу в додаткових можливостях багато виробників СКБД розробили власні діалекти SQL:

1. PL/SQL — для Oracle, додає процедурні конструкції (IF, цикли).
2. T-SQL (Transact-SQL) — для Microsoft SQL Server, розширений набір функцій для обробки даних.
3. PL/pgSQL — для PostgreSQL, подібний до Oracle PL/SQL.
4. SQL/PSM — у MySQL для реалізації процедур і функцій.
5. SQL PL — у IBM DB2.

Структура SQL

SQL поділяється на три основні групи операторів:

1. DDL (Data Definition Language) — визначення структури бази даних:
    - CREATE — створення об'єктів (таблиць, баз даних).
    - ALTER — змінення існуючих об'єктів.
    - DROP — видалення об'єктів.
2. DML (Data Manipulation Language) — маніпулювання даними:
    - SELECT — вибірка даних.
    - INSERT — вставка нових рядків.
    - UPDATE — оновлення існуючих даних.
    - DELETE — видалення рядків.
3. DCL (Data Control Language) — управління доступом:
    - GRANT — надання доступу до ресурсів.
    - DENY — заборона доступу.
    - REVOKE — скасування раніше наданих дозволів.

Приклад оператора SELECT

Оператор SELECT використовується для вибірки даних із бази:

```
sql
SELECT [DISTINCT]
    < * | перелік стовпців | функції >
FROM   < перелік таблиць >
[WHERE < умова відбору >]
[GROUP BY < стовпець > [HAVING < умова >]]
[ORDER BY < стовпець | порядковий номер >];
```

- DISTINCT — виключає дублікати з результатів.
- FROM — вказує таблиці, з яких беруться дані.
- WHERE — задає умови фільтрації.
- GROUP BY — групує результати за заданим стовпцем.
- HAVING — фільтрує результати групування.
- ORDER BY — впорядковує результати.

Приклад: вибір усіх працівників із зарплатою більше 1000, відсортованих за іменем:

```
sql
SELECT name, salary
FROM employees
WHERE salary > 1000
ORDER BY name;
```

Отже, SQL є універсальною мовою для роботи з реляційними базами даних. Незважаючи на існування різних діалектів, її стандартна структура забезпечує зрозумілість і спільність для більшості сучасних СКБД.

2.5 Використання засобу Java FX

JavaFX: Сучасна платформа для створення графічних інтерфейсів

JavaFX — це платформа для розробки насичених інтернет-застосунків (RIA, Rich Internet Applications), що забезпечує потужний набір інструментів для створення графічних інтерфейсів. Розроблена як сучасна альтернатива застарілим бібліотекам, таким як Swing, JavaFX пропонує широкий функціонал для реалізації складних інтерфейсів із підтримкою стилів, 3D-графіки та інтерактивності.[9]

Історія розвитку JavaFX

1. JavaFX.1.0(2008):
   Перша версія використовувала скриптову мову JavaFX Script для опису графічного інтерфейсу.
2. JavaFX.2.0(2011):
   Випущена під егідою Oracle, відмовилася від окремої мови скриптів на користь Java.
3. JavaFX 8 (2014):
   Стала частиною екосистеми Java 8, запропонувала підтримку 3D-графіки, нових візуальних компонентів і стилів CSS. Версії 3–7 були пропущені для синхронізації з нумерацією Java.

Ключові особливості JavaFX

1. Підтримка патерну MVC
   Завдяки архітектурі Model-View-Controller, JavaFX розділяє дані, інтерфейс і логіку програми.
2. Декларативний опис інтерфейсів
   Мова FXML дозволяє описувати візуальні компоненти в XML-форматі.
3. Стилізація через CSS
   Елементи інтерфейсу стилізуються за допомогою CSS, що полегшує дизайн і зміну вигляду програми.

4. Розширені можливості взаємодії
   Підтримка жестів, анімацій та інших інтерактивних елементів.
5. 3D-графіка
   Включає інструменти для створення 2D/3D-об’єктів і їх інтеграції в програми.
6. Сумісність із Swing
   JavaFX побудований поверх Swing, що дозволяє використовувати старі компоненти в нових проектах.
7. Широкий мультимедійний функціонал
   Підтримка популярних форматів зображень і аудіо (наприклад, MP3, AIFF).

Чому обирають JavaFX?

1. Інтерактивність
   У світі, де веб-додатки повинні бути привабливими та функціональними, JavaFX забезпечує розробку динамічних програм із багатим графічним оформленням.
2. Гнучкість та інтеграція
   Використання JavaFX дозволяє створювати інтерактивні додатки, які можуть працювати у браузерах або як окремі програми, а також інтегруватися з існуючими бібліотеками.
3. Сумісність із сучасними браузерами
   JavaFX відповідає зростаючим вимогам до продуктивності та функціональності в умовах ускладнення JavaScript і HTML.
4. Універсальність
   Завдяки можливості використання XML, JavaFX полегшує розділення структури даних і коду програми.

Пакети та можливості JavaFX

1. Геометрія:
   Забезпечує створення 2D/3D-об’єктів.

2. CSS:

   Надає класи для стилізації компонентів.

3. Події:

   Містить класи для обробки різноманітних подій (наприклад, натискання кнопок).

4. Анімації:

   Підтримка вбудованих анімацій для додавання динамічних елементів в інтерфейси.

Приклади можливостей JavaFX

1. Побудова 2D/3D-графіки

   Інтеграція тривимірних об’єктів для візуалізації складних даних.

2. Мультимедіа

   Програми можуть програвати аудіо та відео без необхідності сторонніх бібліотек.

3. Змішування з Swing

   Можливість використання старих компонентів разом із новими технологіями.

4. Простота розгортання

   JavaFX значно спрощує розгортання додатків завдяки вбудованим інструментам.[9]

Отже, JavaFX — це потужний інструмент для розробки графічних інтерфейсів, який поступово витісняє застарілі бібліотеки завдяки сучасному функціоналу, гнучкості та підтримці багатих візуальних компонентів. Він ідеально підходить для створення інтерактивних додатків у середовищі Java.

2.6 Java DB

JDBC: Уніфікований спосіб роботи з базами даних у Java

JDBC (Java Database Connectivity) — це набір інтерфейсів і класів, які забезпечують універсальний спосіб взаємодії між додатками на Java та

базами даних. Основна ідея JDBC полягає у створенні стандартизованого API, який дозволяє писати програмний код незалежно від типу бази даних.

Принцип роботи JDBC

1. Універсальність:

   Код Java не залежить від типу бази даних. Це означає, що робота з Oracle або PostgreSQL виконується за однаковою схемою, за винятком SQL-запитів, які можуть відрізнятися через особливості конкретної СУБД.

2. Єдина архітектура:

   Процес надсилання SQL-запитів та отримання результатів виконується через єдиний набір команд, незалежно від специфіки бази даних.

Основні компоненти JDBC

JDBC API версії 4.0 включає два головні пакети:

1. java.sql

   Забезпечує базові класи та інтерфейси для роботи з базами даних, такі як створення з'єднання, виконання SQL-запитів і обробка результатів.

2. javax.sql

   Розширює функціонал пакету java.sql, пропонуючи додаткові інструменти для роботи з джерелами даних (наприклад, пул з'єднань).

Особливості пакету java.sql

Пакет java.sql є ядром роботи з JDBC і забезпечує такі функції:

- З'єднання з базою даних через об'єкт Connection.
- Створення SQL-запитів за допомогою Statement, PreparedStatement або CallableStatement.
- Обробка результатів через об'єкт ResultSet.
- Обробка транзакцій (команди commit і rollback).
- Обробка помилок через клас SQLException.

Приклад коду JDBC

java

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class JdbcExample {
    public static void main(String[] args) {
        // URL, користувач і пароль до бази даних
        String url = "jdbc:postgresql://localhost:5432/mydatabase";
        String user = "username";
        String password = "password";

        try {
            // Створення з'єднання
            Connection connection = DriverManager.getConnection(url, user, password);

            // Створення SQL-запиту
            Statement statement = connection.createStatement();
            String sql = "SELECT id, name FROM users";
            ResultSet resultSet = statement.executeQuery(sql);

            // Виведення результатів
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                System.out.println("ID: " + id + ", Name: " + name);
            }

            // Закриття ресурсів
            resultSet.close();
```

```
            statement.close();
            connection.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Переваги JDBC

1. Універсальність: Один API для всіх СУБД.
2. Гнучкість: Можливість роботи з будь-якою базою даних за наявності драйвера.
3. Підтримка транзакцій: Контроль над атомарністю операцій.
4. Інтеграція з Java EE: Полегшує створення корпоративних додатків.

JDBC залишається основою для роботи з базами даних у Java, забезпечуючи стандартизований та надійний спосіб взаємодії між програмним забезпеченням і СУБД.

Класи / Інтерфейси	Опис
BLOB	Він представляє значення SQL Blob у програмі Java
CallableStatement	Він використовується для виконання збережених процедур SQL
КЛОБ	Він представляє значення SQL Clob у програмі Java
Підключення	Він створює зв’язок (сеанс) із певною базою даних
Дата	Він забезпечує підтримку типу SQL Date

Водій	Він створює екземпляр драйвера за допомогою Driver Manager
DriverManager	Він надає базову послугу для управління набором драйверів JDBC
ParameterMetaData	Це об’єкт, за допомогою якого можна отримати інформацію про типи та властивості кожного параметра в об’єкті PreparedStatement
ПідготовленоЗаява	Він використовується для створення та виконання параметризованого запиту в програмі Java
ResultSet	Він використовується для доступу до результату по рядках
ResultSetMetaData	Він використовується для отримання інформації про типи та властивості стовпців у об’єкті ResultSet
Ідентифікатор рядка	Він представляє значення SQL ROWID
Savepoint	Він представляє точку збереження в транзакції
SQLData	Він використовується для зіставлення визначеного користувачем типу SQL (UDT) класу в програмі Java
SQLXML	Він представляє тип SQL XML
Заява	Він використовується для виконання статичного оператора SQL

DriverPropertyInfo	Він надає властивості драйвера для встановлення з'єднання
SQLException	Він надає інформацію про помилки бази даних
SQLTimeoutException	Це підклас SQLException, викинутий після закінчення часу очікування, зазначеного в інструкції
Попередження про SQL	Це виняток, який надає інформацію про попередження про доступ до бази даних
Структура	Це стандартне відображення в програмі Java для структурованого типу SQL

(ii) пакет javax.sql

Це API розширення JDBC, дає забезпечення доступу та оброблення даних на стороні сервера в програмі Java.

Класи / Інтерфейси	Опис
CommonDataSource	Це інтерфейс, який визначає методи, загальні між DataSource, XADataSource і ConnectionPoolDataSource
ConnectionPoolDataSource	Це фабрика об’єктів PooledConnection
Джерело даних	Це фабрика для підключень до фізичного DataSource, який представляє об'єкт
PooledConnection	Він використовується для управління пулом

	з'єднань
RowSet	Він надає підтримку JDBC API для Java-компонентної моделі компонентної моделі
RowSetMetadata	Він містить інформацію про стовпці в об'єкті RowSet
ConnectionEvent	Він надає інформацію про виникнення подій, пов’язаних із підключенням
ConnectionEventListener	Він використовується для реєстрації подій об’єкта PooledConnection
RowSetEvent	Він генерується, коли подія відбувається з об’єктом Rowset
StatementEvent	Він надсилається всім StatementEventListeners, які були зареєстровані згенерованим PooledConnection

Табл.1 – клас/інтерфейс та опис

2) Драйвер завантаження

Драйвер можна завантажити одним із двох способів:

1. Class.forName ()
2. DriverManager.registerDriver ()

(i) Class.forName ()

Отже, файл класа драйвера вантажиться до пам’яті саме під час виконання, що не дуже явно завантажує сам драйвер. Сам драйвер за замовчуванням автоматом реєструється в JDBC.

Таким чином, файл класу драйвера завантажується в пам’ять під час виконання. Це неявно завантажує драйвер. Під час завантаження драйвер автоматично реєструється в JDBC.

Ім'я БД	Ім'я драйвера JDBC
MySQL	com.mysql.jdbc.Driver
Oracle	oracle.jdbc.driver.OracleDriver
Microsoft SQL Server	com.microsoft.sqlserver.jdbc.SQLServerDriver
MS Access	net.ucanaccess.jdbc.UcanaccessDriver
PostgreSQL	org.postgresql.Driver
IBM DB2	com.ibm.db2.jdbc.net.DB2Driver
Sybase	com.sybase.jdbcSybDriver
TeraData	com.teradata.jdbc.TeraDriver

Табл. 2.2 – приклади

Робота з DriverManager у JDBC

DriverManager — це клас із пакету java.sql, який виконує роль посередника між програмою на Java та базою даних. Основні завдання DriverManager:

1. Реєстрація драйвера бази даних.
2. Створення з’єднання з базою даних.

Реєстрація драйвера

Перед створенням з'єднання потрібно зареєструвати драйвер бази даних. Це можна зробити двома основними способами:

1. Використання методу Class.forName().
   Цей метод завантажує клас драйвера у JVM. Однак він працює тільки для віртуальних машин, сумісних із JDK.
   java
   Class.forName("oracle.jdbc.driver.OracleDriver");
2. Метод DriverManager.registerDriver().
   Цей метод явно реєструє драйвер у DriverManager, передаючи об'єкт драйвера як параметр.
   java
   DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
   DriverManager.registerDriver(new
   com.microsoft.sqlserver.jdbc.SQLServerDriver());
   Особливості:
   - Викидає SQLException, якщо виникає помилка бази даних.
   - Викидає NullPointerException, якщо переданий драйвер дорівнює null.

   Приклад:
   java
   import java.sql.DriverManager;
   import java.sql.SQLException;

   public class DriverRegistration {
      public static void main(String[] args) {
         try {
            DriverManager.registerDriver(new
   oracle.jdbc.driver.OracleDriver());
            System.out.println("Драйвер зареєстровано успішно!");

```
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Створення з'єднання

Після завантаження і реєстрації драйвера наступний крок — створення з’єднання з базою даних. Для цього використовується метод getConnection() класу DriverManager.

Методи getConnection()

1. getConnection(String url, String user, String password) Приймає три параметри:
   - url — адреса бази даних.
   - user — ім'я користувача для доступу до бази.
   - password — пароль користувача.

   Приклад:

   java

   Копировать код

   ```
   Connection connection = DriverManager.getConnection(
       "jdbc:postgresql://localhost:5432/mydatabase",
       "username",
       "password"
   );
   ```
2. getConnection(String url) Використовується, якщо URL включає ім’я користувача та пароль.

   Приклад URL:

   java

   ```
   jdbc:mysql://localhost:3306/mydatabase?user=username&password=password
   ```

   Код:

java
Connection connection = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/mydatabase?user=username&password=passw
ord"
);
Приклад коду: Реєстрація драйвера та створення з'єднання

```
java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/mydatabase";
        String user = "username";
        String password = "password";

        try {
            // Реєстрація драйвера (необов'язкова для нових драйверів)
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Встановлення з'єднання
            Connection connection = DriverManager.getConnection(url, user,
password);
            System.out.println("З'єднання встановлено!");

            // Закриття з'єднання
            connection.close();
            System.out.println("З'єднання закрито.");
```

```
        } catch (ClassNotFoundException e) {
            System.out.println("Драйвер не знайдено!");
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("Помилка при роботі з базою даних!");
            e.printStackTrace();
        }
    }
}
```

Підсумок:

1. Метод Class.forName() використовується для завантаження драйвера, але він більше не є обов’язковим у сучасних версіях JDBC (починаючи з JDK 6).
2. DriverManager.registerDriver() дозволяє вручну зареєструвати драйвер.
3. Метод DriverManager.getConnection() забезпечує створення з’єднання з базою даних.
4. Під час використання DriverManager рекомендується коректно закривати з'єднання для уникнення витоку ресурсів.

У наступній таблиці перелічені рядки з'єднання JDBC для різних баз даних:

База даних	Рядок підключення / URL-адреса БД
MySQL	jdbc: mysql: // HOST_NAME: PORT / DATABASE_NAME
Oracle	jdbc: oracle: тонкий: @HOST_NAME: PORT: SERVICE_NAME
Microsoft SQL	jdbc: sqlserver: // HOST_NAME: PORT; Ім'я бази даних =

Server	
MS Access	jdbc: ucanaccess: // DATABASE_PATH
PostgreSQL	jdbc: postgresql: // HOST_NAME: PORT / DATABASE_NAME
IBM DB2	jdbc: db2: // ІМЯ ХОСТА: ПОРТ / БАЗА ДАНИХ
Sybase	jdbc: Sybase: Tds: HOSTNAME: PORT / DATABASE_NAME
TeraData	jdbc: teradata: // HOSTNAME / database =, tmode = ANSI, charset = UTF8

Висновки до розділу 2.

У цьому розділі були проаналізовані та описані методи й техніки, що застосовуються для практичної реалізації роботи з базами даних у середовищі Java. Особливу увагу приділено основним перевагам мови програмування Java, які забезпечують її популярність і широке використання в поєднанні з SQL-базами даних.

Огляд включав розбір базових механізмів взаємодії Java із базами даних за допомогою JDBC (Java Database Connectivity), а також підходів до створення, реєстрації драйверів і встановлення з'єднань із базами даних. Було показано, як ці прийоми забезпечують універсальність і простоту роботи з різними реляційними системами керування базами даних (Oracle, PostgreSQL, MySQL тощо).

Крім того, висвітлені ключові переваги Java, такі як незалежність від платформи, наявність багатого API для роботи з базами даних, а також підтримка сучасних підходів до програмування, що забезпечують ефективність і гнучкість у розробці застосунків.

РОЗДІЛ 3
ПРАКТИЧНА ЧАСТИНА

3.1 Характеристика початкових даних

Реабілітаційний період і відновлення здоров'я пацієнта мають індивідуальний характер, оскільки залежать від специфіки травми, її тяжкості та особливостей організму кожного пацієнта. Це створює необхідність адаптації процесів реабілітації для кожного випадку, що унеможливлює застосування єдиного шаблону для всіх. З цієї причини важливо забезпечити лікаря додатковими інструментами, такими як спеціальні поля для внесення індивідуальних коментарів, рекомендацій, прогнозів і зауважень. Це дозволяє підвищити ефективність програмного забезпечення, роблячи його зручним для роботи з пацієнтами з унікальними потребами.

3.2 Етапи реабілітації.

Реабылытацыю кожного пацієнта діляться на етапи так, як зображено на рис.3.1.

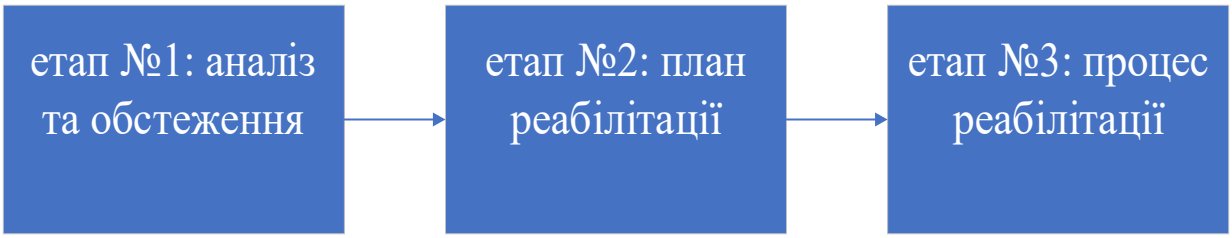


Рисунок 3.1 – етапи реабілітації

Етап 1: Аналіз та обстеження

На першому етапі лікар проводить аналіз історії хвороби пацієнта, використовуючи наявні дані та медичні записи. Потім, відповідно до

призначення, здійснюється обстеження для отримання актуальної інформації про стан здоров’я пацієнта. Цей процес дозволяє виявити можливі помилки в попередніх висновках і отримати точні дані для подальшого планування реабілітації.

Етап 2: Планування реабілітації

На основі результатів обстеження лікар формує детальний опис травми, встановлює діагноз і розробляє план реабілітації. До цього етапу входить визначення графіка наступних прийомів, складання рекомендацій і призначення процедур, які сприятимуть відновленню здоров’я пацієнта.

Етап 3: Процес реабілітації

Цей етап передбачає активну роботу лікаря-реабілітолога, який слідкує за відновленням здоров’я пацієнта, фіксує зміни в його стані та оновлює план лікування. Лікар має можливість коригувати початкові рекомендації відповідно до прогресу пацієнта, забезпечуючи індивідуальний підхід на кожному етапі реабілітації.(табл.3.1.).

Таблиця 3.1.

Дії лікаря та користь на етапах реабілітації

Етап	Дії	Користь
Аналіз та обстеження	Внесення даних, детальна робота з історією пацієнта та його інформацією	Внесення актуальних даних стосовно пацієнта та його здоров’я
План реабілітації	Чіткий опис прийому, коментарі відносно прийому, занесення загальної інформації	Зручність передачі даних, чіткість та грамотна система роботи

Процес реабілітації	Якісний аналіз прогресу/регресу, можливість аналізу задля зміни даних	Комфортна і грамотна робота як і з даними з минулого пацієнта, так і з актуальними даними

3.3 Алгоритми системи

Алгоритм взаємодії під час прийому міє лікарем та його пацієнтом детально зображено на рис.3.2.

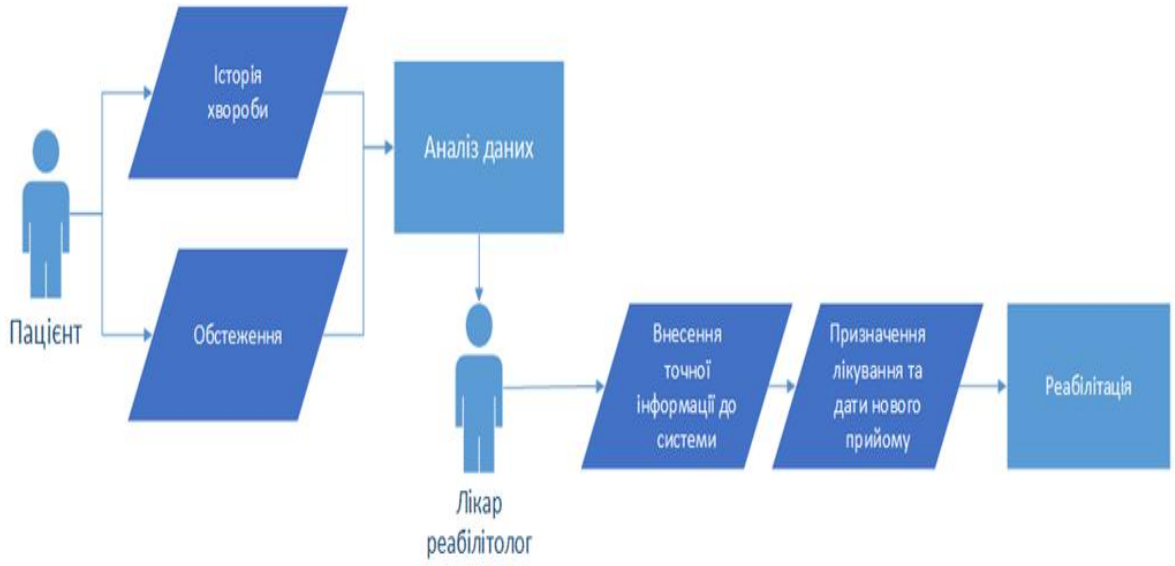


Рисунок 3. 2 – алгоритм роботи системи

Алгоритм роботи системи починається із взаємодії пацієнта з лікарем. Цей процес включає кілька етапів, які забезпечують комплексний підхід до реабілітації, починаючи від аналізу даних пацієнта до активного моніторингу його прогресу.

Опис алгоритму роботи системи

Етапи взаємодії пацієнта – прийом лікаря, комплексне обстеження та виконання рекомендацій лікаря по відновленню.

1. Прийом у лікаря: пацієнт приходить до лікаря з історією хвороби. Ця інформація є основою для оцінки попереднього лікування та початкового стану пацієнта.
2. Комплексне обстеження: за рекомендацією лікаря пацієнт проходить обстеження, яке дозволяє отримати актуальні дані про стан його здоров'я. Це включає як фізіологічні показники (серцебиття, тиск тощо), так і спеціалізовані тести залежно від характеру травми.
3. Виконання рекомендацій лікаря: після консультації пацієнт отримує план реабілітації, рекомендації та, за необхідності, інструкції з використання реабілітаційних пристроїв чи гаджетів. Пацієнт зобов'язаний дотримуватись цих рекомендацій для досягнення позитивних результатів.

Етапи роботи лікаря

1. Аналіз даних: лікар проводить детальний аналіз отриманих результатів обстеження та інформації з історії хвороби. Він зіставляє ці дані, виявляє можливі неточності, коригує їх та формує достовірну картину стану здоров'я пацієнта.
2. Розробка плану реабілітації: на основі отриманих даних лікар створює індивідуальний план реабілітації. План включає прогноз, рекомендації щодо процедур і фізичних навантажень, а також графік контрольних візитів.
3. Моніторинг прогрессу: після кожного прийому лікар оцінює прогрес пацієнта, вносить зміни до плану реабілітації за необхідності та фіксує результати в системі. Це дозволяє адаптувати реабілітацію до поточного стану пацієнта.

Реєстрація даних у системі

Під час роботи лікар заносить до системи такі дані:

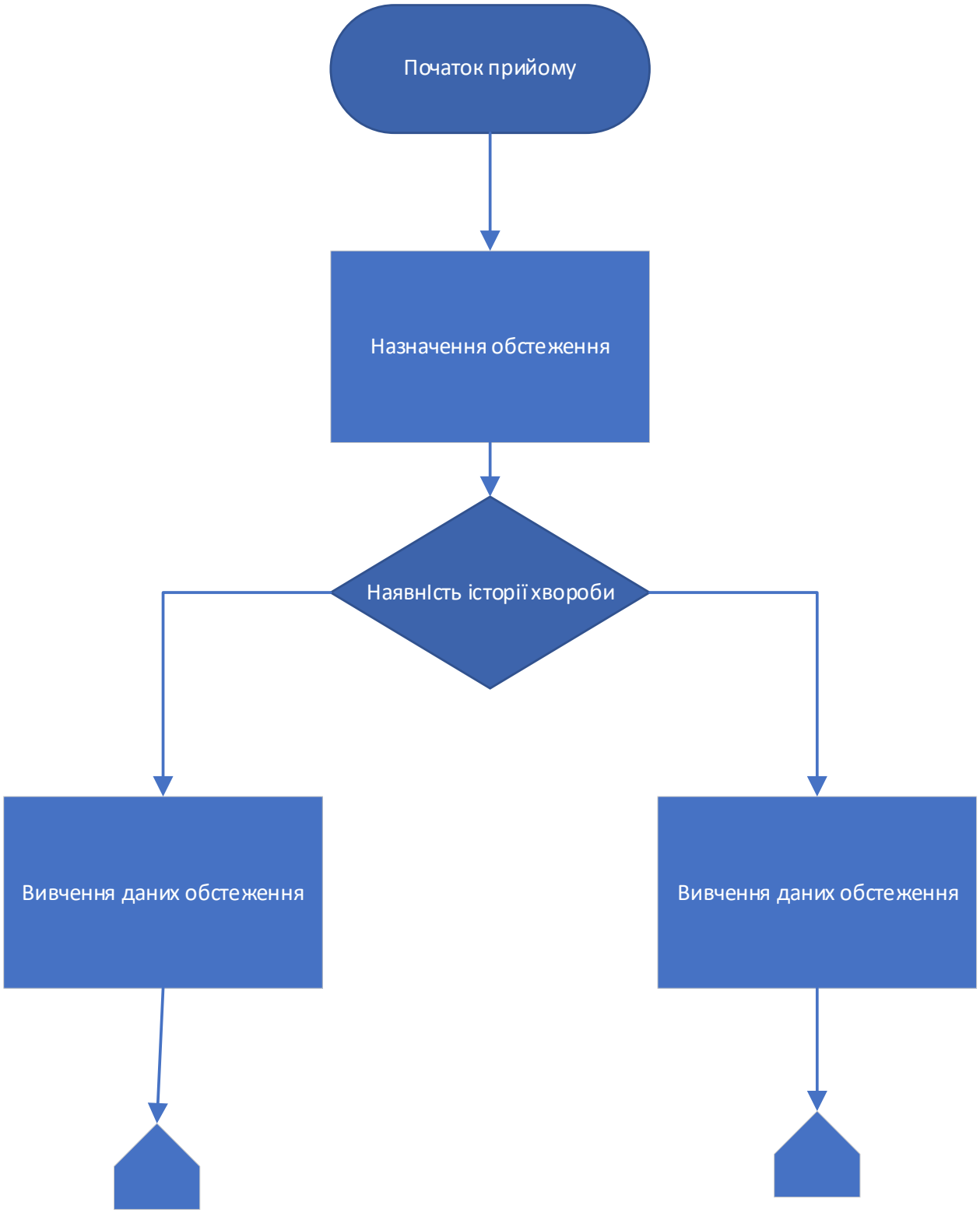
- Особисті контактні дані пацієнта.
- Фізіологічні показники: частота серцевих скорочень, артеріальний тиск тощо.

- Опис травми: її причини, тяжкість і деталі.
- Рекомендації та прогноз: план відновлення, додаткові зауваження лікаря.

Ця інформація дозволяє системі забезпечувати актуальний і зручний інструментарій для роботи лікаря з кожним пацієнтом індивідуально.

Отже, система забезпечує комплексний процес реабілітації, орієнтований на пацієнта. Завдяки ретельному збору даних, індивідуальному підходу та постійному моніторингу, пацієнти отримують ефективну допомогу, яка відповідає їхнім потребам і стану здоров’я.

Прийом, який відбувається з сторони лікаря безпосередньо показаний блок-схемою на рис. 3.3


Початок прийому
Назначення обстеження
Наявність історії хвороби
Вивчення даних обстеження
Вивчення даних обстеження

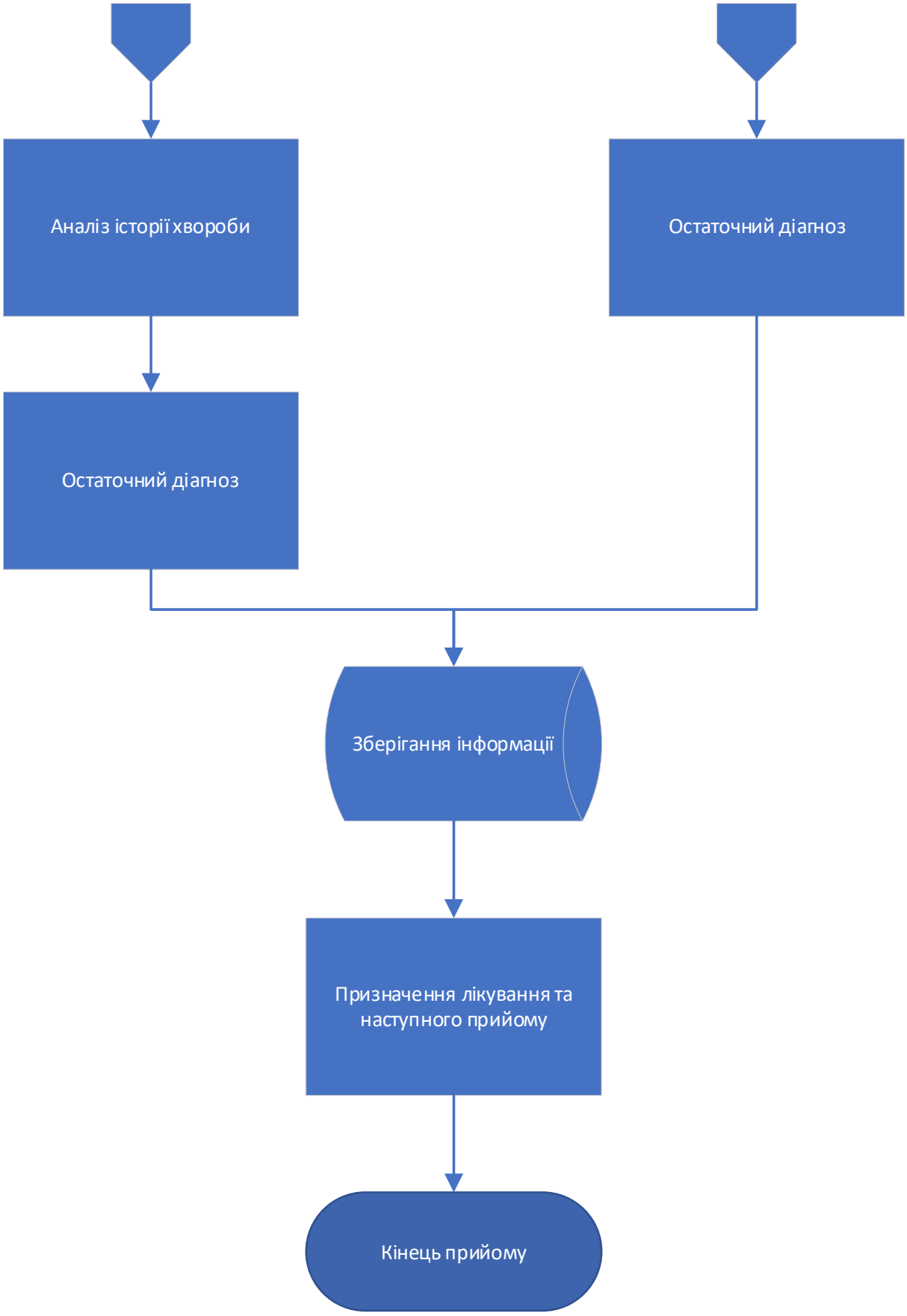
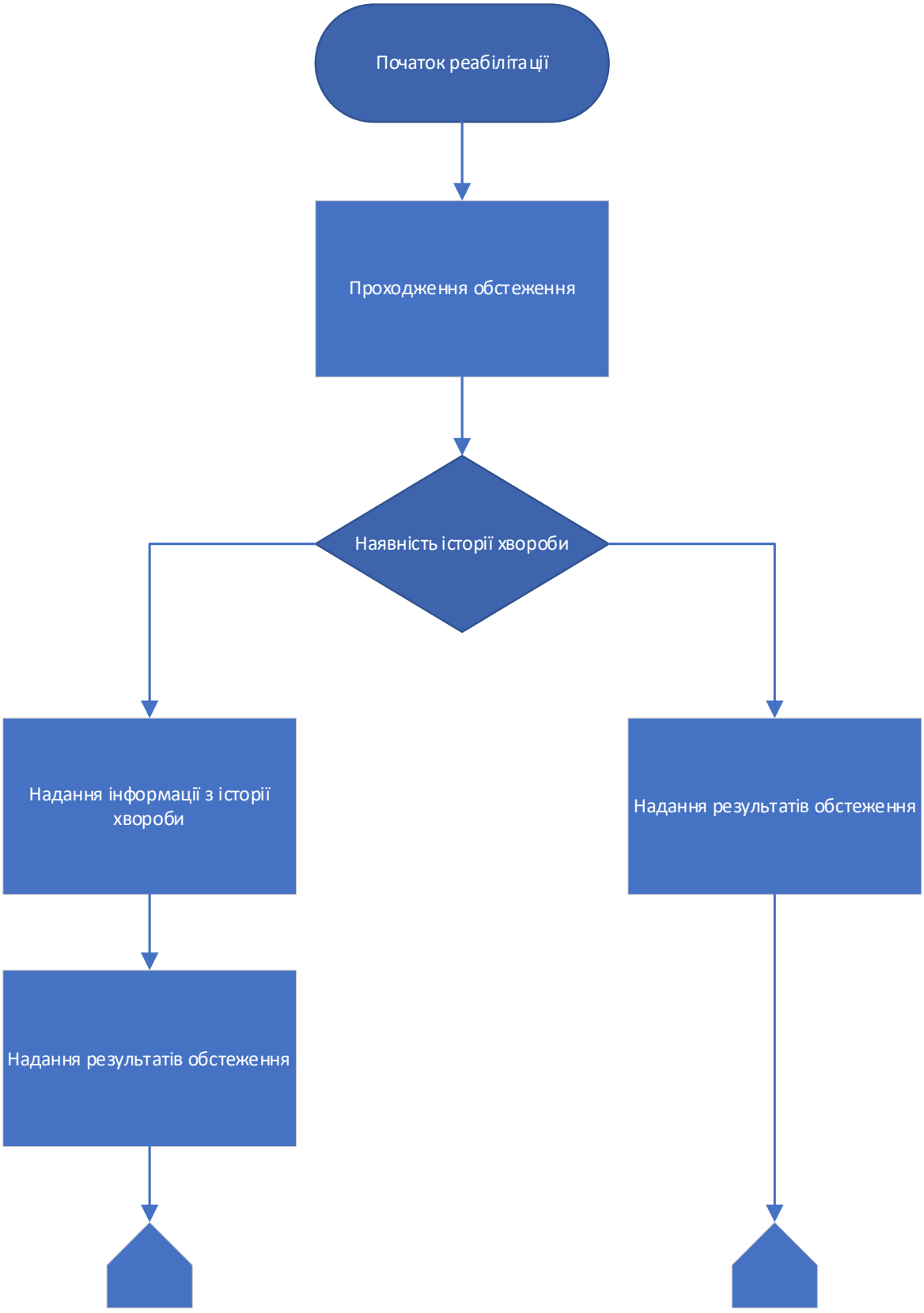


Рис. 3.3 – алгоритм прийому

Лікування пацієнта зображено алгоритмом на рис. 3.4 зіставленою блок-схемою.



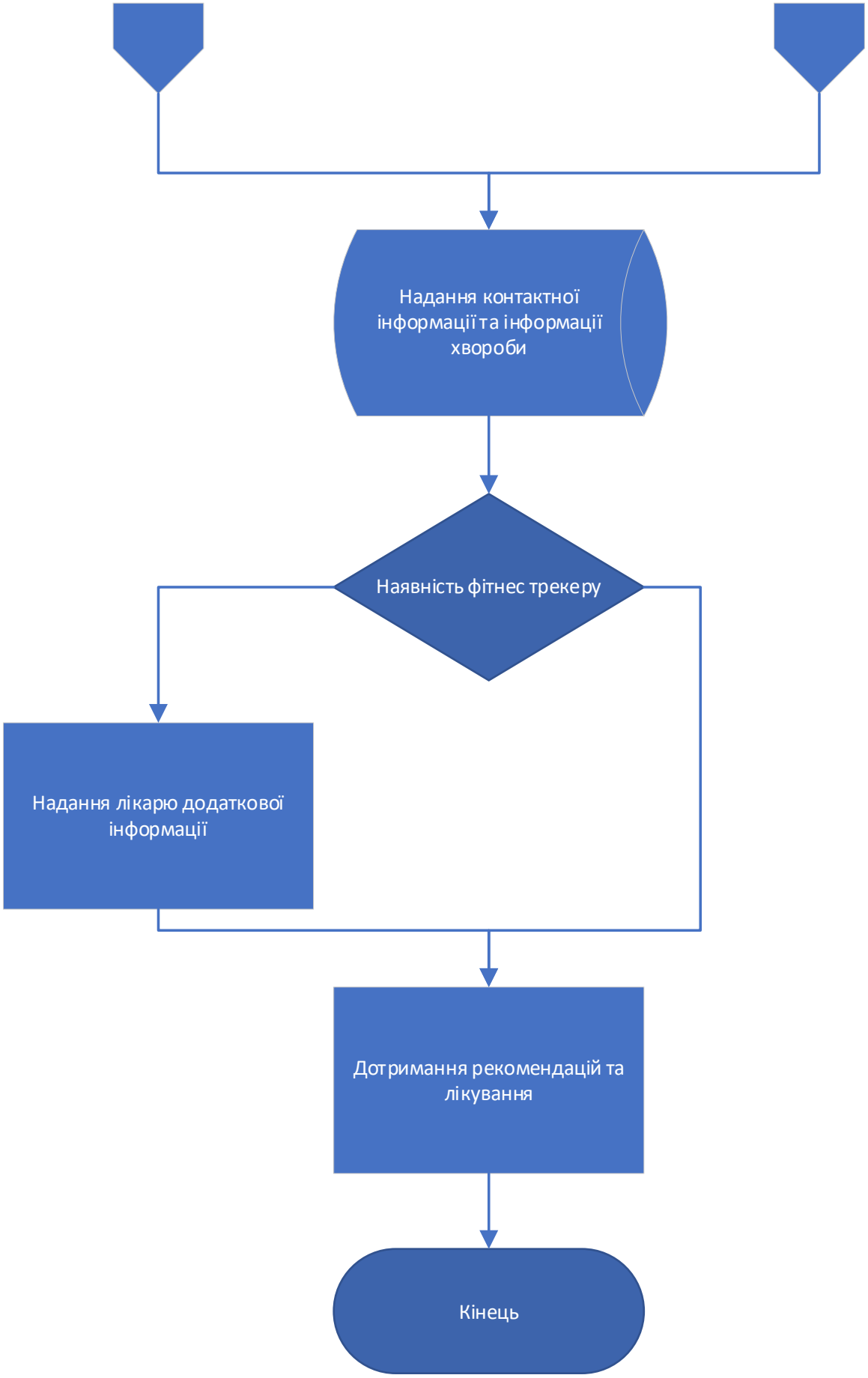
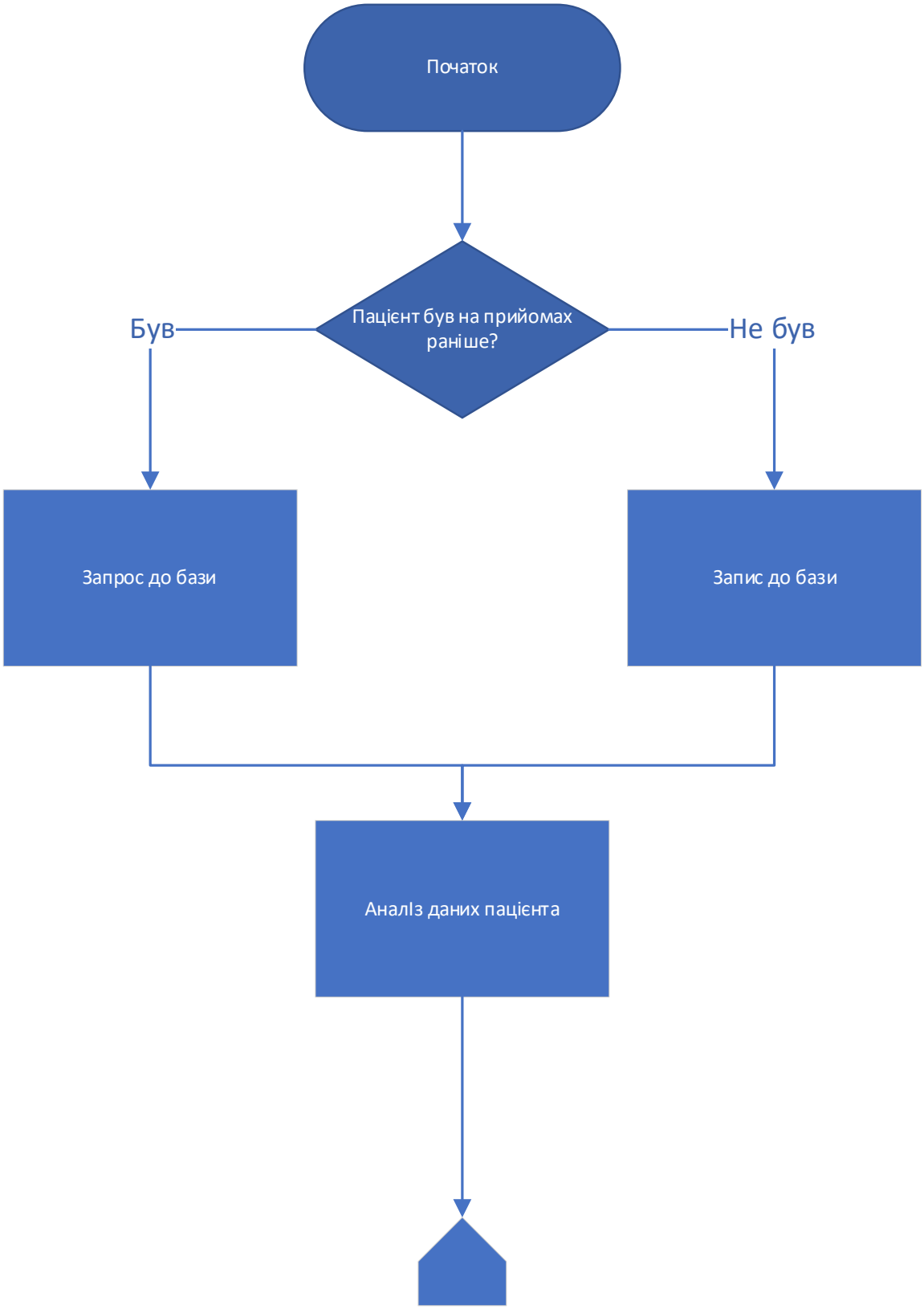
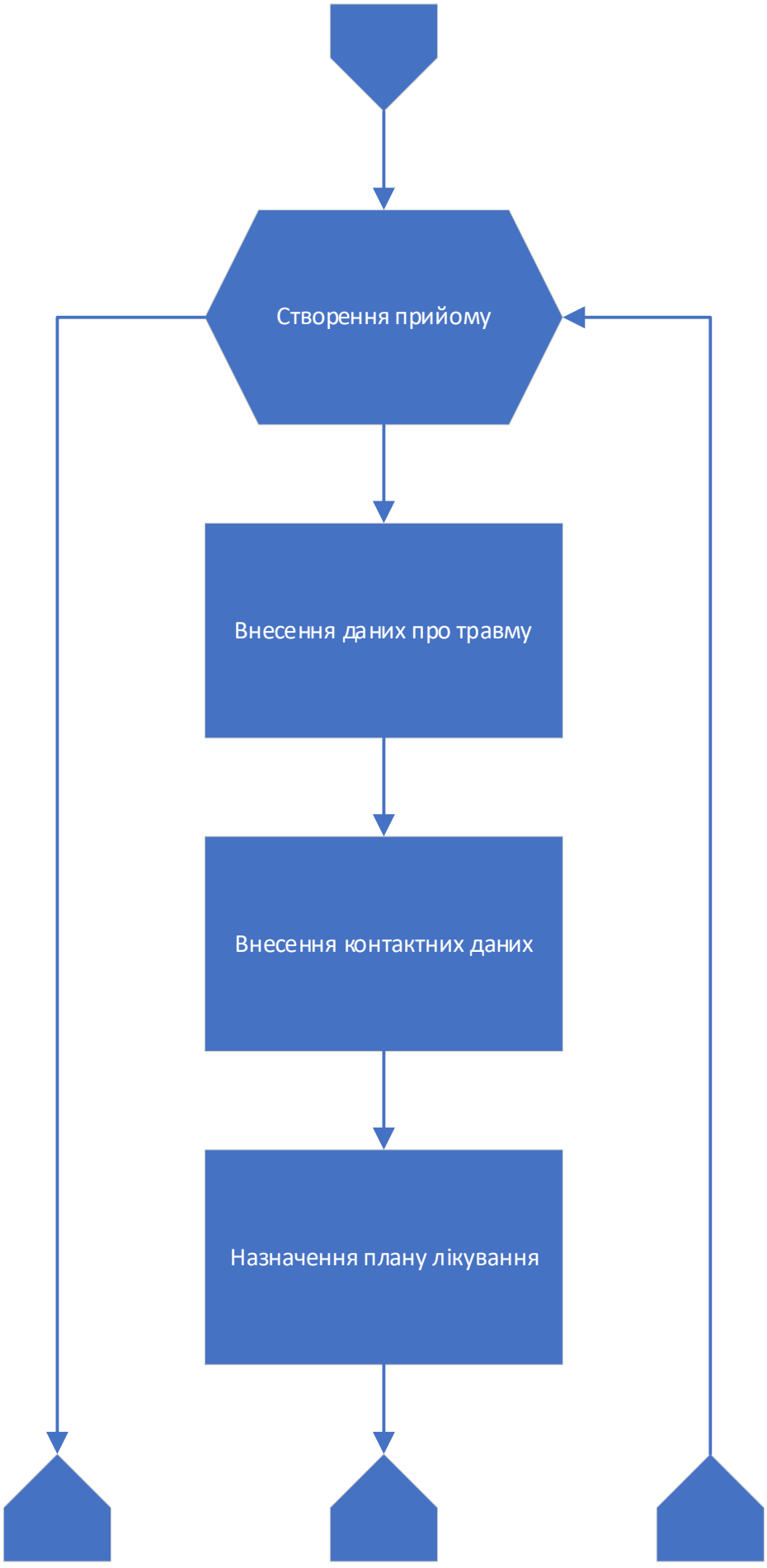
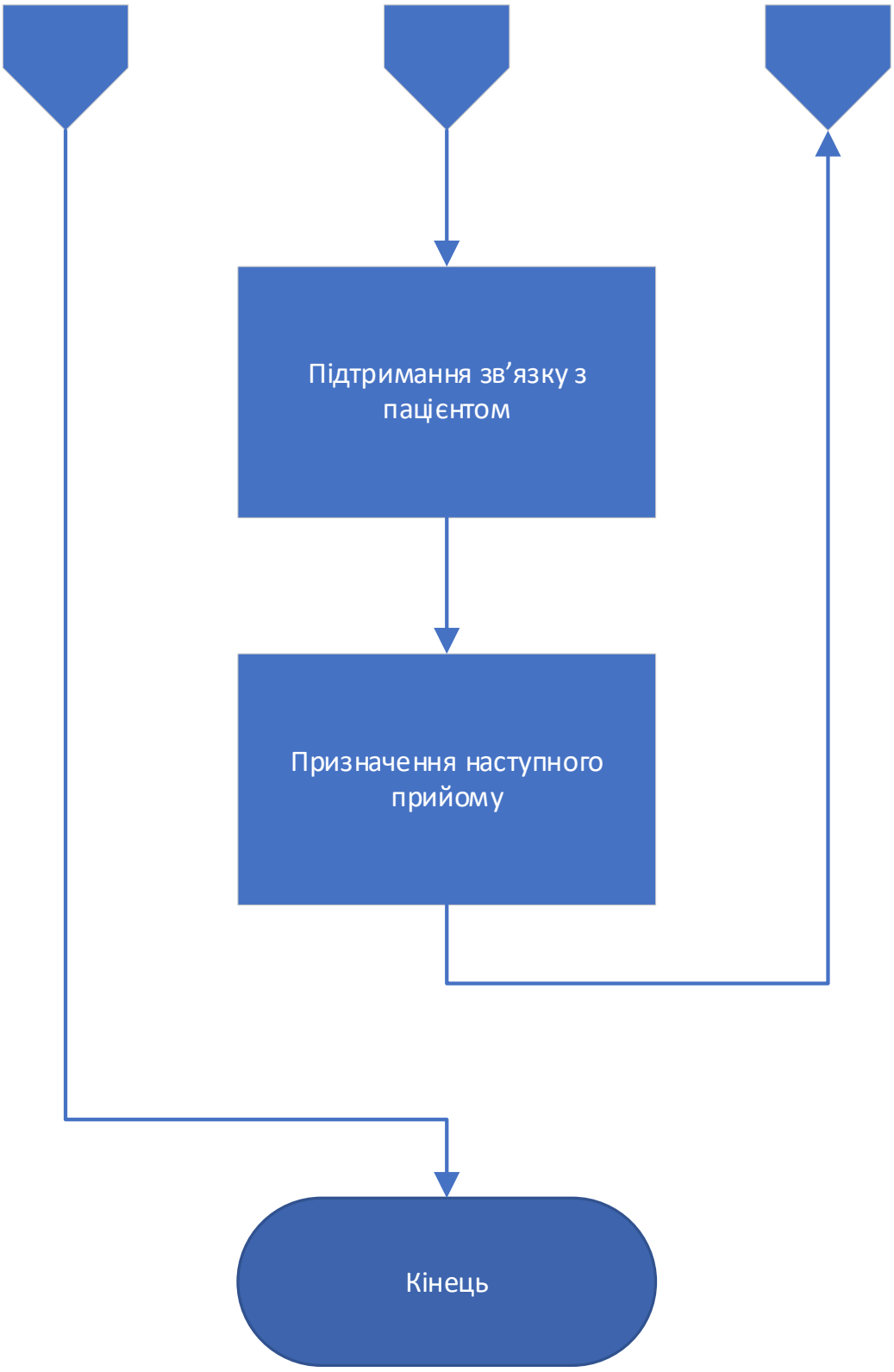


Рис. 3.4 – алгоритм лікування

Загальний алгоритм цілої системи показано на рис.3.5 блок схемою




Створення прийому
Внесення даних про травму
Внесення контактних даних
Назначення плану лікування



3.4 Інтерфейс програми

Лікарі проводять дуже багато часу за комп’ютером, тому найголовнішою ціллю інтерсейсу було створення такої системи, де весь функціонал та вигляд

буде максимально зрозумілий для користування, без великої кількості інформацію, що в свою чергу не буде навантажувати лікаря. Також були обрано кольори, які не будуть втомлювати очі впродовж довготривалої роботи за монітором.

Під час входу до системи, лікар відразу потрапляє у вікно, яке дозволяє йому авторизуватися. Увівши потрібні дані, а саме нік і пароль, система надає доступ до його кабінету для ведення роботи. Приклад такого зображений на рис. 3.3

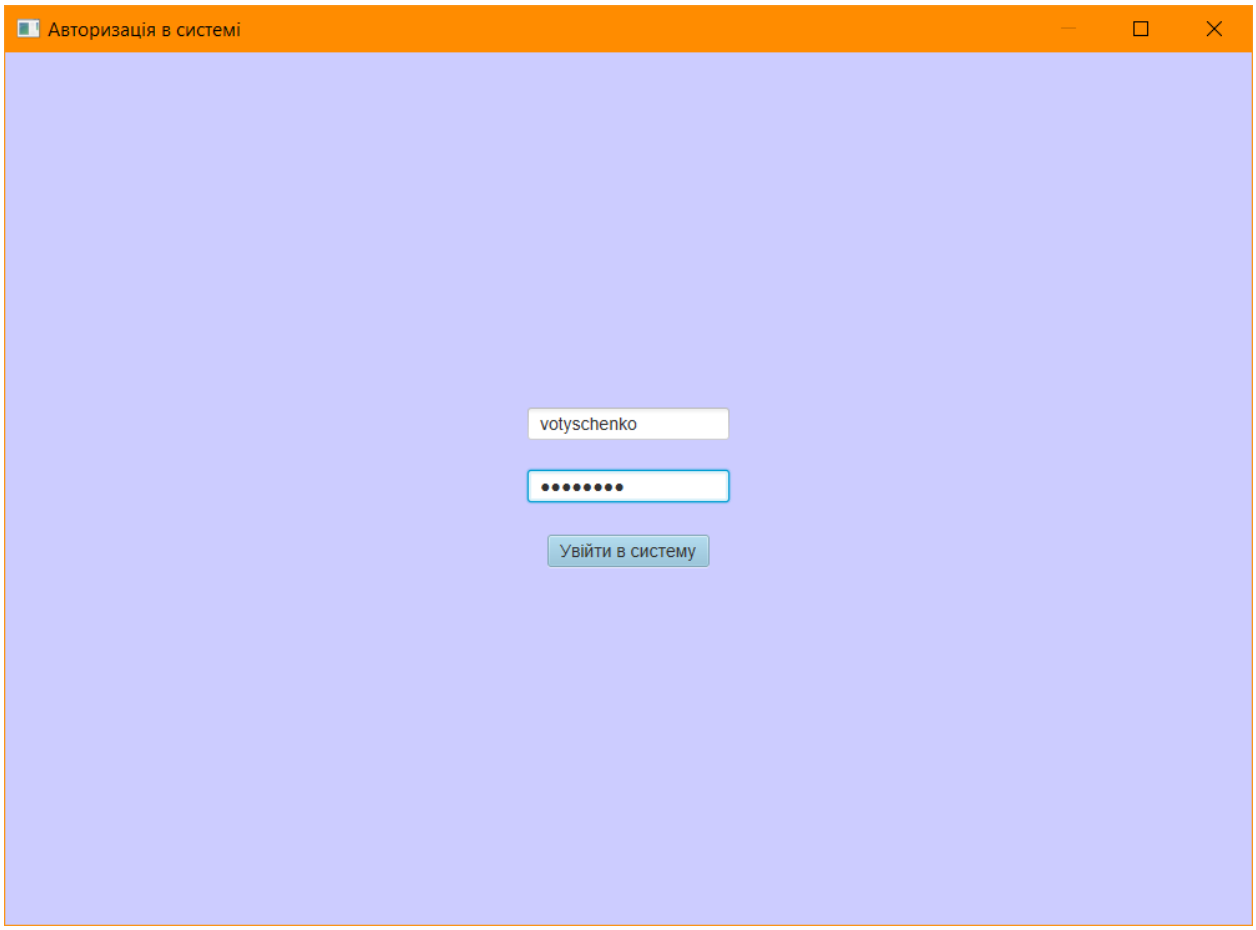


Рисунок 3.3 – початкове вікно

Зайшовши до самої системи, лікар проходить до свого особистого кабінету, зображено на рис.3.4

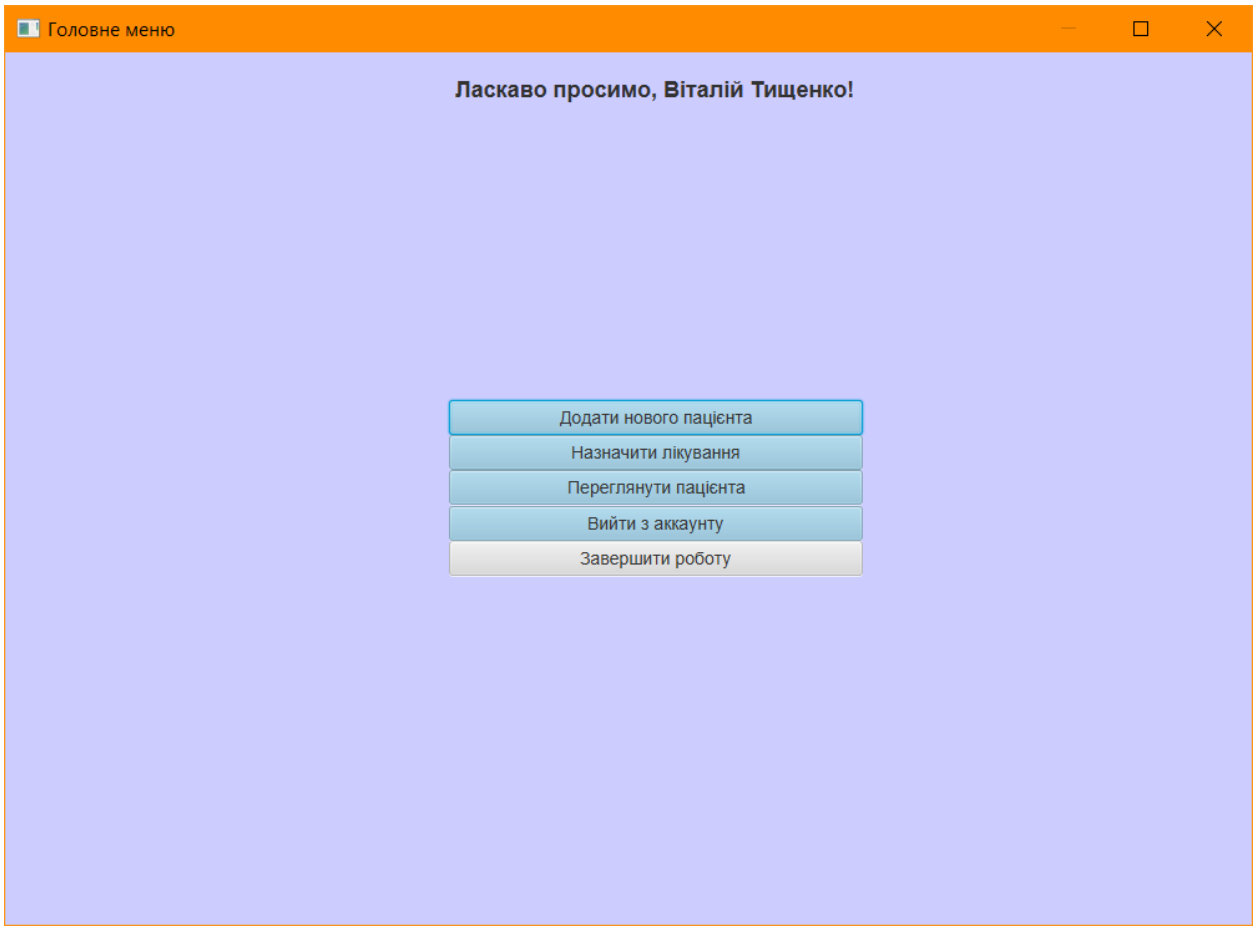


Рисунок 3.4 – робоче вікно

Внесення пацієнта зображено на рис. 3.5.

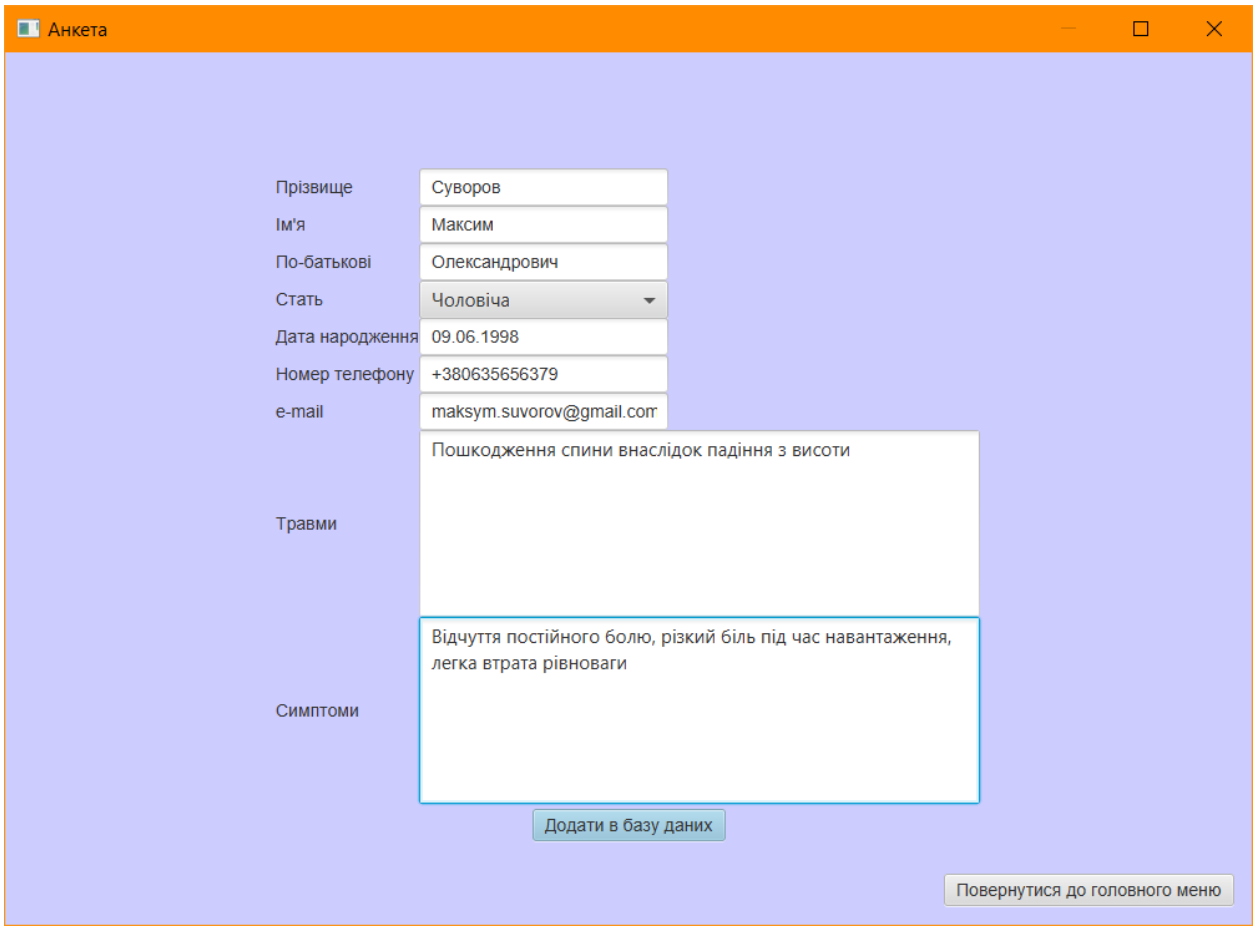


Рисунок 3.5 – реєстрація пацієнта

Будь-який пацієнт, що був на прийомі у лікаря, зберігається в базі. Приклад зображений на 3.6.

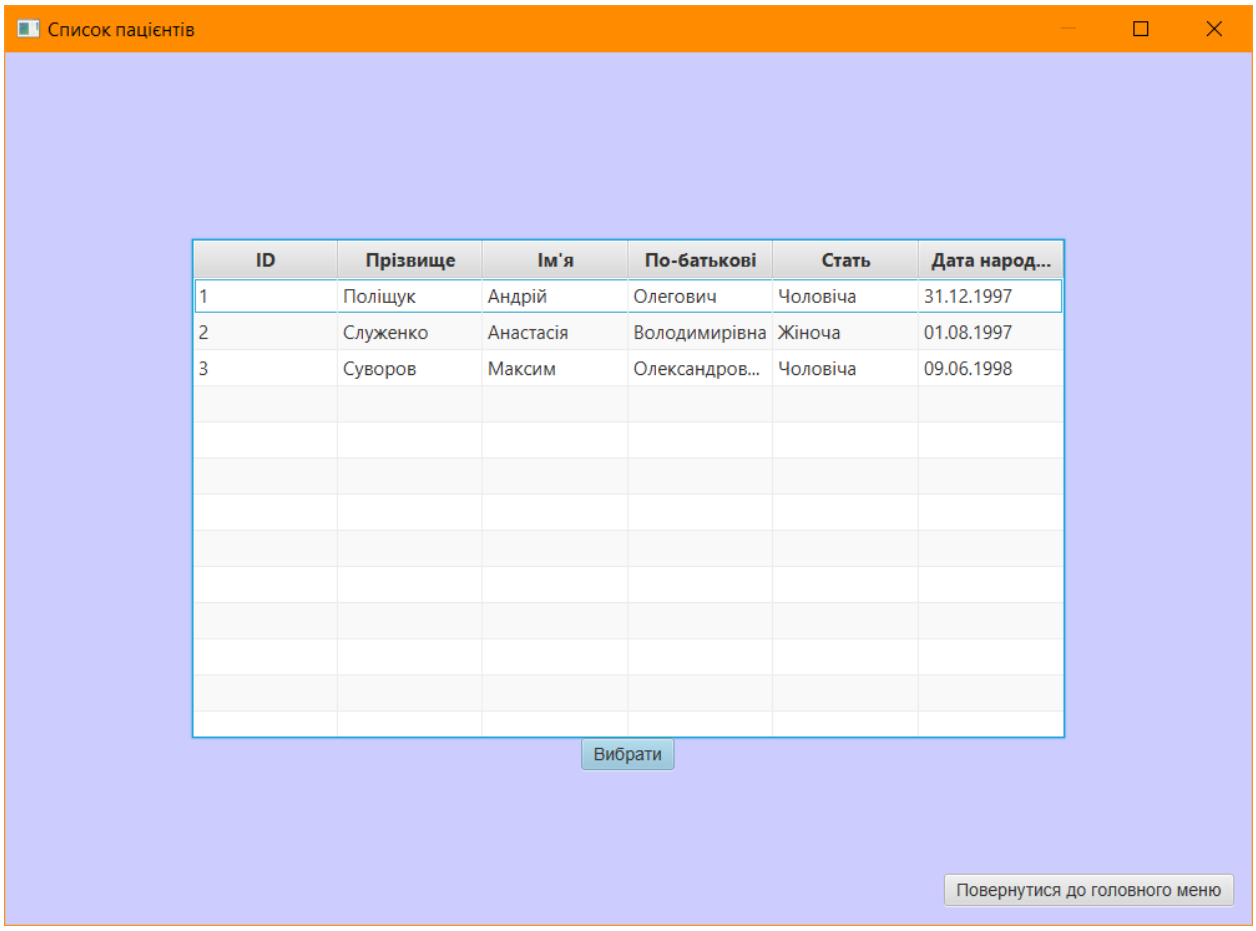


Рисунок 3.6 – база пацієнтів

Після реєстрації самого пацієнта, лікар має можливість вносити детальну інформацію по кожному пацієнту.

Приклад інтерфейсу розміщений на рис.3.7.

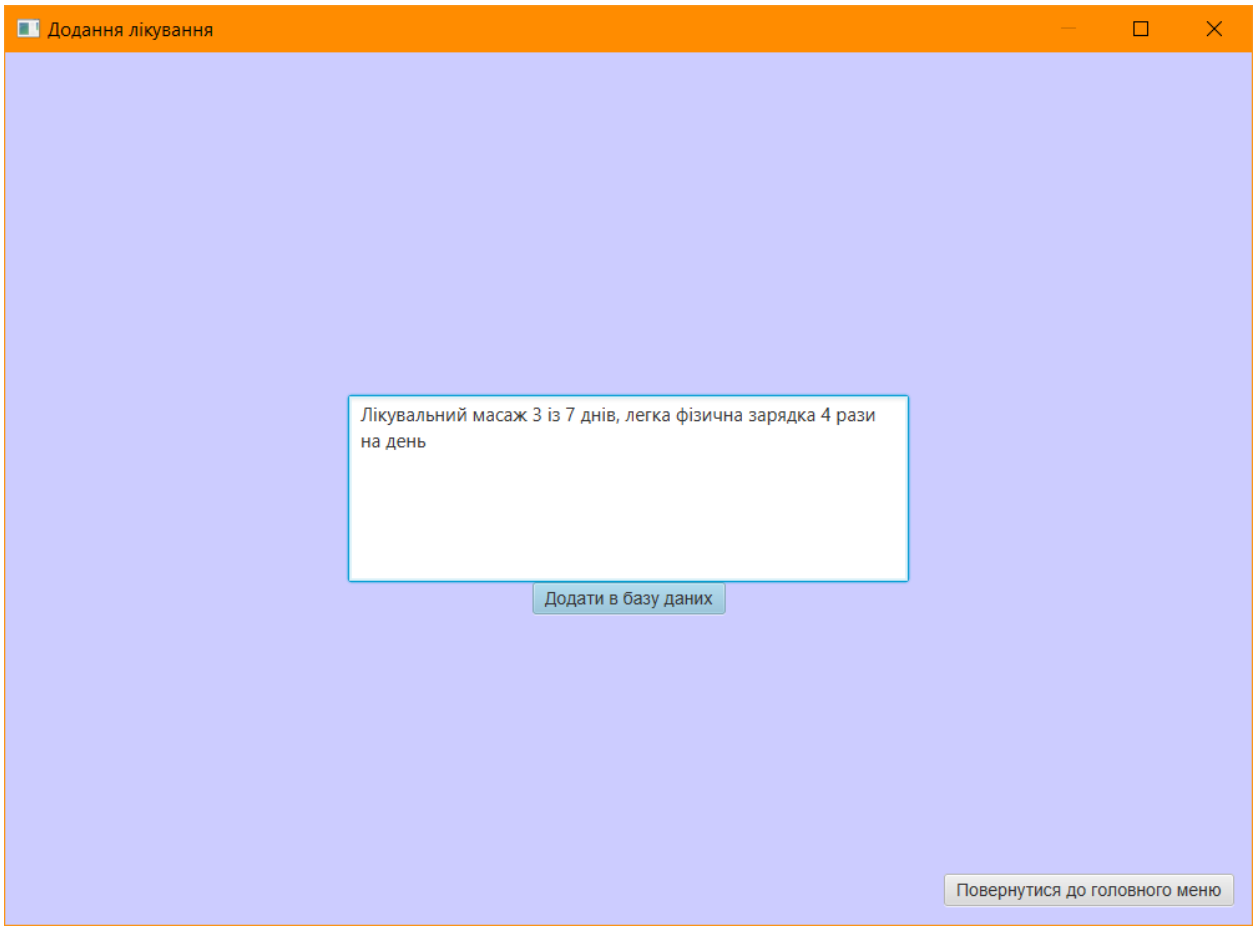


Рисунок 3.7 – приклад лікування

Уся збережена інформація по пацієнту стає доступна для перегляду. Приклад зображений на рис.3.8.

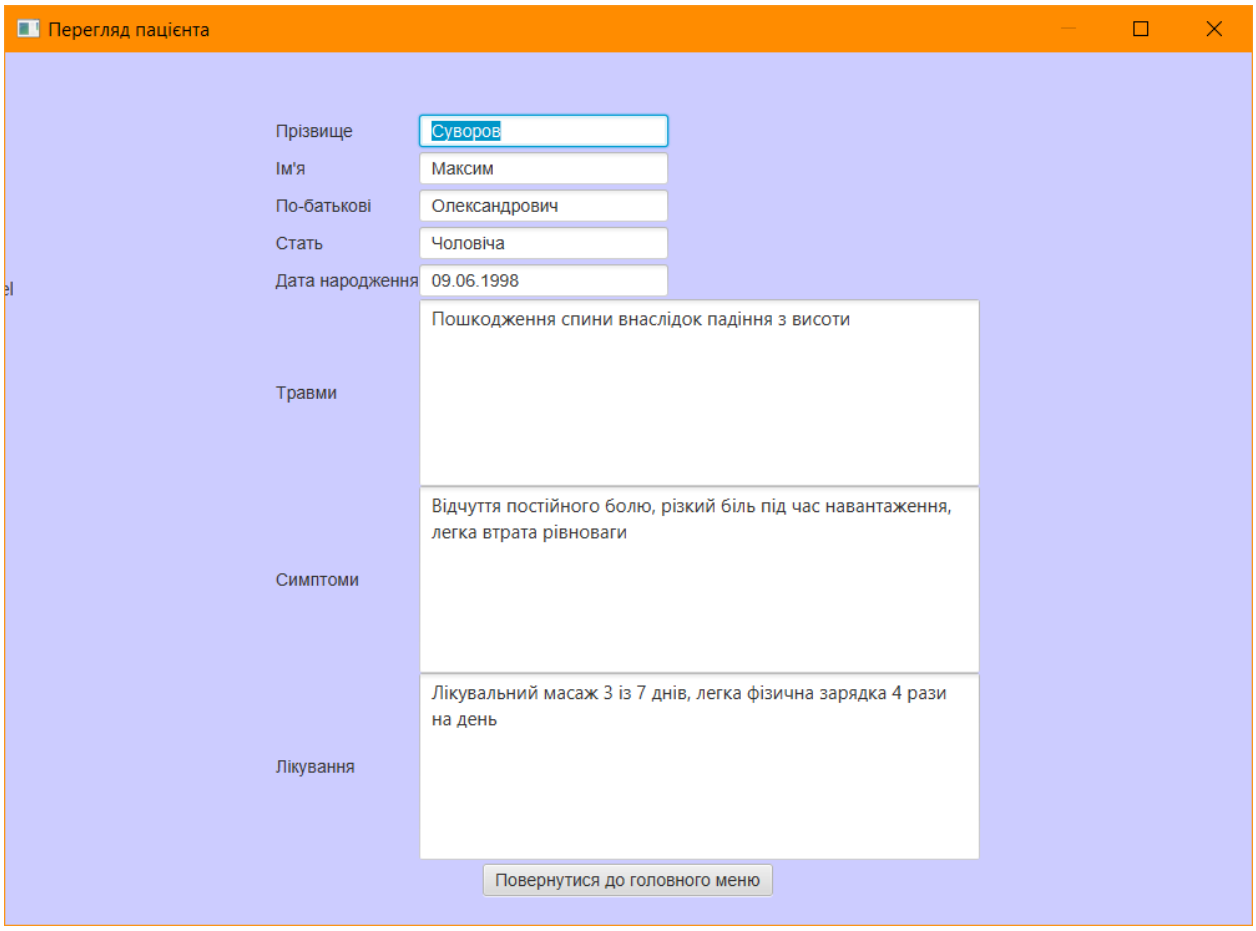


Рисунок 3.8 – приклад інтерфейсу

Дані пацієнта заповнюються за прикладом, що зображений на рис 3.9.

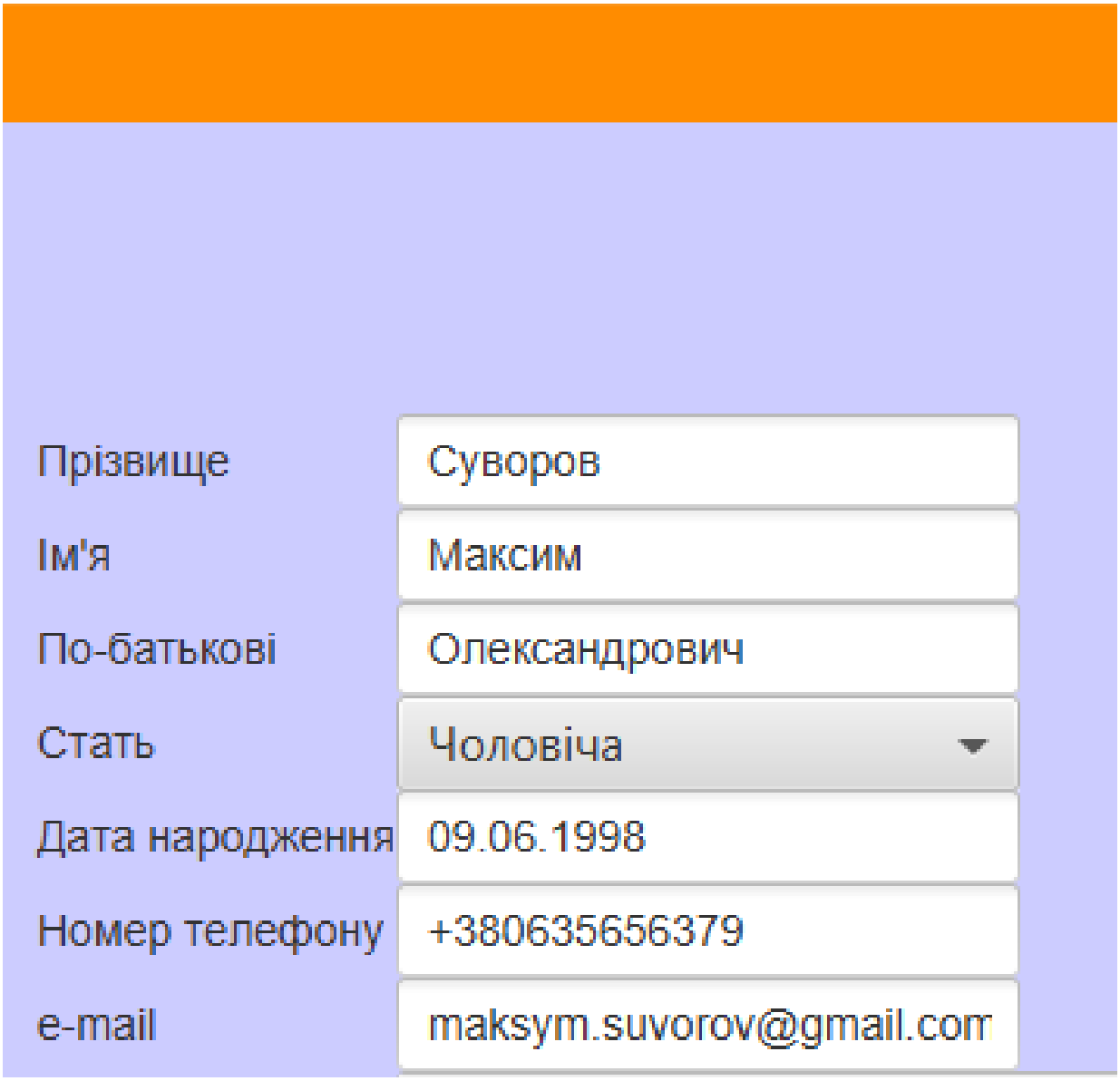


Рис. 3.9. – приклад заповнення контактних даних

Приклад заповнення травм та сиптомів пацієнта зображено на рис. 3.10. У цих полях лікарем реабілітологом заповнюється інформація щодо травми, у даному випадку «Пошкодження спини внаслідок падіння з висоти» та записуються симптоми, на які скаржиться пацієнт (відчуття постійного болю, різкий сильний біль під час навантажень, легка втрата рівноваги).

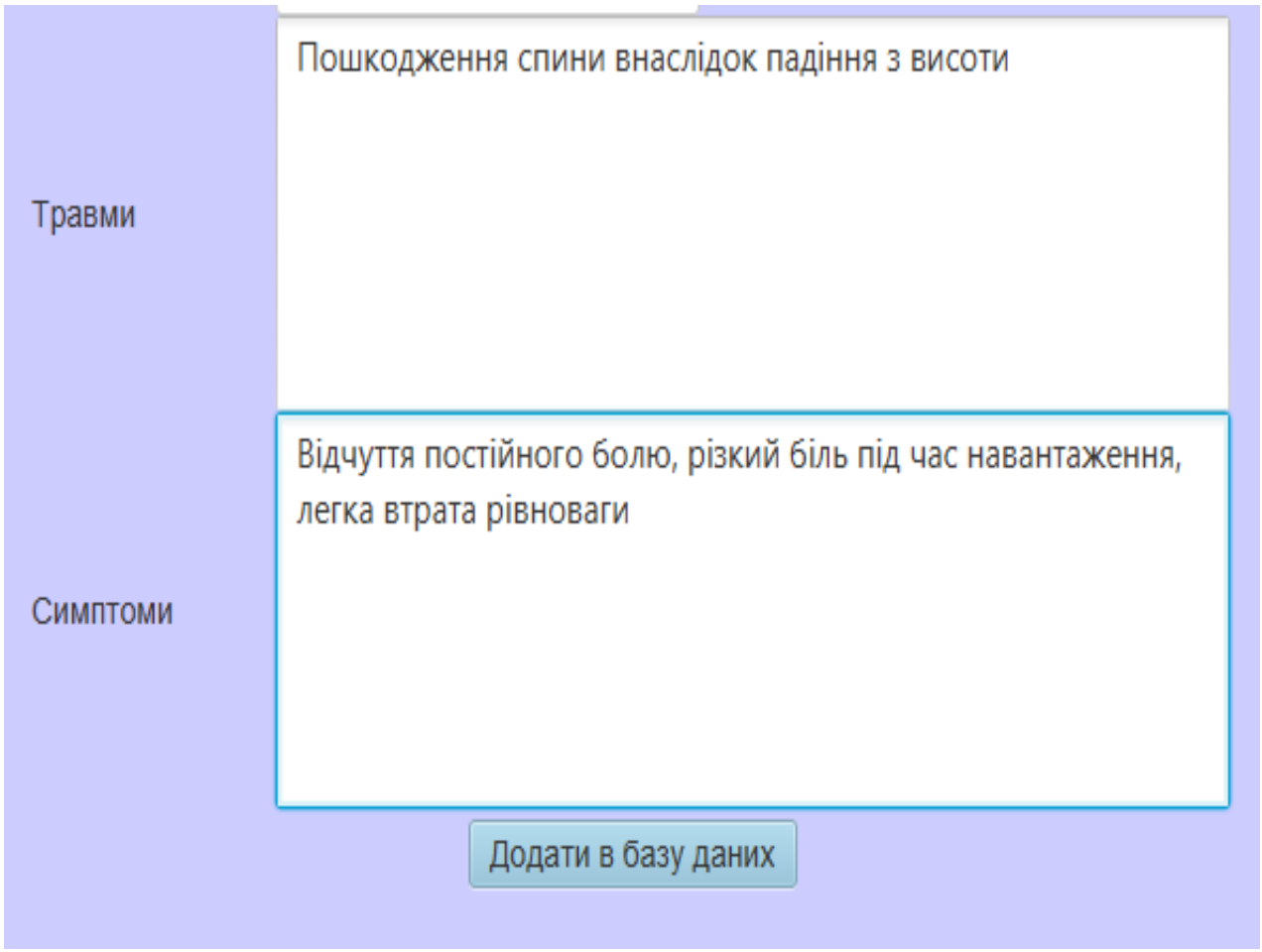


Рис. 3.10 – приклад заповнення травми та симптомів

Також для повноцінного використання програмою було створено кнопки:

- «увійти в систему», рис. 3.11

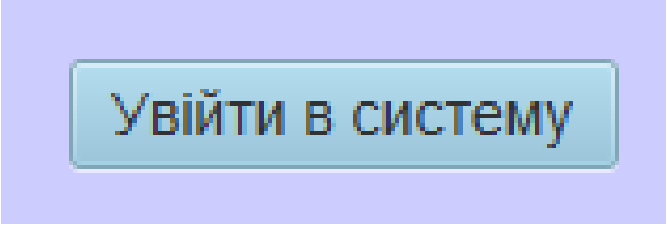


Рис. 3.11 – «увійти в систему»

- «вибрати», рис. 3.12

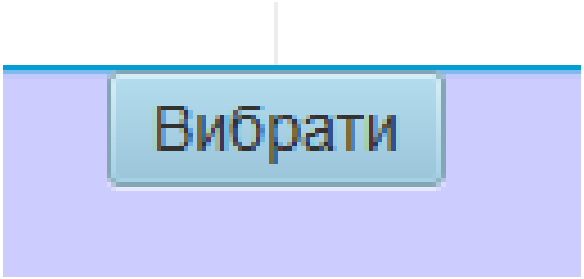


Рис. 3.12 – «Вибрати»

- «Повернутися до головного меню», рис 3.13

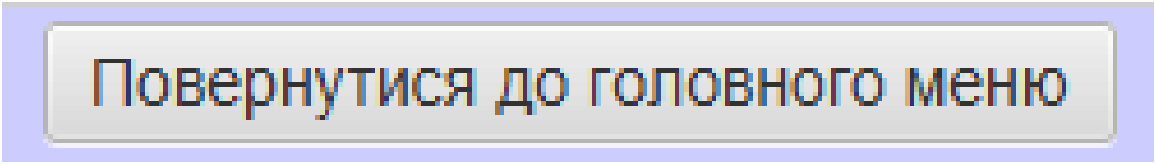


Рис. 3.13 – «Повернутися до головного меню»

- «Додати в базу даних», рис. 3.14

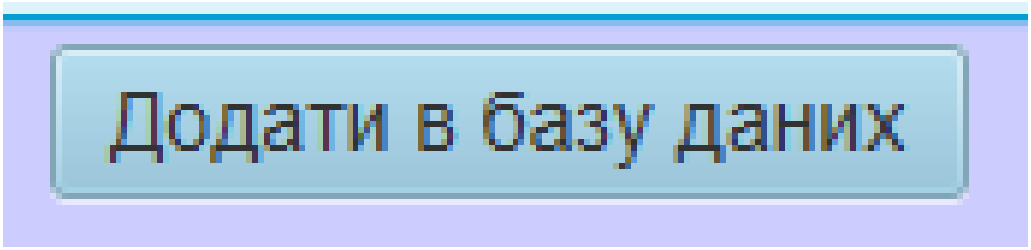


Рис. 3.14 – «Додати в базу даних»

Висновки до розділу 3.

У цьому розділі було детально розглянуто алгоритми функціонування системи та структуру даних, з якими вона працює. Проаналізовано ключові етапи роботи, починаючи від введення даних пацієнта до відстеження його прогресу в реабілітаційному процесі.

Також було представлено опис інтерфейсу програми, що є одним із головних елементів її функціональності. Наведено приклад заповнення інформаційних полів із поясненнями щодо кожного етапу взаємодії користувача з програмою. Це включає:

- Введення персональних даних пацієнта.
- Заповнення історії хвороби та результатів обстеження.
- Внесення плану реабілітації та записів про прогрес лікування.

Такий підхід дозволяє візуалізувати процес роботи системи, забезпечити інтуїтивно зрозумілий функціонал та оптимізувати робочий час лікаря.

ЗАГАЛЬНІ ВИСНОВКИ

У сучасному світі автоматизація процесів є ключовим напрямком розвитку. Машини та програмні системи дедалі більше замінюють ручну працю, забезпечуючи високу точність, швидкість і зниження затрат людських ресурсів. Водночас є сфери діяльності, які неможливо повністю автоматизувати через потребу в людській експертизі, емпатії та гнучкому підході.

Розробка спеціалізованої системи для лікарів реабілітологів є важливим кроком до оптимізації їхньої роботи. Впровадження такої системи:

- Зменшує потребу у використанні паперових документів, що спрощує документообіг.
- Економить час лікаря, що дозволяє приділяти більше уваги безпосередньому лікуванню пацієнтів.
- Знижує ймовірність помилок через автоматизацію рутинних операцій.
- Сприяє швидкому доступу до даних та їх оперативному обміну між медичними працівниками.

Основні завдання, які ставилися під час розробки:

1. Полегшення роботи лікаря:
    - Зменшення рутинних операцій.
    - Збереження сил, пильності та концентрації медичних працівників.
2. Відмова від паперового документообігу:
    - Використання цифрових форматів для ведення історій хвороб та звітів.
    - Швидке опрацювання інформації та обмін даними в межах медичних установ.
3. Підвищення ефективності лікувального процесу:
    - Створення комфортних умов для лікарів і пацієнтів.

- Прискорення процесів прийому, діагностики та реабілітації.

Проєктування та функціонал

У ході роботи було розроблено систему, яка:

- Забезпечує лікарів інструментами для ведення історій хвороб, записів прийомів та рекомендацій.
- Включає алгоритми прийому пацієнтів і розробки плану лікування.
- Надає можливість відстежувати процес реабілітації пацієнта в режимі реального часу.

Алгоритми роботи системи

Для візуалізації роботи системи розроблено блок-схеми, які відображають:

- Дії пацієнта: запис на прийом, проходження обстеження, виконання рекомендацій лікаря.
- Дії лікаря: аналіз даних, розробка плану лікування, оцінка прогресу пацієнта.
- Процеси взаємодії між лікарем і пацієнтом у рамках лікувального циклу.

Впроваджена система повністю відповідає поставленим вимогам, забезпечуючи:

- Автоматизацію рутинних операцій.
- Зниження навантаження на лікарів.
- Відмову від паперових носіїв інформації.
- Поліпшення якості та швидкості надання медичних послуг.

Таким чином, система створює умови для сучасного, ефективного та комфортного виконання професійних обов’язків лікаря реабілітолога.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Клас File [електронне джерело] – https://tdmuv.com/

2. Фізична реабілітація[електронне джерело] – https://tdmuv.com/

3. My SQL [електронне джерело] – http://ruszura.in.ua/uncategorized/

4. Лікар реабілітолог [електронне джерело] – https://altmed.if.ua/

5. Java як мова программування [електронне джерело] – http://ruszura.in.ua/uncategorized/

6. Переваги [електронне джерело] – https://qagroup.com.ua/

7. My SQL [електронне джерело] – http://www.znannya.org/

8. Інтерфейси [електронне джерело] – http://iwanoff.inf.ua/java_ua/

9. Java FX [електронне джерело] – https://uk.photo-555.com/

10. Java FX [електронне джерело] – https://metanit.com/