**ВСТУП**

Інформаційні системи в першу чергу призначені оптимізувати та полегшити виконання рутинної ручної роботи працівників різних галузей. Особливо актуально це в медицині, де швидкість виконання роботи та прийняття рішень часто надважливі для здоров’я людини. Зокрема, при проведенні ендоскопічних досліджень в значній мірі лікар зайнятий формуванням протоколу та заповненням іншої стандартної документації, інформація в якій часто повторяється.

Основною проблемою є те, що різні лікарі спостерігаючи одну ситуацію описують її по-різному. Це призводить до того, що втрачається можливість порівняння та аналізу результатів досліджень, зроблених у різних центрах. Тому постала гостра потреба у використанні стандартної термінології в даній галузі.

За виконаними ендоскопічними дослідженнями кожен лікар формує звітність. Звіти бувають місячними, квартальними та річними. Формування звіту зводиться до підрахунку кількості та систематизації проведених досліджень різного виду. Ця робота займає досить багато часу, а комп’ютер виконає її за лічені секунди.

На сьогоднішній день подібної системи в Україні не існує, а весь облік результатів ведеться вручну, що призводить до часових та ресурсних затрат, втрат деякої інформації.

Враховуючи вище зазначені проблеми, виникла потреба в розробці інформаційної системи, яка б надавала можливість формування протоколу ендоскопічного дослідження з використанням універсальної стандартної термінології, автоматично формувала заключення та рекомендації, заповнювала щоденний журнал, створювала звіти за потрібний проміжок часу.

У зв’язку з інтенсивним розвитком медичної галузі, новими відкриттями та дослідженнями, з часом виникне потреба в розширенні та редагуванні використовуваної термінології.

Для розробки програмного забезпечення було обрано платформу Java EE [1].

Для комп’ютеризації процесу створення протоколу ендоскопічного дослідження було використано спеціально розроблену для цього мінімальну стандартну термінологію (МСТ) [2]. Її легко можна представити у вигляді таблиць бази даних, що дає змогу виробити ієрархію класів, а також розробити засоби для роботи із ними.

Сьогодні популярності набувають системи у вигляді веб-застосунків. Такі системи є багатокористувацькими та легкодоступними для користувачів більшості пристроїв. Інформаційна система, представлена в даній роботі, була розроблена з використанням фреймворків Spring MVC та Hibernate 3. Обидва фреймворки використовують анотації для задання налаштувань, а також чудово співпрацюють разом.

Spring відповідає за обробку запитів від користувача, формування моделі та вибір відображення для даних, які отримає користувач. Для роботи з інформацією, яка знаходиться в базі даних, використовується Hibernate. Він надає засоби для роботи із таблицями з бази даних як із Java класами, що полегшує написання запитів та роботу із отриманою інформацією.

Для розробки інтерфейсу користувача використано популярні бібліотеки Bootstap, jQuery та технологію JSP. Ці засоби дозволили внести динаміку до формування веб-сторінок та зробили їх одностилевими.

Темою дипломної роботи було обрано створення інформаційної системи, призначеної для створення та обробки протоколів ендоскопічних досліджень та роботи із термінологією, що використовується в ній.

1. Задача розробки Інформаційної системи обліку результатів ендоскопічних досліджень

Метою дипломної роботи є розробка й створення інформаційної системи для ведення обліку результатів ендоскопічних досліджень з додатковим функціоналом.

Мета інформаційної системи полягає у тому, щоб створити протокол дослідження за вхідними даними, що задає користувач, занести його в журнал, надати можливість редагування протоколів, формування звітів, обліку пацієнтів.

**1.1. Основні вимоги до розроблюваної системи**

Розроблюваний програмний продукт повинен бути багатокористувацьким та мати захист у вигляді логіна та пароля. Реєстрація нових користувачів повинна проводитися вручну через адміністратора.

Підсистема формування протоколу повинна:

* надавати можливість вибору пацієнта, що проходив обстеження, медичного апарату та дезінфекційного розчину, які використовувались, направлення пацієнта, вибір типу обстеження;
* мати зручний інтерфейс для формування протоколу з використанням універсальної структурованої мінімальної стандартної термінології (МСТ) [2];
* надавати можливість добавити маніпуляції та операції та їх описи відповідно;
* автоматично формувати заключення і рекомендації.

Підсистема обліку маніпуляцій повинна:

* фільтрувати маніпуляції з результатом і без нього;
* надавати можливість внесення результату;
* надавати можливість редагування опису.

**1.2. Засоби управління МСТ та звітами**

Не менш важливими задачами системи є надання засобів для управління словниками, формування річних, квартальних і місячних звітів.

Для роботи з термінологією необхідно розробити підсистему, яка б могла:

* добавляти нові терміни;
* редагувати записи;
* видаляти записи;
* забезпечувати зв’язок відповідно до ієрархії.

Звіти та щоденник повинні формуватися відповідно до наказу МОЗ України №393 [2].

База даних повинна зберігати інформацію про дані, що вніс користувач, а саме інформацію про пацієнтів, про кожне обстеження, маніпуляцію та операцію. Також база даних повинна містити ієрархію словників. Всі дані повинні зберігатися на віддаленому сервері під керуванням MySQL Server.

Користувацький інтерфейс повинен бути інтуїтивно зрозумілим і простим. Оскільки паперові медичні карточки не відмінені законодавством, повинна бути функція для друку протоколу ендоскопічного дослідження. Це також потрібно тому, що пацієнт не є користувачем системи і може вимагати письмовий доказ проведення обстеження для інших установ.

**2. Аналіз проблеми Обліку результатів ендоскопічних досліджень**

Основна проблема полягає в тому, що не існує ендоскопічної систем звіту, яка б використовувала універсальну стандартизовану термінологію, в результаті чого втрачається можливість для аналізу і порівняння даних, зібраних у різних центрах.

**2.1. Огляд існуючих рішень**

Програмна система, розроблена робочою групою ASGE, використовувала для створення протоколів ендоскопічних досліджень всеосяжну термінологію, що призвело до перенавантаження і ускладнення системи.

Для вирішення цієї проблеми було використано мінімальну стандартну термінологію (МСТ) версії 2.0 [2]. Ця термінологія має чітку структуру у вигляді дерева і призначена для використання у програмних системах. У вершині вид обстеження, що ділиться на ограни, які необхідно описати при його проведенні. Кожен орган поділяється на ділянки, вони в свою чергу на терміни, терміни на ознаки, ознаки на характеристики. Термін, ознака чи характеристика можуть бути останньою гілкою дерева. У цьому випадку потребується введення інформації від користувача. В залежності від змісту це може бути ділянка в якій щось виявлено чи розміри того що виявлено. Ця інформація повністю непередбачувана тому автоматизувати її неможливо. Завдяки такій структурі значно зменшується кількість інформації що потрібно вводити вручну, а створені протоколи досліджень може прочитати і зрозуміти кожен спеціаліст цієї галузі.

Ще однією проблемою є формування звітів, що зводиться до ручного підрахунку кількості виконаних досліджень за певний період часу. Звіт представляється у вигляді таблиці [3]. Для її автоматичного заповнення потрібно розробити алгоритм, що буде підраховувати інформацію для кожної колонки таблиці. Вся інформація необхідна для цього повинна знаходитись у базі даних, що зводить вирішення задачі до написання необхідних SQL запитів.

Великі витрати часу пов’язані з записом інформації в журнал щоденного обліку, в якому дублюється інформація з медичної карточки пацієнта.

**2.2. Актуальність системи в майбутньому**

Для збереження актуальності розроблюваної системи з часом виникає потреба розробки механізму управління термінологію (словниками). Набір інструментів для додавання, видалення, редагування та встановлення зв’язків між елементами мінімальної стандартної термінології забезпечить гнучкість системи і зробить її підтримку легшою.

Із системою повинні одночасно працювати необмежена кількість користувачів тому виникає необхідність створення багатокористувацького додатку із серверною частиною. Для цього пропонується розробити систему у вигляді веб-сайту.

**3. Методи реалізації програмної системи**

Для розробки серверної частини було обрано мову програмування Java, що чудово підходить для написання клієнт-серверних застосунків, в тому числі веб-застосунків. Для розробки використано середовище програмування IntelliJ IDEA, сервер Apache Tomcat та MySQL сервер для бази даних.

**3.1. Огляд фреймворків Spring MVC та Hibernate 3**

Для написання серверної частини було обрано сучасний фреймворк Spring MVC 4.1.1 та Hibernate 3. Вони надають зручні засоби для розробки web-застосунків та засоби для роботи із базою даних.

Java — [об’єктно-орієнтована](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [мова програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), випущена компанією [Sun Microsystems](https://uk.wikipedia.org/wiki/Sun_Microsystems) у [1995](https://uk.wikipedia.org/wiki/1995) році як основний компонент [платформи Java](https://uk.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0_Java). Зараз мовою займається компанія [Oracle](https://uk.wikipedia.org/wiki/Oracle_Corporation), яка придбала [Sun Microsystems](https://uk.wikipedia.org/wiki/Sun_Microsystems) у [2009](https://uk.wikipedia.org/wiki/2009) році. Синтаксис мови багато в чому схожий на [C](https://uk.wikipedia.org/wiki/C_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та [C++](https://uk.wikipedia.org/wiki/C%2B%2B). У офіційній реалізації Java-програми [компілюються](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80) у [байт-код](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), який при виконанні інтерпретується [віртуальною машиною](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_Java) для конкретної платформи [1].

Мова значно запозичила синтаксис із [C](https://uk.wikipedia.org/wiki/C_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) і [C++](https://uk.wikipedia.org/wiki/C%2B%2B). Зокрема, взято за основу об'єктну модель С++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в С/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій java дозволяє викликати підпрограми, написані іншими мовами програмування [6].

У створенні мови програмування Java було чотири початкові цілі:

* синтаксис мови повинен бути “простим, об'єктно-орієнтовним та звичним”;
* реалізація має бути “безвідмовною та безпечною”, а також “високопродуктивною”;
* повинна зберегтися “незалежність від архітектури та портативність”;
* мова має бути “динамічною, інтерпретованою та підтримувати мультиопрацьовування”.

Spring Framework — це програмний каркас ([фреймворк](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)) з відкритим кодом та контейнера з підтримкою [інверсії управління](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D1%96%D1%8F_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F) для платформи Java.

Spring має власну MVC платформу веб-застосувань, яка не була запланована спочатку. Розробники Spring вирішили написати її як реакцію на те, що вони сприйняли як невдачу конструкції (тоді) популярного Apache Struts, а також інших доступних веб-фреймворків [7].

Клас DispatcherServlet є основним контролером фреймворка і відповідає за делегування управління різним інтерфейсам на всіх етапах виконання HTTP-запиту.

Як і Struts, Spring MVC є фреймворком, орієнтованим на запити. У ньому визначені стратегічні інтерфейси для всіх функцій сучасної запито-орієнтованої системи. Мета кожного інтерфейсу — бути простим і зрозумілим, щоб користувачам було легко його заново імплементувати, якщо вони того побажають. MVC прокладає шлях до більш чистого front-end-коду. Всі інтерфейси тісно пов'язані з Servlet API. Цей зв'язок розглядається деякими як нездатність розробників Spring запропонувати для веб-застосувань абстракцію більш високого рівня. Однак цей зв'язок залишає особливості Servlet API доступними для розробників, полегшуючи роботу з ним.

Центральною частиною Spring Framework є контейнер Inversion of Control, який надає засоби конфігурації і управління об'єктами Java за допомогою відображення. Контейнер відповідає за управління життєвим циклом об'єкта: створення об'єктів, виклик методів ініціалізації і конфігурація об'єктів шляхом зв'язування їх між собою.

Об'єкти, які створюються контейнером, також називаються керованими об'єктами (beans). Зазвичай конфігурація контейнера здійснюється шляхом завантаження XML-файлів, що містять визначення bean’ів і надають інформацію, необхідну для створення bean’ів.

Об'єкти можуть бути отримані одним з двох способів:

* пошук залежності — шаблон проектування, в якому об'єкт, що викликає, запитує у об'єкта-контейнера екземпляр об'єкта з певним ім'ям або певного типу.
* впровадження залежності — шаблон проектування, в якому контейнер передає екземпляри об'єктів по їх імені іншим об'єктам за допомогою конструктора, властивості або фабричного методу.

Найбільш важливі інтерфейси, визначені Spring MVC, перераховані нижче:

* HandlerMapping: вибір класу і його методу, які повинні обробити даний вхідний запит на основі будь-якого внутрішнього або зовнішнього для цього запиту атрибута або стану;
* Controller: управляє процесом перетворення вхідних запитів в адекватні відповіді. Діє як ворота, направляючі всю інформацію, що надходить, перемикає потік інформації з моделі в представлення і назад;
* View: відповідальний за повернення відповіді клієнту у вигляді текстів і зображень, деякі запити можуть йти прямо до View, не заходячи в Model, інші проходять через всі три шари;
* ViewResolver: вибір, яке саме View має бути показано клієнту;
* LocaleResolver: отримання і, можливо, збереження локальних налаштувань (мова, країна, часовий пояс) користувача;
* MultipartResolver: забезпечує Upload — завантаження на сервер локальних файлів клієнта.

Spring MVC надає розробнику наступні можливості:

* ясний і прозорий поділ між шарами в MVC і запитах;
* стратегія інтерфейсів — кожен інтерфейс робить тільки свою частину роботи;
* інтерфейс завжди може бути замінений альтернативною реалізацією;
* інтерфейси тісно пов'язані з Servlet API;
* високий рівень абстракції для веб-застосувань.

Spring надає свій шар доступу до баз даних і підтримує всі популярні бази даних. [JDBC](https://ru.wikipedia.org/wiki/JDBC), [iBatis](https://ru.wikipedia.org/w/index.php?title=IBATIS&action=edit&redlink=1)/[MyBatis](https://ru.wikipedia.org/w/index.php?title=MyBatis&action=edit&redlink=1), [Hibernate](https://ru.wikipedia.org/wiki/Hibernate_(%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0)), [JDO](https://ru.wikipedia.org/w/index.php?title=Java_Data_Objects&action=edit&redlink=1), [JPA](https://ru.wikipedia.org/wiki/Java_Persistence_API), [Oracle TopLink](https://ru.wikipedia.org/w/index.php?title=TopLink&action=edit&redlink=1), [Apache OJB](https://ru.wikipedia.org/w/index.php?title=Ojb&action=edit&redlink=1), [Apache Cayenne](https://ru.wikipedia.org/w/index.php?title=Apache_Cayenne&action=edit&redlink=1) тощо.

Для всіх цих фреймворків Spring надає такі особливості:

* управління ресурсами — автоматичне отримання і звільнення ресурсів бази даних;
* обробка виключень — перевід виключень при доступі до даних в виключення Spring;
* транзакційність — прозорі транзакції в операціях з даними;
* розпаковка ресурсів — отримання об’єктів бази даних із пулу з’єднання;
* абстракція для обробки BLOB і CLOB.

Анотація @Controller слугує для того щоб повідомити Spring про те, що даний клас є bean’ом і його необхідно підгрузити при запуску за стосунку.

Анотація @RequestMapping(value = “url”) повідомляє, що даний контролер буде обробляти запит, URl якого “url”.

Анотація @PathVariable(value = “name”) використовується в параметрах методу і вказує на то, що даний параметр отримується із адресної строки. Ім’я змінної не обов’язково повинне співпадати з тим як вона визначена в @RequestMapping, тому вказується value. Це дозволяє явним чином використовувати декілька параметрів в одній адресній строці, наприклад, @RequestMapping(value = “users/{userId}/pages/{pageId}”) [8].

Анотація @RequestParam(“name”) транслює значення параметру запиту в змінну. Так само як і у випадку із @PathVariable, немає необхідності піклуватись про тип, Spring приведе змінну до потрібного типу сам.

Анотація @ModelAttribute(“name”) зв’язує параметр методу або значення, що повертається методом, з атрибутом моделі, яка буде використовуватись при виводі jsp-сторінки [9].

Анотація @Autowired дозволяє автоматично встановити значення поля [10].

Для збереження об’єктів в сесії в Spring MVC передбачена спеціальна анотація @SessionAttributes. Вона декларує атрибути сесії, що використовуються певним обробником. В анотації @SessionAttributes перед класом вказується які об’єкти будуть зберігатися в сесії. Наприклад, @SessionAttributes(“user”) [11].

Взаємодію компонентів Spring MVC можна побачити на рисунку 3.1.

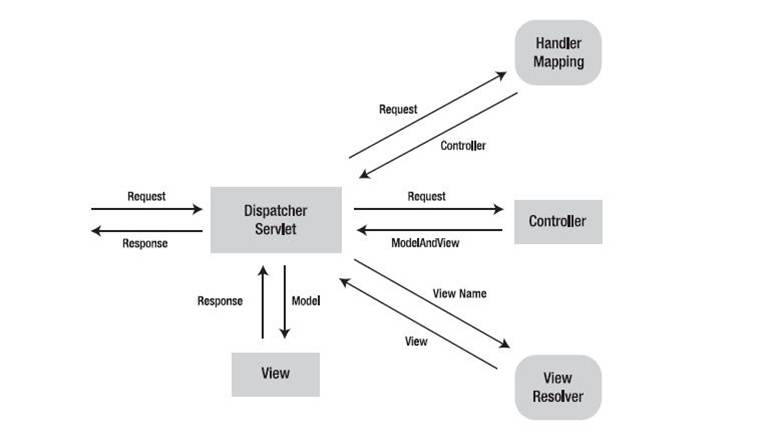


Рисунок 3.1 — Взаємодія компонентів Spring MVC

Спочатку Dispatcher Servlet получає запит, потім у відповідності до своїх налаштувань, визначає який контролер використовувати (Handler Mapping). Після отримання імені контролера запит потрапляє до нього. Там відбувається обробка запиту і назад посилається модель (самі дані) і представлення(як відображати дані). Dispatcher Servlet на основі отриманої назви View шукає представлення, що йому потрібно (View Resolver). В представлення відправляється Model (дані) і назад, якщо потрібно, відповідь від представлення.

Для роботи із базою даних було обрано фреймворк Hibernate 3, що надає набір класів для доступу до даних, а також чудово поєднується разом із фреймворком Spring.

Фреймворк Hibernate — засіб відображення між [об'єктами](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та [реляційними структурами](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) (object-relational mapping, [ORM](https://uk.wikipedia.org/wiki/ORM)) для платформи [Java](https://uk.wikipedia.org/wiki/Java). Hibernate є [вільним](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) програмним забезпеченням, яке поширюється на умовах [GNU Lesser General Public License](https://uk.wikipedia.org/wiki/GNU_Lesser_General_Public_License). Hibernate надає легкий для використання каркас ([фреймворк](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)) для відображення між об'єктно-орієнтованою моделлю даних і традиційною [реляційною базою даних](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) [4].

Метою Hibernate є звільнення розробника від значних типових завдань із програмування взаємодії з [базою даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85). Розробник може використовувати Hibernate як при розробці з нуля, так і для вже існуючої бази даних.

Hibernate піклується про зв'язок [класів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) з таблицями бази даних (і [типів даних](https://uk.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) мови програмування із типами даних [SQL](https://uk.wikipedia.org/wiki/SQL)), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і [JDBC](https://uk.wikipedia.org/wiki/JDBC) коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації [об'єктів](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) і забезпечення сумісності із різними базами даних.

Hibernate забезпечує прозору підтримку збереження даних, тобто їхньої [персистентності](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BD%D1%82%D0%BD%D1%96%D1%81%D1%82%D1%8C) ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) persistence) для [“POJO](https://uk.wikipedia.org/w/index.php?title=Plain_Old_Java_Object&action=edit&redlink=1)”-об'єктів, себто для звичайних Java-об'єктів; єдина сувора вимога до класу, що зберігається — [конструктор](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) за замовчанням

Mapping (зіставлення, буквально — картування) Java класів з таблицями бази даних здійснюється за допомогою конфігураційних [XML](https://uk.wikipedia.org/wiki/XML) файлів або [Java анотацій](https://uk.wikipedia.org/w/index.php?title=Java_%D0%B0%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D1%96%D1%97&action=edit&redlink=1). При використанні файлу XML, Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання (persistent). У цьому немає необхідності, якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних.

Забезпечуються можливості з організації відношення між класами “один-до-багатьох” і “багато-до-багатьох”. На додаток до управління зв'язками між об'єктами, Hibernate також може керувати рефлексивними асоціаціями, де об'єкт має зв'язок “один-до-багатьох” з іншими примірниками свого власного типу даних.

Hibernate забезпечує прозоре збереження POJO ([Plain Old Java Objects](https://uk.wikipedia.org/wiki/Plain_Old_Java_Objects) — простих старих об'єктів Java). Єдина сувора вимога для персистентного класу — [конструктор](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) без аргументів, не обов'язково публічний. Для правильної поведінки деяких програм також потрібна особлива увага до методів equals() і hashCode().

Колекції об'єктів даних, як правило, зберігаються у вигляді колекцій Java-об'єктів, таких як набір (Set) і список (List). Підтримуються [узагальнені класи](https://uk.wikipedia.org/wiki/%D0%A3%D0%B7%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (Generics), введені в Java 5. Hibernate може бути налаштований на “ледачі” (відкладені) завантаження колекцій. Відкладені завантаження є варіантом за замовчуванням, починаючи з Hibernate 3.

Зв'язані об'єкти можуть бути налаштовані на каскадні операції. Наприклад, батьківський клас, Album (музичнй альбом), може бути налаштований на каскадне збереження і/або видалення свого нащадка Track. Це може скоротити час розробки і забезпечити [цілісність](https://uk.wikipedia.org/wiki/%D0%A6%D1%96%D0%BB%D1%96%D1%81%D0%BD%D1%96%D1%81%D1%82%D1%8C). Функція перевірки зміни даних (dirty checking) дозволяє уникнути непотрібного запису дій в базу даних, виконуючи SQL оновлення тільки при зміні полів персистентних об'єктів.

Hibernate забезпечує використання [SQL](https://uk.wikipedia.org/wiki/SQL)-подібної мови [Hibernate Query Language](https://uk.wikipedia.org/w/index.php?title=Hibernate_Query_Language&action=edit&redlink=1) (HQL), яка дозволяє виконувати SQL-подібні запити, записані поряд з об'єктами даних Hibernate. Запити критеріїв надаються як об'єктно-орієнтована альтернатива до HQL.

Hibernate може використовуватись як у самостійних програмах [Java](https://uk.wikipedia.org/wiki/Java), так і в програмах [Java EE](https://uk.wikipedia.org/wiki/Java_EE), що виконуються на сервері (наприклад, [сервлети](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82) чи [EJB](https://uk.wikipedia.org/w/index.php?title=EJB&action=edit&redlink=1) session beans). Також він може включатись як додаткова можливість до інших мов програмування. Наприклад, [Adobe](https://uk.wikipedia.org/wiki/Adobe_Systems) інтегрував Hibernate у дев'яту версію [ColdFusion](https://uk.wikipedia.org/wiki/ColdFusion) (що запускається на серверах з підтримкою додатків [J2EE](https://uk.wikipedia.org/wiki/J2EE)) з рівнем абстракції нових функцій і синтаксису, доданих до [CFML](https://uk.wikipedia.org/w/index.php?title=CFML&action=edit&redlink=1).

Архітектуру Hibernate можна зрозуміти за допомогою рисунку 3.2.



Рисунок 3.2 — Архітектура Hibernate

Java Application — сукупність всіх java файлів і інших ресурсів.

Persistent Object — короткоживучі одно поточні об’єкти, що містять постійний стан і бізнес-функції. Зазвичай це прості Java POJOs (Plain Old Java Object). Вони зв’язані тільки з однією сесією.

Session Factory — потокобезпечний, незмінний кеш скомпільованих відображень для однієї бази даних.

Session — однопоточкий короткоживучий об’єкт, що представляє взаємодію між додатком і постійною пам’яттю. Він слугує обгорткою для JDBC з’єднання і є фабрикою для Transaction.

Transaction — однопоточний, короткоживучий об’єкт, що використовується додатком для вказівки atomic змінних роботи.

**3.2. Середовище розробки IntelliJ IDEA та сервер Apache Tomcat**

Для розробки проекту було обрано сучасне середовище розробки IntelliJ IDEA 14. Цей вибір зумовлений тим, що даний програмний продукт надає повний пакет інструментів необхідних для вирішення поставлених задач, а також має зручний і зрозумілий інтерфейс.

IntelliJ IDEA — комерційне [інтегроване середовище розробки](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8) для різних мов програмування ([Java](https://uk.wikipedia.org/wiki/Java), [Python](https://uk.wikipedia.org/wiki/Python), [Scala](https://uk.wikipedia.org/wiki/Scala), [PHP](https://uk.wikipedia.org/wiki/PHP) та ін.) від компанії [JetBrains](https://uk.wikipedia.org/wiki/JetBrains). Система поставляється у вигляді урізаної по функціональності [безкоштовної](https://uk.wikipedia.org/wiki/%D0%91%D0%B5%D0%B7%D0%BF%D0%BB%D0%B0%D1%82%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) версії «Community Edition» і повнофункціональної [комерційної](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B5%D1%80%D1%86%D1%96%D0%B9%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. [Первинні тексти](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D1%86%D0%B5%D0%B2%D1%96_%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%B8" \o "Сирцеві тексти) Community-версії поширюються рамках [ліцензії Apache 2.0](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%86%D0%B5%D0%BD%D0%B7%D1%96%D1%8F_Apache). Двійкові збірки підготовлені для [Linux](https://uk.wikipedia.org/wiki/Linux), [Mac OS X](https://uk.wikipedia.org/wiki/Mac_OS_X) і [Windows](https://uk.wikipedia.org/wiki/Windows) [5].

Середовище розробки IntelliJ IDEA надає наступні можливості:

* розумне авто доповнення, інструменти для аналізу якості коду, зручна навігація, розширені ре факторинги і форматування Java, Groovy, Scala, HTML, CSS, JavaScript, CoffeeScript, ActionScript, LESS, XML і багато інших мов;
* підтримка всіх популярних фреймворків і платформ, включаючи Java EE, Spring Framework, Grails, Play Framework, GWT, Struts, Node.js, AngularJS, Android, Flex, AIR Mobile і багато інших;
* інтеграція з серверами за стосунків, включаючи Tomcat, TomEE, GlassFish, JBoss, WebLogic, WebSphere, Geronimo, Resin, Jetty і Virgo;
* інструменти для роботи з базами даних і SQL файлами, включаючи зручний клієнт і редактор для схеми бази даних;
* інтеграція з комерційними системами управління версіями Perforce, Team Foundation Server, ClearCase, Visual SourceSafe;
* інструменти для запуску тестів і аналізу покриття коду, включаючи підтримку всіх популярних фреймворків для тестування.

Apache Tomca — контейнер [сервлетів](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82), розроблений [Apache Software Foundation](https://uk.wikipedia.org/wiki/Apache_Software_Foundation). Повністю написаний мовою програмування [Java](https://uk.wikipedia.org/wiki/Java) та реалізує специфікацію [сервлетів](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82) і [Java Server Pages](https://uk.wikipedia.org/wiki/Java_Server_Pages) від [Sun Microsystems](https://uk.wikipedia.org/wiki/Sun_Microsystems), що є стандартами для розробки [веб-застосунків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) на [Java](https://uk.wikipedia.org/wiki/Java) [14].

Члени ASF і незалежні добровольці розвивають та підтримують Tomcat. Користувачі мають вільний доступ до вихідного коду Tomcat за умовами [Apache License](https://uk.wikipedia.org/wiki/Apache_License). Першою релізною версією Tomcat стала версія 3.0.x (попередні версії були випущені Sun для внутрішнього користування).

Tomcat використовується в якості самостійного веб-сервера, в якості сервера контенту в поєднанні з веб-сервером Apache HTTP Server, а також в якості контейнера сервлетів в серверах застосунків JBoss і GlassFish.

Tomcat версії 4.х був випущений з Jasper (перепроектований механізм JSP), Catalina (перепроектований контейнер сервлетів) та Coyote (стек HTTP).

Catalina це контейнер сервлетів Tomcat, який реалізує специфікацію сервлетів Servlet API. Servlet API є основою для всіх інших технологій Java, що стосуються Web і дає можливість динамічно генерувати любий web-контент, використовуючи любі бібліотеки, доступні для Java.

Coyote це компонент стека HTTP Tomcat, який підтримує протокол HTTP 1.1 для веб-сервлетів або контейнеру застосунку. Coyote прослуховує вхідні з’єднання на певному TCP порту сервера, пересилає запити в механізм Tomcat для обробки запитів і відправляє відповідь назад клієнту, що запитує.

Jasper це механізм JSP Tomcat. Він аналізує JSP-файли, щоб компілювати їх в Java код як сервлети (які можуть бути оброблені за допомогою Catalina). Під час виконання Jasper може автоматично виявляти зміни JSP-файлу і перекомпільовувати його.

Для розробки було вибрано версію Tomcat 8.0.26, що підтримує всі попередні.

**3.3. Огляд технології JSP**

Для динамічного формування веб-сторінок, формування форм та таблиці було використано технологію JSP. Ця технологія надає можливість формувати логіку відображення даних, а саме списків та таблиць, форм. Наприклад, JSP має вбудований цикл forEach, що дозволяє вивести всі елементи списку, що був переданий в модель.

JSP (Java Server Pages) — технологія, що дозволяє веб-розробникам динамічно генерувати [HTML](https://uk.wikipedia.org/wiki/HTML), [XML](https://uk.wikipedia.org/wiki/XML) та інші веб-сторінки. Згодом JSP було включено у складі [Java EE](https://uk.wikipedia.org/wiki/Java_EE) — програмної платформи для програмування веб-додатків. Технологія дозволяє вставляти [Java](https://uk.wikipedia.org/wiki/Java)-код, в статичний вміст сторінки. Також можуть використовуватись бібліотеки JSP тегів для вставки їх в JSP-сторінки. Сторінки компілюються JSP-компілятором в [сервлети](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82), які є Java-класами, і виконуються на сервері. Сервлети також можуть бути написані розробником, не використовуючи JSP-сторінки. Ці технології доповнюють одна одну [15].

JSP — одна із високопродуктивних технологій, оскільки весь код сторінки транслюється в java-код сервлету за допомогою компілятора JSP сторінок (напр. [Jasper](https://uk.wikipedia.org/w/index.php?title=Jasper&action=edit&redlink=1" \o "Jasper (ще не написана))), а потім компілюється в [байт-код](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) віртуальної машини [J](https://uk.wikipedia.org/wiki/Java)ava ([JVM](https://uk.wikipedia.org/wiki/JVM)).

Специфікація JSP розрізняє три типи скрипкових елементів:

* об’явлення “<%!” одна або декілька декларацій “%>”;
* вирази “<%=” один вираз “%>”;
* скриплети “<%” скрип лет “%>”.

В JSP сторінках елементи записуються за допомогою знаків (умовних знаків, міток, марок). Множину знаків, які JSP контейнер може тлумачити, можна розширити за допомогою так званих бібліотек знаків. Також до розширеної множини знаків можна приєднати дії, внаслідок чого відбувається розширення самої мови JSP. Дії можна розділити на стандартні і власні. Загальний спосіб запису може виглядати так: <%@ taglib uri=”URI до бібліотеки знаків” prefix=” префікс знаку” >. Бібліотеку знаків необхідно ідентифікувати за допомогою URI адреси.

Починаючи з версії 2.0, Spring надає повний набір даних для обробки елементів форми при використанні JSP і Spring Web MVC. Кожен тег забезпечує підтримку набору атрибутів його аналогу в HTML, що робить їх знайомими та інтуїтивно зрозумілими. Бібліотека форм поставляється в spring-webmvc.jar. Рядок підключення виглядає наступним чином: <%@ taglib prefix=”form” uri=“<http://www.springframework.org/tags/form>” %>. Де form префікс імені тегу [16].

**3.4. Огляд засобів розробки клієнтської частини**

Для розробки користувацького інтерфейсу у вигляді веб-сторінок було використано мову розмітки HTML([англ.](http://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) HyperText Markup Language — Мова розмітки [гіпертекстових](http://uk.wikipedia.org/wiki/%D0%93%D1%96%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82) документів) — стандартна [мова розмітки](http://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D1%80%D0%BE%D0%B7%D0%BC%D1%96%D1%82%D0%BA%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) [веб-сторінок](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0) в [Інтернеті](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Більшість [веб-сторінок](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0) створюються за допомогою мови HTML (або [XHTML](http://uk.wikipedia.org/wiki/XHTML)). Документ HTML оброблюється [браузером](http://uk.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) та відтворюється на екрані у звичному для людини вигляді.

HTML є похідною мовою від [SGML](http://uk.wikipedia.org/wiki/SGML), успадкувавши від неї [визначення типу документу](http://uk.wikipedia.org/wiki/Document_Type_Definition) та ідеологію структурної розмітки тексту.

HTML разом із [каскадними таблицями стилів](http://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%BA%D0%B0%D0%B4%D0%BD%D1%96_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%96_%D1%81%D1%82%D0%B8%D0%BB%D1%96%D0%B2) та [вбудованими скриптами](http://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) — це три основні технології побудови [веб-сторінок](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0).

HTML впроваджує засоби для:

* створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
* отримання інформації із Всесвітньої мережі через [гіперпосилання](http://uk.wikipedia.org/wiki/%D0%93%D1%96%D0%BF%D0%B5%D1%80%D0%BF%D0%BE%D1%81%D0%B8%D0%BB%D0%B0%D0%BD%D0%BD%D1%8F);
* створення інтерактивних форм;
* включення зображень, звуку, відео, та інших об’єктів до тексту.

Для поліпшення взаємодії, [SGML](http://uk.wikipedia.org/wiki/SGML) вимагає щоб кожна похідна мова (HTML у тому числі) визначала свою [кодову таблицю](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D1%81%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D1%96%D0%B2) для кожного документа, яка складається з репертуара (перелік різноманітних символів) та позиції символа (перелік цифрових посилань на символи з репертуара). Кожен документ HTML — це послідовність символів з репертуара.

Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа.

Для написання алгоритму формування протоколу ендоскопічного обстеження було використано скриптову мову програмування JavaScript

JavaScript — назва реалізації стандарту мови програмування ECMAScript компанії Netscape. Вибір зумовлено тим, що розроблений алгоритм повинен працювати на стороні клієнта, а JavaScript є насамперед клієнтською мовою програмування. Також він надає можливість асинхронного зв’язку із сервером, що необхідно при даній реалізації.

Мова JavaScript являє собою мову сценаріїв для Web. На сьогодні підтримується усіма браузерами. JavaScript є підмножиною мови Java, але не є частиною мови Java. Корпорація Sun не підтримує мову JavaScript. Через свою обмеженість JavaScript не може бути використана для програмування багатьох складних функцій [17].

На відміну від аплетів Java, які принципово відрізняються від ресурсів HTML і повинні динамічно завантажуватися при звертанні до сторінки web, сценарій JavaScript включаються в сторінку HTML за допомогою дескриптора та інтерпретування браузером в режимі реального часу. Крім того, JavaScript можливо застосовувати для створення серверних застосувань.

В якості попередників JavaScript можливо розглянути ряд невеликих прототипно-орієнтованих мов, наприклад, HyperTalk, dBase та LiveScript. На таких мовах, завдяки їх синтаксичній простоті, вмонтованій функціональній простоті створення об’єктів може програмувати навіть початківець. JavaScript дозволяє вносити до web-сторінки інтерактивність, забезпечує взаємодію з користувачем, підтримує заповнення форм введення та переміщення по web-документу. Деякі потужні типи систем інтерактивної взаємодії вдається реалізувати за рахунок комбінації можливостей JavaScript з іншими властивостями web-сторінок, наприклад, роботи з фреймами та вмонтованими додатками. JavaScript стала новим відкритим стандартом мови сценаріїв Internet, яка підтримується багатьма компаніями.

При використанні в рамках технології [DHTML](https://uk.wikipedia.org/wiki/DHTML) JavaScript код включається в [HTML](https://uk.wikipedia.org/wiki/HTML)-код сторінки і виконується [інтерпретатором](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80), вбудованим в [браузер](https://uk.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80). Код JavaScript вставляється в теги <script></script> з обов'язковим по специфікації HTML 4.01 атрибутом type=“text/javascript”, хоча в більшості браузерів мова сценаріїв за умовчанням саме JavaScript.

Структурно JavaScript можна представити у вигляді трьох чітко розділених одна від одної частин:

* ядро;
* об’єктна модель браузера;
* об’єктна модель документа.

Об'єктна модель браузера — браузер-специфічна частина мови, що є прошарком між ядром і об'єктною моделлю документа. Основне призначення об'єктної моделі браузера — управління вікнами браузера і забезпечення їх взаємодії. Кожне з вікон браузера представляється об'єктом window, центральним об'єктом DOM. Об'єктна модель браузера на даний момент не стандартизована, однак специфікація знаходиться в розробці WHATWG і W3C.

Об'єктна модель документа — інтерфейс програмування додатків для HTML і XML-документів. Згідно DOM, документ (наприклад, веб-сторінка) може бути представлений у вигляді дерева об'єктів, що володіють рядом властивостей, які дозволяють виробляти з ним різні маніпуляції:

* генерація і додавання вузлів;
* отримання вузлів;
* зміна вузлів;
* зміна зв'язків між вузлами;
* видалення вузлів.

JavaScript використовується в клієнтській частини веб-застосунків: клієнт-серверних програм, в якому клієнтом є браузер, а сервером — веб-сервер, що мають розподілену між сервером і клієнтом логіку. Обмін інформацією в веб-застосутках відбувається по мережі. Одним з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-застосунки є кросплатформенними сервісами.

JavaScript використовується в AJAX, популярному підході до побудови призначених для користувача інтерфейсів веб-застосунків, що полягає в «фоновому» асинхронному обміні даними браузера з веб-сервером. В результаті, при оновленні даних веб-сторінка не перезавантажується повністю та інтерфейс веб-застосунку стає швидшим, ніж це відбувається при традиційному підході (без застосування AJAX) [18].

Для зручної роботи з DOM при написанні алгоритму формування протоколу ендоскопічного обстеження було використано JavaScript бібліотеку jQuery.

jQuery — популярна [JavaScript](https://uk.wikipedia.org/wiki/JavaScript) бібліотека з [відкритим кодом](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%BA%D1%80%D0%B8%D1%82%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F). Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, jQuery використовується понад половиною від мільйона найбільш відвідуваних [сайтів](https://uk.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82). jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день [19].

jQuery є [вільним програмним забезпеченням](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) під [ліцензією MIT](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%86%D0%B5%D0%BD%D0%B7%D1%96%D1%8F_MIT). Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів [DOM](https://uk.wikipedia.org/wiki/DOM), створенню анімації, обробки подій, і розробки [AJAX](https://uk.wikipedia.org/wiki/AJAX)-[застосунків](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA). jQuery також надає можливості для розробників, для створення [плагінів](https://uk.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D1%96%D0%BD) у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних [веб-сторінок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0).

Основне завдання jQuery — це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації [DOM](https://uk.wikipedia.org/wiki/DOM) об'єктів за допомогою [CSS](https://uk.wikipedia.org/wiki/CSS) та [XPath](https://uk.wikipedia.org/wiki/XPath) селекторів. Також даний фреймворк надає інтерфейси для [Ajax](https://uk.wikipedia.org/wiki/Ajax)-[застосунків](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), обробників подій і простої анімації.

Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів.

Можливості jQuery:

* движок кросбраузерних CSS-селекторів, що виділився в окремий проект;
* перехід по дереву DOM, включаючи підтримку XPath як плагіна;
* події;
* візуальні ефекти;
* AJAX доповнення;
* JavaScript плагіни.

jQuery має обгортку для здійснення AJAX запиту. Це низькорівневий метод, що має велику кількість налаштувань. Приклад запиту: jQuery.ajax(settings), де settings — параметр в якому можна задати налаштування для даного запиту. Задається за допомогою об’єкта. Жодна із налаштувань не є обов’язковою.

Для досягнення однаковості, та зручного інтуїтивно зрозумілого користувацького інтерфейсу, було використано фреймворк Bootsrtap.

Bootstrap — це безкоштовний набір інструментів з відкритим [первинним кодом](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D1%86%D0%B5%D0%B2%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4), призначений для створення [веб-сайтів](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82) та [веб-застосунків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), який містить [шаблони](https://uk.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D1%81%D0%B0%D0%B9%D1%82%D1%83) [CSS](https://uk.wikipedia.org/wiki/CSS) та [HTML](https://uk.wikipedia.org/wiki/HTML) для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення [JavaScript](https://uk.wikipedia.org/wiki/JavaScript). Він спрощує розробку динамічних веб-сайтів і веб-застосунків [20].

Bootstrap — це клієнтський [фреймворк](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D0%BA%D0%B0%D1%80%D0%BA%D0%B0%D1%81), тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на [сервері](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80). [Репозиторій](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D1%96%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F" \o "Репозиторій програмного забезпечення) з даним фреймворком є одним з найбільш популярних на [GitHub](https://uk.wikipedia.org/wiki/GitHub).

Bootstrap (початкова назва — Twitter Blueprint) був розроблений Марком Отто та Джейкобом Торнтоном ([Twitter](https://uk.wikipedia.org/wiki/%D0%A2%D0%B2%D1%96%D1%82%D1%82%D0%B5%D1%80)) у якості фреймворку для забезпечення однаковості внутрішніх інструментів Twitter. До появи Bootstrap у розробці [інтерфейсу](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0) застосовувалися різні [бібліотеки](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC), що призводило до появи суперечностей та ускладнювало [супровід](https://uk.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D1%80%D0%BE%D0%B2%D1%96%D0%B4_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F).

Bootstrap сумісний з останніми версіями [браузерів](https://uk.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) [Google Chrome](https://uk.wikipedia.org/wiki/Google_Chrome), [Firefox](https://uk.wikipedia.org/wiki/Mozilla_Firefox), [Internet Explorer](https://uk.wikipedia.org/wiki/Internet_Explorer), [Opera](https://uk.wikipedia.org/wiki/Opera) і [Safari](https://uk.wikipedia.org/wiki/Safari).

Bootstrap має модульну структуру і складається переважно з наборів [таблиць стилів](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%BA%D0%B0%D0%B4%D0%BD%D1%96_%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%96_%D1%81%D1%82%D0%B8%D0%BB%D1%96%D0%B2) [LESS](https://uk.wikipedia.org/wiki/LESS_(%D0%BC%D0%BE%D0%B2%D0%B0_%D1%81%D1%82%D0%B8%D0%BB%D1%96%D0%B2)), які реалізують різні компоненти цього набору інструментів. Розробники можуть самостійно налаштовувати файли Bootstrap, обираючи компоненти для свого проекту.

Основні інструменти Bootstrap:

* сітки (grid) — наперед задані, готові до використання колонки;
* шаблони (template) — фіксовані чи адаптивні шаблони сторінок;
* типографіка (typography) — опис та визначення класів для шрифтів, таких як шрифти для коду, цитат тощо;
* мультимедіа (media) — засоби управління зображеннями та відео;
* таблиці (table) — засоби оформлення таблиць, які зокрема забезпечують сортування;
* форми (form) — класи для оформлення як форм, так і деяких подій;
* навігація (nav, navbar) — класи для оформлення вкладок, сторінок, меню і панелей навігації;
* сповіщення (alert) — класи для оформлення діалогових вікон, підказок і спливаючих вікон;
* іконочний шрифт (icon font) — набір іконок у вигляді шрифту, складається майже з 500 компонентів.

**4. ОПИС ПРОГРАМНОї РЕАЛіЗАЦії системи**

Оскільки дана система призначена для роботи з багатьма користувачами, вона виконана у вигляді веб-сайту. Користувач через браузер взаємодіє з серверною частиною для досягнення необхідних цілей. Таким чином клієнтом буде виступати браузер.

Поділ на підсистеми дозволяє змінювати і редагувати систему в залежності від потреб. Також така структура дозволяє розмістити систему територіально віддалено.

**4.1. Структура програмної системи**

Проект було розроблено по шаблону MVC, тому його структура буде також йому відповідати (рисунок 4.1). Як видно на рисунку, java класи розділені на пакети: контролери, домени, репозиторії та валідатори. Контролери слугують для обробки HTTP запиту та відправки відповіді клієнту. Домени це POJO класи, що зіставляються із табличками з бази даних. Репозиторії це спеціальні класи у яких здійснюються дії над базою даних, а саме додавання до таблиць нової інформації, редагування інформації в таблицях, видалення записів та вибірка за певними критеріями. Валідатори призначені для перевірки правильності введених чи відредагованих користувачем даних. Кожен валідатор написаний для певного POJO класу з доменів.

Пакети webapp.resouces.\* містять ресурсні файли, такі як зображення, скрипти, css стилі.

Пакет WEB-INF.pages містить всі JSP сторінки, що використовуються для відображення даних. Пакет WEB-INF.tags містить шаблони для шапки та меню.

XML файли пакету WEB-INF задають налаштування для проекту.

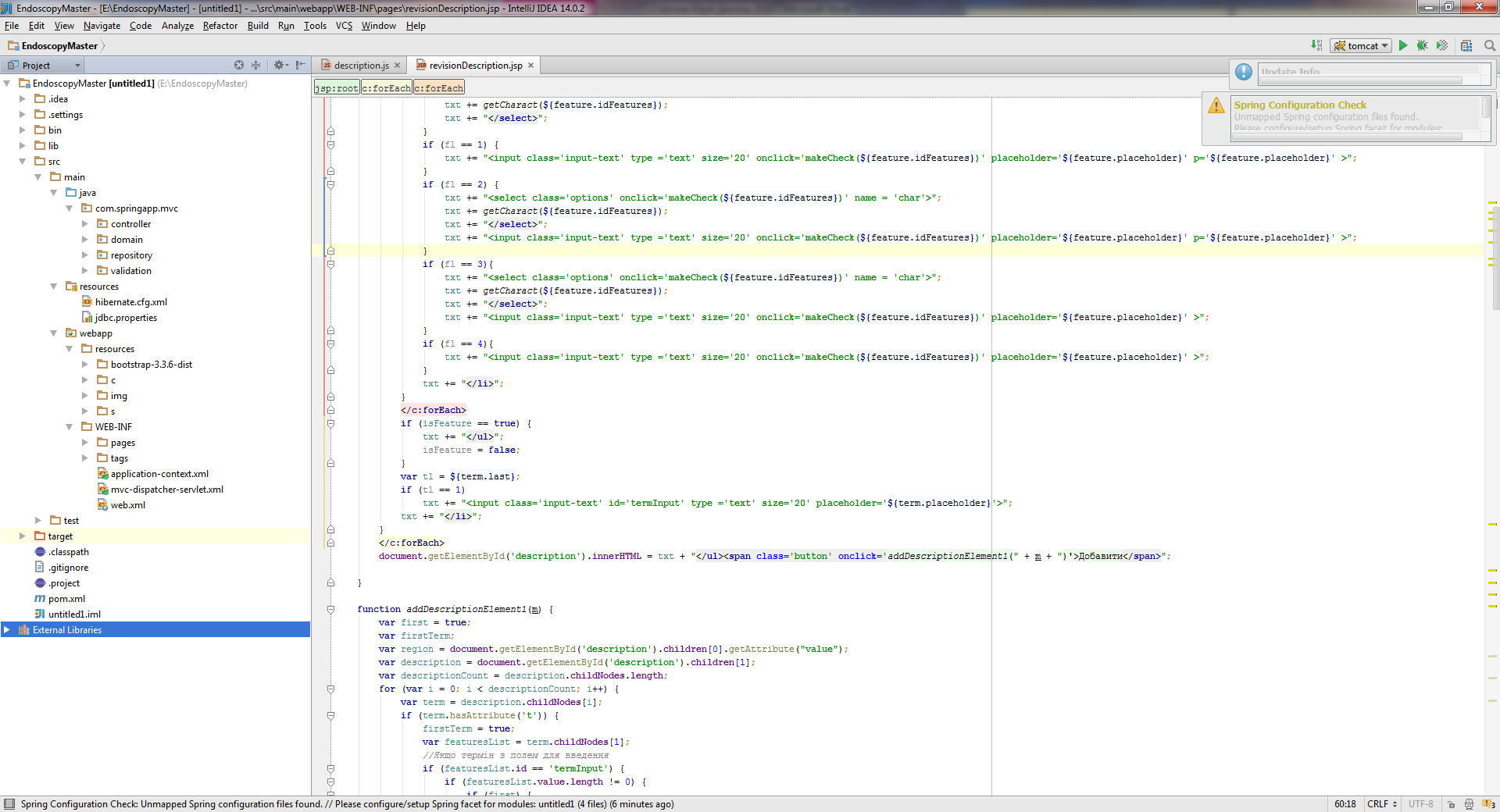


Рисунок 4.1 — Структура програмної системи

**4.2. Опис серверної частини**

Для реалізації серверної частини було використано фреймворки Spring MVC у поєднанні з Hibernate 3. Перший — це програмний каркас ([фреймворк](https://uk.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)) з відкритим кодом та контейнером з підтримкою [інверсії управління](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D1%96%D1%8F_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F) для платформи Java. Другий — фреймворк для роботи із базою даних.

Авторизацію користувача було створено на основі сесії. Для цього після обробки запиту “validate” створюється сесійний атрибут, що перевіряється при зверненні до кожного методу-обробника запитів. Цей сесійний атрибут є об’єктом класу user і містить поле role. Це поле слугує для визначення типу користувача, а саме чи це лікар чи адміністратор. На початку кожного методу-обробника робиться перевірка, чи дозволено користувачу даного типу звертатись до цього методу.

Було створено три класи контролери: AdminController, DoctorController, DataController. Перші для містять в собі методи для обробки запитів від адміністратора і лікаря відповідно. DataController містить всі необхідні методи для роботи із доменами, тобто методи, в яких відбувається проста робота із таблицями бази даних (редагування, видалення, додавання нових записів). Кожен клас контролер позначений анотацією @Controller та містить конструктор, позначений анотацією @Autowired. Кожен метод-обробник має анотацію @RequestMapping, в якій вказується який саме запит цей метод повинен обробляти (рисунок 4.2).

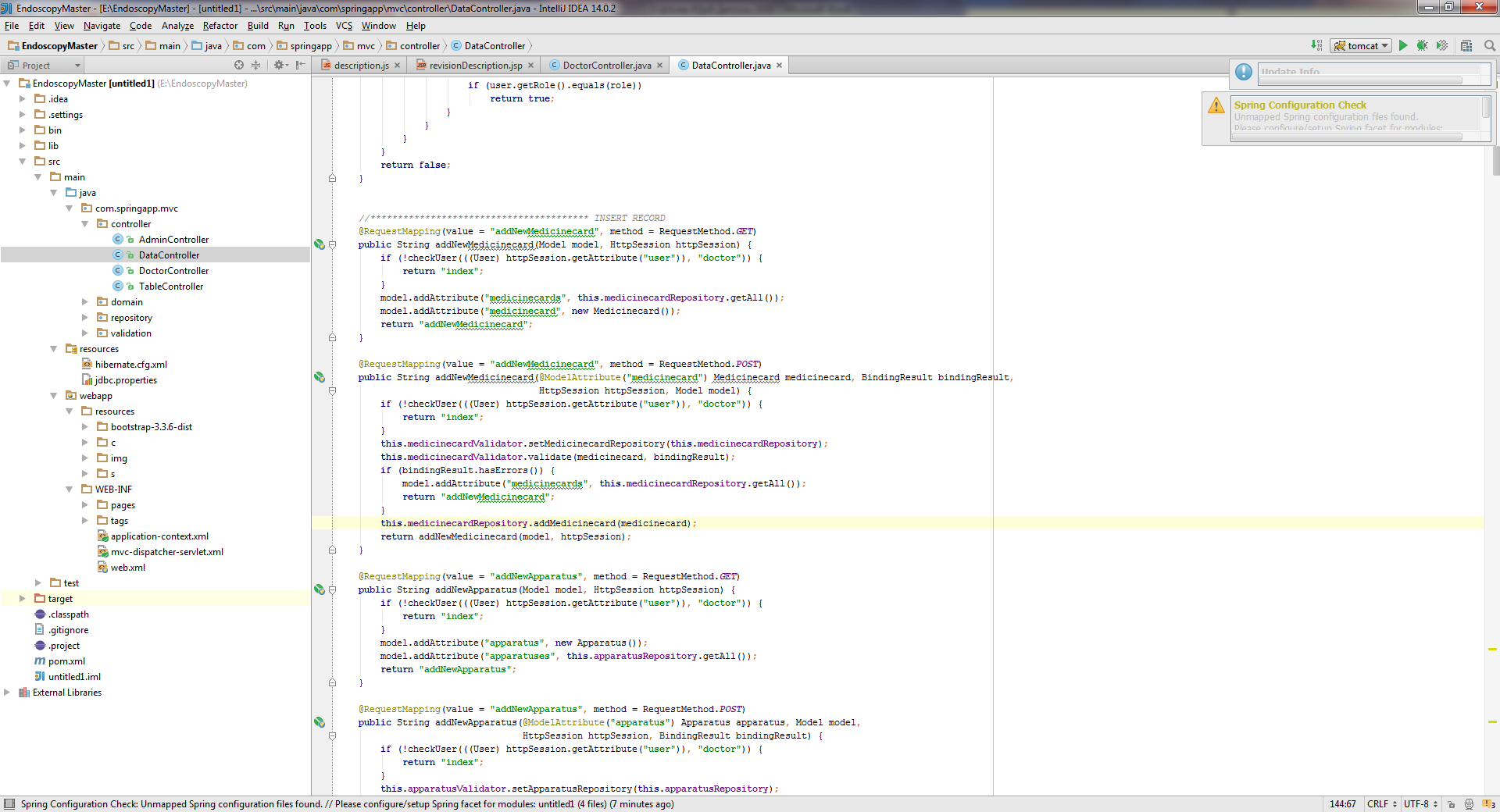


Рисунок 4.2 — Приклад методу-обробника запиту

Кожен клас домен позначений анотацією @Entity. Поля класу мають типи, що ставляться у відповідність типам даних MySQL. Також у цих класах використовуються наступні анотації: @id (вказує, що це поле є первинним ключем), @GeneratedValue (вказує, що це поле є автоінкрементним), @Column (вказує, яке поле в базі даних відповідає даному. Всі класи домени були сформовані засобами IntelliJ IDEA 14.

Кожен клас репозиторій позначається анотаціями @Repository і @Transactional та містить @Autowired об’єкт SessionFactory. Методи цих класів використовують цей об’єкт для отримання сесії і виконують дії з базою даних (рисунок 4.3).

Класи валідатори позначаються анотацією @Component та реалізують інтерфейс Validator. Цей інтерфейс має два методи для визначення supports і validate. Метод validate приймає об’єкт, що необхідно перевірити, та об’єкт, в який записуються помилки при перевірці. Для запису помилок було використано клас BindingResults.

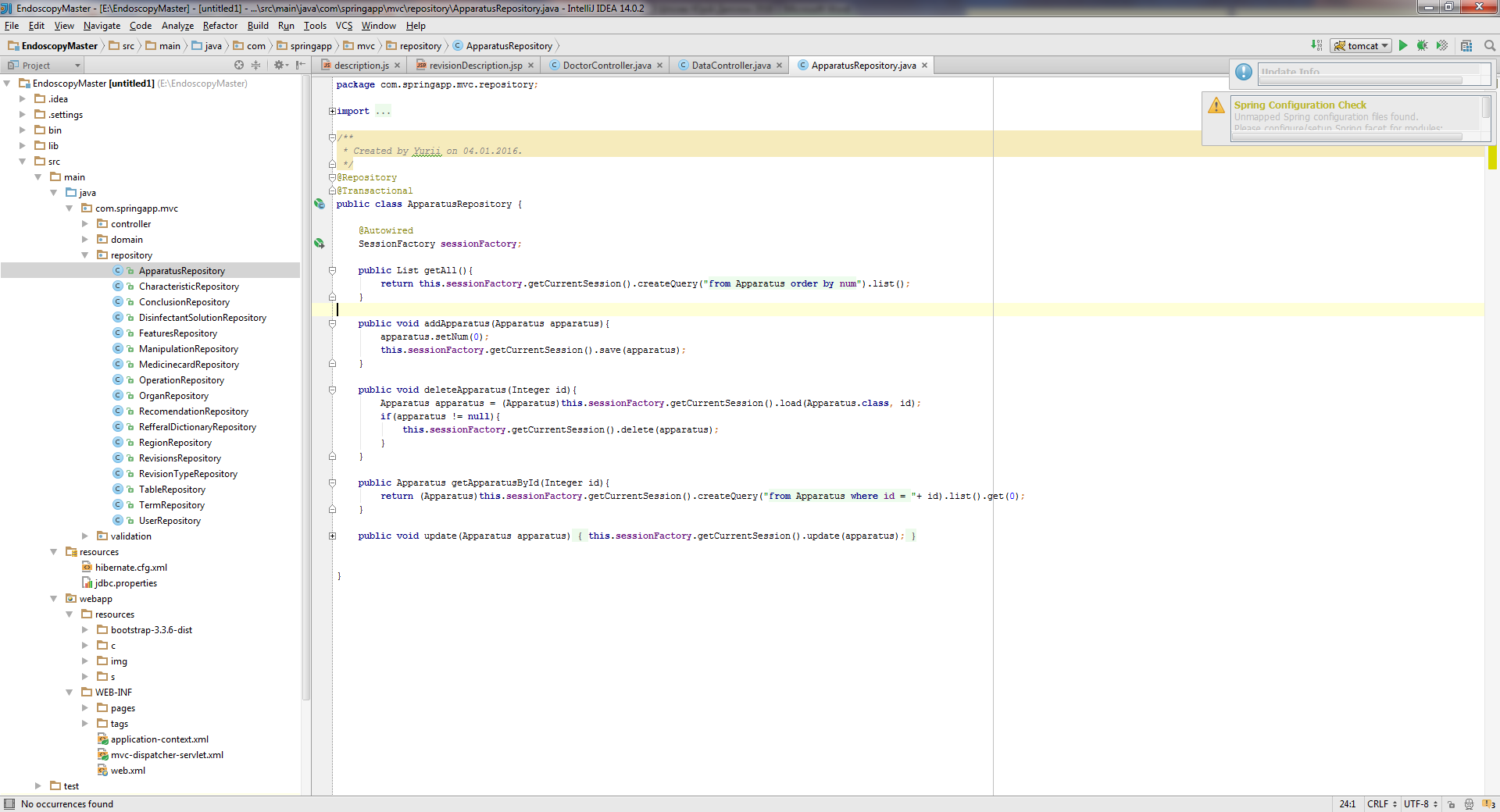


Рисунок 4.3 — Приклад методу класу репозиторію.

У файлі jdbc.properties задається метаінформація для підключення до бази даних. Файл hibernate.cnf.xml містить мапінг для доменів, що були сформовані.

**4.3. Опис бази даних**

Для розробки бази даних було обрано середовище розробки MySQL 5.6. Це остання стабільна версія на момент розробки. Цей програмний продукт надає можливість локально розробити базу даних та протестувати її.

MySQL — [вільна](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) [система керування реляційними базами даних](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D0%BC%D0%B8_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85).

MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на [mSQL](https://uk.wikipedia.org/w/index.php?title=MSQL&action=edit&redlink=1), проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних [веб-сторінок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0), оскільки має чудову підтримку з боку різноманітних [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [13].

MySQL має подвійне ліцензування та може розповсюджуватися відповідно до умов ліцензії [GPL](https://uk.wikipedia.org/wiki/GPL). Але за умовами GPL, якщо якась програма використовує бібліотеки MySQL, то вона теж повинна розповсюджуватися за ліцензією GPL. Проте це може розходитися з планами розробників, які не бажають відкривати [первинні тексти](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D1%86%D0%B5%D0%B2%D1%96_%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%B8) своїх програм. Для таких випадків передбачена комерційна ліцензія компанії [Oracle](https://uk.wikipedia.org/wiki/Oracle_Corporation), яка також забезпечує якісну сервісну підтримку. В разі використання та розповсюдження програмного забезпечення з іншими вільними ліцензіями, такими як [BSD](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%86%D0%B5%D0%BD%D0%B7%D1%96%D1%8F_BSD), [Apache](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%86%D0%B5%D0%BD%D0%B7%D1%96%D1%8F_Apache), [MIT](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%86%D0%B5%D0%BD%D0%B7%D1%96%D1%8F_MIT) та інші, MySQL дозволяє використання бібліотек MySQL за ліцензією GPL.

MySQL виникла як спроба застосувати [mSQL](https://uk.wikipedia.org/w/index.php?title=MSQL&action=edit&redlink=1) до власних розробок компанії: таблиць, для яких використовувалися [ISAM](https://uk.wikipedia.org/w/index.php?title=ISAM&action=edit&redlink=1) — підпрограми низького рівня. У результаті був вироблений новий [SQL](https://uk.wikipedia.org/wiki/SQL)-інтерфейс, але [API](https://uk.wikipedia.org/wiki/API)-інтерфейс залишився в спадок від mSQL.

Логотип MySQL у вигляді дельфіна носить ім'я «Sakila». Він був обраний з великого списку запропонованих користувачами «імен дельфіна». Ім'я «Sakila» було відправлено Open Source-розробником Ambrose Twebaze.

MySQL — компактний [багатопотоковий](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B3%D0%B0%D1%82%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2%D1%96%D1%81%D1%82%D1%8C) [сервер баз даних](https://uk.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85&action=edit&redlink=1). Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. [Первинні коди](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D1%86%D0%B5%D0%B2%D1%96_%D0%BA%D0%BE%D0%B4%D0%B8" \o "Сирцеві коди) [сервера](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80) компілюються на багатьох платформах. Найповніше можливості сервера виявляються в [UNIX](https://uk.wikipedia.org/wiki/UNIX)-системах, де є підтримка багатопотоковості, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

* простота у встановленні та використанні;
* підтримується необмежена кількість користувачів, що одночасно працюють із БД;
* кількість рядків у таблицях може досягати 50 млн;
* висока швидкість виконання команд;
* наявність простої і ефективної системи безпеки.

MySQL версії 5.1 має 20 відомих серйозних помилок в додаток до 35 дефектів версії 5.0. Критичні помилки іноді не виправляються протягом тривалих періодів часу.

MySQL показує низьку ефективність при використанні її як сховища даних, це частково пов'язано з нездатністю використовувати декілька процесорів для обробки одного запиту. До того ж, MySQL часто критикують за те, що вона має розходження зі стандартом SQL щодо трактування NULL значень і значення за замовчуванням.

SQL ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Structured query language — мова структурованих запитів) — [декларативна мова програмування](https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) для взаємодії користувача з [базами даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), що застосовується для формування запитів, оновлення і керування [реляційними](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) базами даних, створення схеми бази даних і її модифікації, системи контролю за доступом до бази даних. Сам по собі SQL не є ні системою керування базами даних, ні окремим програмним продуктом. Не бувши мовою програмування в тому розумінні, як [C](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_C) або [Pascal](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_Pascal), SQL може формувати інтерактивні запити або, бувши вбудованою в прикладні програми, виступати в якості інструкцій для керування даними. Стандарт SQL, крім того, вміщує функції для визначення зміни, перевірки і захисту даних.

SQL — це діалогова мова програмування для здійснення [запиту](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%B7%D0%B0%D0%BF%D0%B8%D1%82) і внесення змін до бази даних, а також управління базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, оновлення, і вилучення даних, використовуючи систему управління і адміністративні функції. SQL також включає CLI (Call Level Interface) для доступу і управління базами даних дистанційно [12].

Критика SQL включає відсутність [крос-платформенності](https://uk.wikipedia.org/w/index.php?title=%D0%9A%D1%80%D0%BE%D1%81-%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D1%96%D1%81%D1%82%D1%8C&action=edit&redlink=1), невідповідною обробкою відсутніх даних, і іноді неоднозначна граматика і [семантика](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0) мови.

Незважаючи на наявність діалектів і відмінностей в [синтаксисі](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), в більшості своїй тексти SQL-запитів, що містять, [DDL](https://uk.wikipedia.org/wiki/DDL) і [DML](https://uk.wikipedia.org/wiki/DML), можуть бути досить легко перенесені з однієї бази давних в іншу. Існують системи, розробники яких спочатку закладалися на застосування щонайменше кількох баз даних (наприклад: система електронного документообігу [Documentum](https://uk.wikipedia.org/w/index.php?title=Documentum&action=edit&redlink=1) може працювати як з [Oracle](https://uk.wikipedia.org/wiki/Oracle_Database), так і з [Microsoft SQL Server](https://uk.wikipedia.org/wiki/Microsoft_SQL_Server)та [IBM DB2](https://uk.wikipedia.org/wiki/IBM_DB2)). Природно, що при застосуванні деяких специфічних для реалізації можливостей такої переносимості добитися вже дуже важко.

Наявність стандартів і набору тестів для виявлення сумісності і відповідності конкретній реалізації SQL загальноприйнятому стандарту тільки сприяє «стабілізації» мови. Правда, варто звернути увагу, що сам по собі стандарт місцями занадто формалізований і роздутий в розмірах, наприклад, Core-частину стандарту SQL: включає понад 1300 сторінок тексту.

За допомогою SQL програміст описує тільки те, які дані потрібно витягнути або модифікувати. Тобто, яким чином це зробити, вирішує система безпосередньо при обробці SQL-запиту. Проте не варто думати, що це повністю універсальний принцип — програміст описує набір даних для вибірки або модифікації, проте йому при цьому корисно уявляти, як база даних розбиратиме текст його запиту. Особливо критичні такі моменти стають при роботі з великими базами даних і зі складними запитами — чим складніше сконструйований запит, тим більше він допускає варіантів написання, різних за швидкістю виконання, але тих самих за набором даних.

Творець реляційної моделі даних [Едгар Кодд](https://uk.wikipedia.org/wiki/%D0%95%D0%B4%D0%B3%D0%B0%D1%80_%D0%9A%D0%BE%D0%B4%D0%B4), [Крістофер Дейт](https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D1%96%D1%81%D1%82%D0%BE%D1%84%D0%B5%D1%80_%D0%94%D0%B5%D0%B9%D1%82) та їхні прихильники указують на те, що SQL не є істинно реляційною мовою. Зокрема вони указують на такі проблеми SQL:

* рядки, що повторюються;
* невизначені значення (null);
* явна вказівка порядку стовпчиків зліва направо;
* стовпці без імені та імена стовпчиків, що дублюються;
* відсутність підтримки властивості “=”;
* використання покажчиків;
* висока надлишковість.

Хоча SQL і замислювався, як засіб роботи кінцевого користувача, врешті-решт він став настільки складним, що перетворився на інструмент програміста.

Незважаючи на наявність міжнародного стандарту ANSI SQL-92, багато компаній (наприклад, [Oracle](https://uk.wikipedia.org/wiki/Oracle_Corporation), [Sybase](https://uk.wikipedia.org/wiki/Sybase), [Microsoft](https://uk.wikipedia.org/wiki/Microsoft), [MySQL](https://uk.wikipedia.org/wiki/MySQL)), що займаються розробкою, вносять зміни до мови SQL, вживаної в розроблених ними системах, тим самим відступаючи від стандарту. Таким чином з'являються специфічні для кожної конкретної системи баз даних діалекти мови SQL.

Центральною сутністю бази даних є табличка revisions (рисунок 4.4) В ній міститься номер обстеження (id), дата (date), опис (description), вперше виявлена патологія (first\_patalogy) та посилання на інші таблиці: user (doctor\_id), revisiontype (revisiontype\_id), apparatus (apparatus\_id), disinfectant\_ solution (disinfectant\_solution\_id), referral\_dictionary (referral\_dictionary\_id), medicinecard1 (medicinecard\_id) . Ця таблиця моделює щоденний журнал [3].

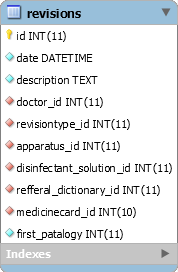


Рисунок 4.4 — Табличка revision

Таблиця medicinecard1 (рисунок 4.5) — моделює електронну медичну карточку і містить наступні поля: номер карточки (id), ПІБ (pib), адреса проживання (adress), дата народження (burnDate), ідентифікаційний код (identifyCode).

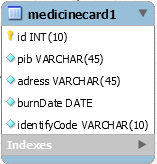


Рисунок 4.5 — Табличка medicinecard1

Таблиця user (рисунок 4.6) — використовується для зберігання інформації про користувачів: ід (id), логін (login), пароль (pass), роль (role), ПІБ (pib), місце роботи (work\_place).

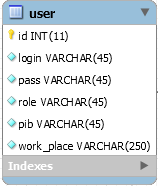


Рисунок 4.6 — Табличка user

Табличка manipulations (рисунок 4.7) — містить в собі записи про всі виконані маніпуляції. Поля: ід (id), значення (value), місце виконання (place). Зв’язки: ід обстеження (revision\_id), ід маніпуляції (manipulation\_dictionary\_id).

Табличка operations (рисунок 4.8) — містить в собі записи про всі виконані операції. Поля: ід (id), опис (description). Зв’язки: ід операції (operation\_dictionary\_id), ід обстеження (revision\_id).

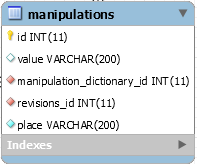


Рисунок 4.7 — Табличка manipulations

Для зручного користування мінімальною стандартною термінологію її було представлено у вигляді табличок в базі даних із зв’язками у вигляді ієрархії (рисунок 4.9) [1]. Верхній рівень займає тип обстеження (revisiontype). Кожен тип обстеження містить органи які відносяться до цього обстеження (organ\_dictionary). Кожен орган поділяється на ділянки (region\_dictionary). Для опису ділянки використовується певний список термінів (term\_dictionary). Кожен терм містить певні ознаки (features\_dictionary), які мають свої варіанти характеристик  (characteristic\_dictionary).

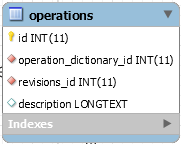


Рисунок 4.8 — Табличка operations

Для автоматичного формування списку заключень і рекомендацій було зроблено прив’язку заключень (conclusions\_dictionary) до конкретних термінів, ознак і характеристик. Таким чином, коли користувач описує певний термін із певною ознакою і характеристикою робиться перевірка, чи є в нього якесь заключення. Якщо є, воно добавляється в табличку заключень (conclusions).

Одне заключення може містити одну чи декілька рекомендацій (recommendation\_dictionary) або не містити їх зовсім.

Таблички term\_dictionary, features\_dictionary, characteristic\_dictionary містять поле placeholder, що виступає підказкою для введення інформації. Наприклад, в поля для введення буде placeholder “ділянка(ки)”. Також значення поля last вказує чи це кінцева гілка ієрархії, і в залежності від цього формує control для інформації (поле для введення чи випадаючий список та їх комбінації).

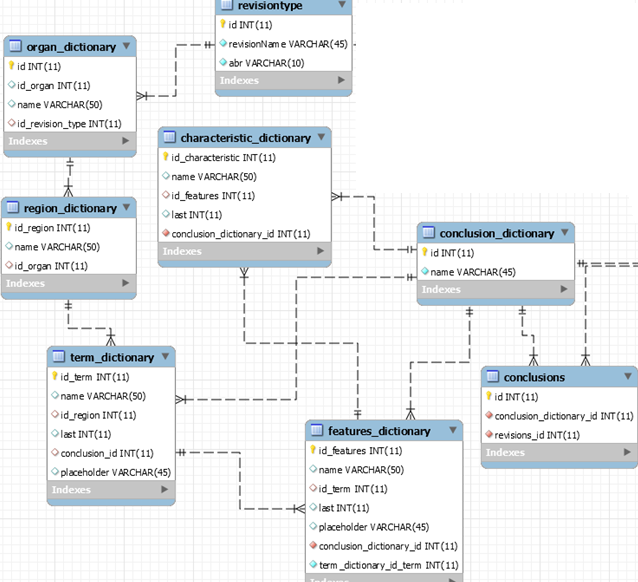


Рисунок 4.9 — Таблиці-словники для формування протоколу ендоскопічного обстеження

**4.4. Опис клієнтської частини**

Для розробки зручного користувацького інтерфейсу було використано HTML 5, CSS 3, Bootstrap 3 та jQuery.

Для адаптації під розміри екрану застосовувались класи bootstrap, що забезпечили читабельність тексту і функціональність на пристроях із різним розміром екрану. jQuery використано для анімацій та для динамічного формування контенту, а також для AJAX запитів.

Для створення зручного, динамічного та функціонального інтерфейсу для сторінки формування протоколу нового дослідження було використано бібліотеку jQuery.

Оскільки для формування протоколу потрібно надати всю можливу термінологію, потрібно було розбити її на дві частини. Перша частина містила органи та їх ділянки, а друга частина формувалася в залежності від того, яку ділянку першої було вибрано. Це дозволило помістити всю необхідну для опису одної ділянки термінологію на одному екрані, що значно полегшує роботу користувача.

За формування другої частини відповідала JavaScript функція innerHtml, що приймала номер ділянки і її назву. Ця функція шукала всі терміни, що містять номер заданої ділянки, для кожного знайденого терміну — ознаки, для ознак — характеристики. В залежності від значення поля last термінів, ознак чи характеристик, вибиралося яке буде для цього елемента відображення. Наприклад, ознака могла бути останньою гілкою, тоді для неї могло бути поле для введення з плейсхолдером. Плейсхолдери бувають двох видів: ті що вносяться до загального опису і ті що слугують тільки підказками. Якщо ознака не остання в ієрархії, то для неї є вибір характеристики. Характеристика аналогічно містить поле для введення.

Враховуючи те, що зазвичай при створені протоколу використовуються лише деякі терміни, немає потреби вимагати заповнення всіх елементів. Для цього була надана можливість заповнювати лише те що необхідно, проте враховувалася інформація лише тих елементів, що були відмічені галочкою. Галочка ставилася автоматично при введенні інформації для будь-якого елементу, що лежав в ієрархії нижче.

Після опису ділянки при натисненні кнопки добавити запускається функція addDecriptionElement1, що робить обхід по всіх елементах блоку опису та вибирає відмічені. Для кожного відміченого елементу перевіряються всі нижче лежачі. Таким чином формуються тези для кожного терміну із ряду ознак з характеристиками. Сукупність цих тез утворює опис обстеження.

**5. РОбота користувача з системою**

Для роботи із системою необхідно пройти авторизацію (ввести логін і пароль, рисунок 5.1). Всього доступно два види користувачів: адміністратор (видає доступ новим користувачам, забирає його, редагує інформацію користувачів) та лікар.

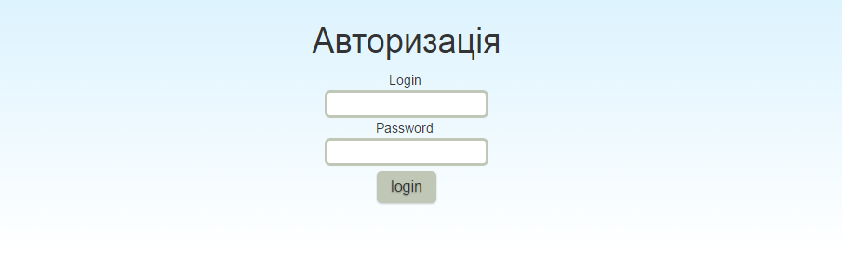


Рисунок 5.1 — Авторизація

**5.1. Робота із щоденним журналом**

Після авторизації користувачеві доступне функціональне меню. За замовчуванню відкривається щоденний журнал (рисунок 5.2).

Користувач може виконувати фільтрацію обстежень по даті, пацієнту, типу обстеження або використовувати декілька фільтрів одночасно. Фільтр по даті дозволяє вибрати всі протоколи сформовані за певний період часу. Фільтр по пацієнту дозволяє вибрати всі протоколи конкретного пацієнта.

При кліку на конкретному обстеженні відкривається його детальний опис (рисунок 5.3).



Рисунок 5.2 — Щоденний журнал

У цьому режимі дозволяється редагування опису, видалення заключень, редагування чи видалення операцій та маніпуляцій.

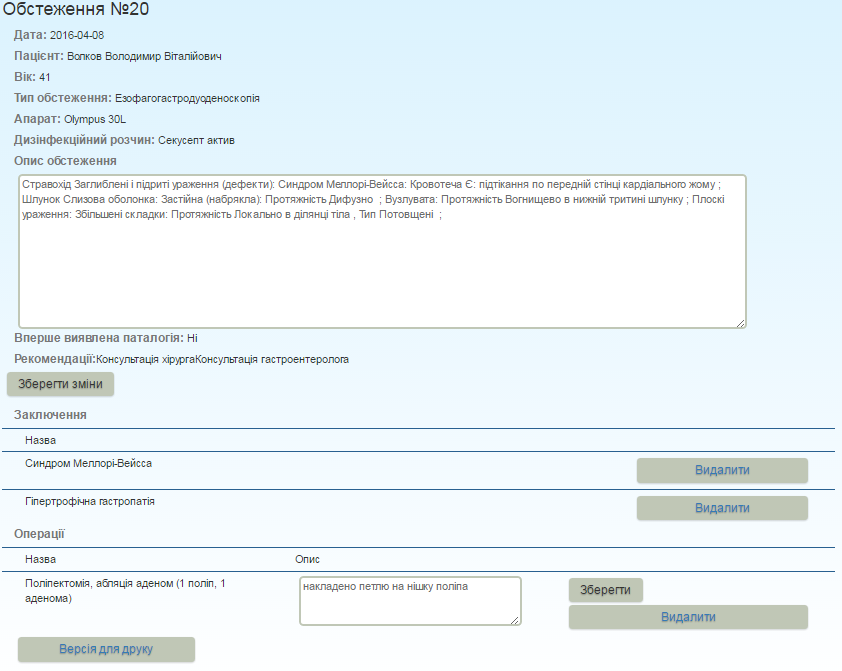


Рисунок 5.3 — Детальний опис обстеження

Також користувач може вибрати версію для друку.У цьому випадку відкриється нова веб-сторінка, на якій буде вся необхідна інформація про ендоскопічне дослідження у вигляді, що відповідає шаблону.

**5.2. Створення нового протоколу дослідження**

Наступна функція — нове обстеження. Формування нового протоколу дослідження відбувається в декілька етапів.

Спочатку задаються початкову налаштування, такі як вибір пацієнта, апарату, яким робиться обстеження, дезінфекційного розчину, хто направив пацієнта та вибір типу обстеження (рисунок 5.4).

Якщо потрібного пацієнта немає у списку вибору, тоді необхідно спочатку створити його медичну карточку в розділі меню “Пацієнти”

Другий етап — формування протоколу ендоскопічного дослідження з використанням мінімальної стандартної термінології (рисунок 5.5). Форма умовно поділена на дві колонки. Зліва — органи і ділянки. При виборі ділянки справа генерується форма для цієї ділянки. Відповідно до інформації з бази даних формуються контроли. Вони бувають декількох видів:

* поле для введення з підказкою, що не виводиться в опис;
* поле для введення з підказкою, що виводиться в опис;
* випадаючий список із характеристиками та поле для введення з підказкою, що не виводиться в опис;
* випадаючий список із характеристиками та поле для введення з підказкою, що виводиться в опис;
* випадаючий список без поля для введення.

Поле для введення може бути виведене для терміна, якщо в нього немає ознак, для ознаки, якщо в неї немає характеристик, для характеристик, якщо потрібно деяке уточнення.

Найчастіше зустрічається випадаючий список разом із полем для введення. До термінів прив’язані заключення, що автоматично добавляються при описі.

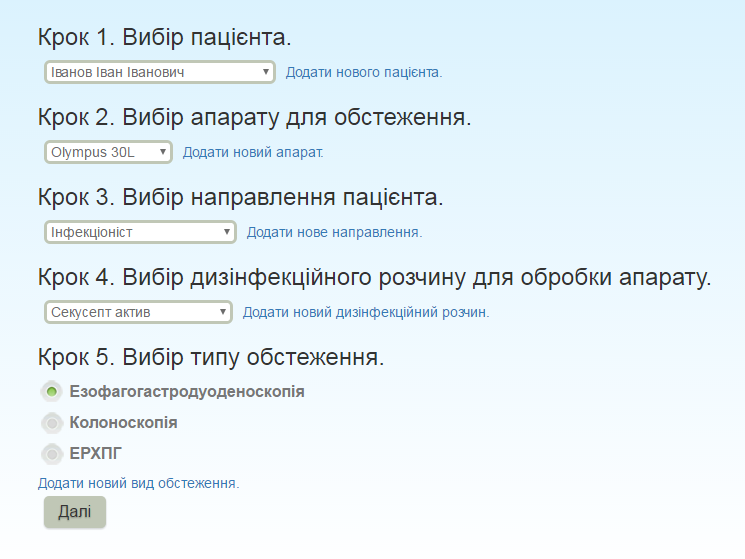


Рисунок 5.4 — Перший етап створення нового обстеження

При введенні інформації у відповідне поле чи виборі певної характеристики, автоматично буде ставитись галочка біля ознаки, до якой належить цей контрол. Ознаки, не відмічені галочками до кінцевого опису не птраплять.

При завершенні опису конкретної ділянки необхідно підтвердити введену інформацію натиснувши на клавішу “добавити”. У результаті в лівій частині форми біля ділянки, що описувалась, появиться галочка та поле з текстом, в якому сформований опис для цієї ділянки. Цей текст у разі необхідності можна відредагувати.

На цьому ж етапі описуються виконані маніпуляції та операції (рисунок 5.6). Для тощо щоб добавити нову маніпуляцію необхідно вибрати її з списку, ввести опис і натиснути кнопку “добавити”. Аналогічно з операцією.

У кожному протоколі незалежно від виду обстеження є контрол для вперше виявленої патології. Він потрібен для підрахунку патологій у звіті.

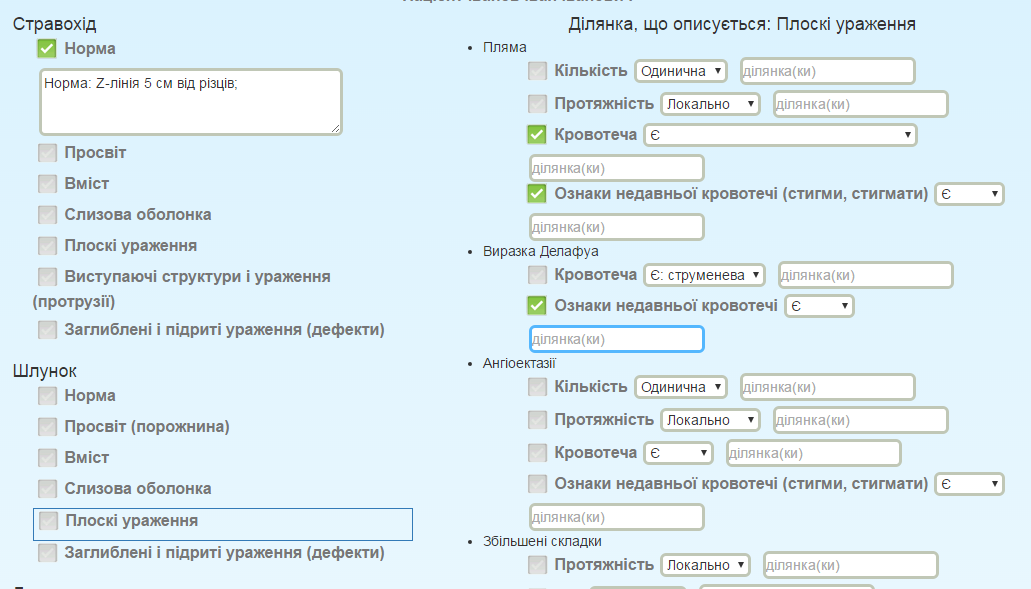


Рисунок 5.5 — Другий етап створення нового обстеження

Щоб завершити формування протоколу ендоскопічного обстеження необхідно натиснути кнопку завершити. Після цього опис кожної ділянки, відміченої галочкою та всі добавлені маніпуляції з операціями будуть занесені до бази даних у відповідні таблиці.

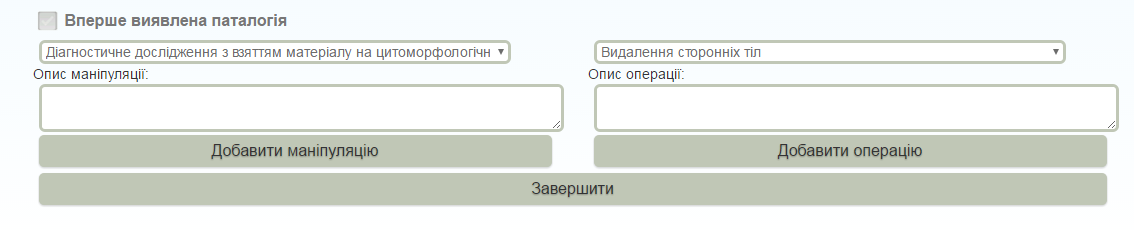


Рисунок 5.6 — Добавляння маніпуляцій та операцій

**5.3. Керування виконаними маніпуляціями**

Ще одна функція — керування виконаними маніпуляціями. На цій сторінці є перемикач між всіма маніпуляціями та маніпуляціями без результатів. За замовчуванням відкривається список всіх виконаних маніпуляцій (рисунок 5.7). У цьому списку можна переглянути інформацію про кожну виконану маніпуляцію, її номер, номер обстеження під час якого вона була виконана, назву, опис та результат. Поля опису та результату доступні для редагування. Для цього необхідно внести зміни і натиснути кнопку “зберегти”. Також можна видалити маніпуляцію у разі потреби.

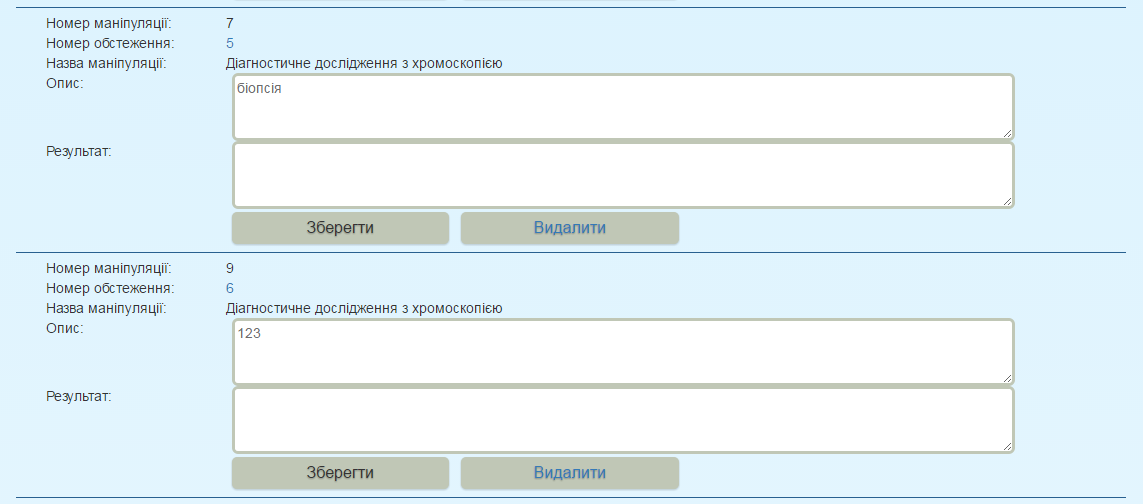


Рисунок 5.7 — Керування маніпуляціями

**5.4. Формування звітів**

Для швидкого формування звітів розроблено декілька їх видів, що базуються на місячному звіті [3]. Всього існує чотири види: місячний, за перший квартал (перші три місяці року), за другий квартал (перша половина року), за третій квартал (перші дев’ять місяців року) та за рік. Для того, щоб сформувати звіт необхідно вибрати потрібний вид і вказати відповідно місяць чи рік, після чого натиснути кнопку “сформувати” (рисунок 5.8). В результаті отримуємо відповідний звіт у вигляді таблиці 5.1. У ній в колонці під номером 1 — число місяця або місяць в залежності від виду звіту. Число у графі 2 дорівнює сумі чисел граф 8+14 та сумі чисел граф 3+4+6. Число у графі 3 дорівнює

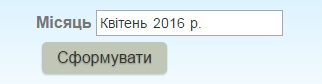


Рисунок 5.8 — Формування звіту

сумі чисел граф 9+15. Число у графі 4 дорівнює сумі чисел граф 10+16. Число у графі 5 (11, 17) із загальної кількості ендоскопічних маніпуляцій — графа 4 (10, 16) дорівнює сумі чисел граф 11+17. Число у графі 6 дорівнює сумі чисел граф 12+18. Число у графі 7 дорівнює сумі чисел граф 13+19. Число у графі 8 дорівнює сумі чисел граф 9+10+12. До графи 9 вноситься загальна кількість проведених діагностичних езофагогастродуоденоскопій [3].

Таблиця 5.1. Щоденний обліку роботи ендоскопічного кабінету

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Число місяця | **Усього** | | | | | | **Езофагогастродуоде-носкопія** | | | | | | **Колоноскопія** | | | | | |
| Ендоскопічних досліджень | Діагностичних досліджень | Ендоскопічних маніпуляцій | Взято матер. на цитоморфологічне дослідження | Ендоскопічних операцій | Кількість хворих з вперше виявленою. онкопаталогією | Ендоскопічних досліджень | Діагностичних досліджень | Ендоскопічних маніпуляцій | Взято матер. на цитоморфологічне дослідження | Ендоскопічних операцій | Кількість хворих з вперше виявленою. онкопаталогією | Ендоскопічних досліджень | Діагностичних досліджень | Ендоскопічних маніпуляцій | Взято матер. на цитоморфологічне дослідження | Ендоскопічних операцій | Кількість хворих з вперше виявленою. онкопаталогією |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

**5.5. Налаштування користувача та список пацієнтів**

Для кожного користувача існують значення за замовчуванням для апарату та дезінфекційного розчину. Щоб це зробити необхідно скористатися пунктом меню налаштування (рисунок 5.9). Якщо вони встановлені, то при створенні нового протоколу ендоскопічного обстеження за замовчуванням убде вказаний відповідний апарат чи розчин.

Також на цій сторінці можна змінити пароль та особисті дані користувача. Для зміни паролю необхідно ввести старий пароль і новий, який буде використовуватись в майбутньому. Для зміни місця роботи та ПІБ необхідно відредагувати відповідні поля і натиснути кнопку “зберегти зміни”. Місце роботи вказується на кожному протоколі ендоскопічного дослідження при його роздруківці.

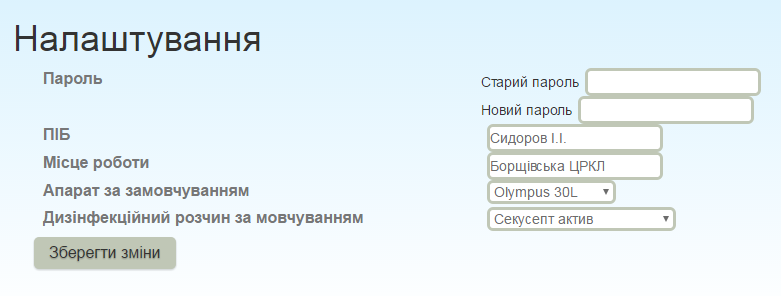


Рисунок 5.9 — Сторінка налаштувань

Для роботи із пацієнтами необхідно скористатись відповідним пунктом меню. На цій сторінці (рисунок 5.10) можна переглянути список пацієнтів та інформацію про них.

Щоб добавити нового пацієнта необхідно внизу сторінки ввести його прізвище, ім.’я, по батькові, ідентифікаційний код, адрес постійного місця проживання, дату народження та натиснути кнопку “добавити”.

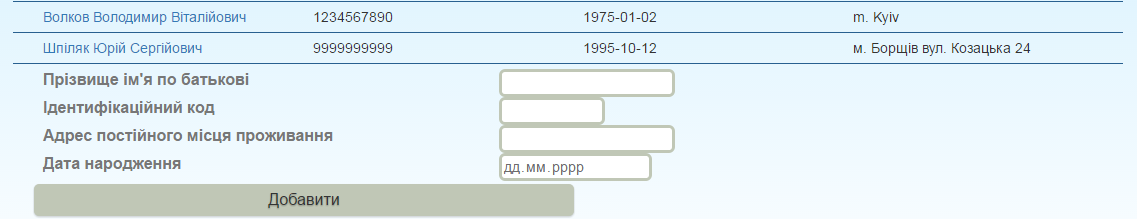


Рисунок 5.10 — Сторінка пацієнтів

Щоб відредагувати дані конкретного пацієнта чи видалити його із списку, необхідно перейти на детальну сторінку клацнувши по його імені в загальному списку. Детальна сторінка аналогічна блоку додавання новго пацієнта.

Редагувати інформацію користувачів системи, проводити їх реєстрацію, видавати доступ може адміністратор. Для цього необхідно зайти під відповідним акаунтом в систему. На цій сторінці доступний список всіх користувачів, доступне їх редагування, видалення, зміна прав доступу.

Сторінка адміністратора (рисунок 5.11) містить список користувачів та їх основну інформацію.

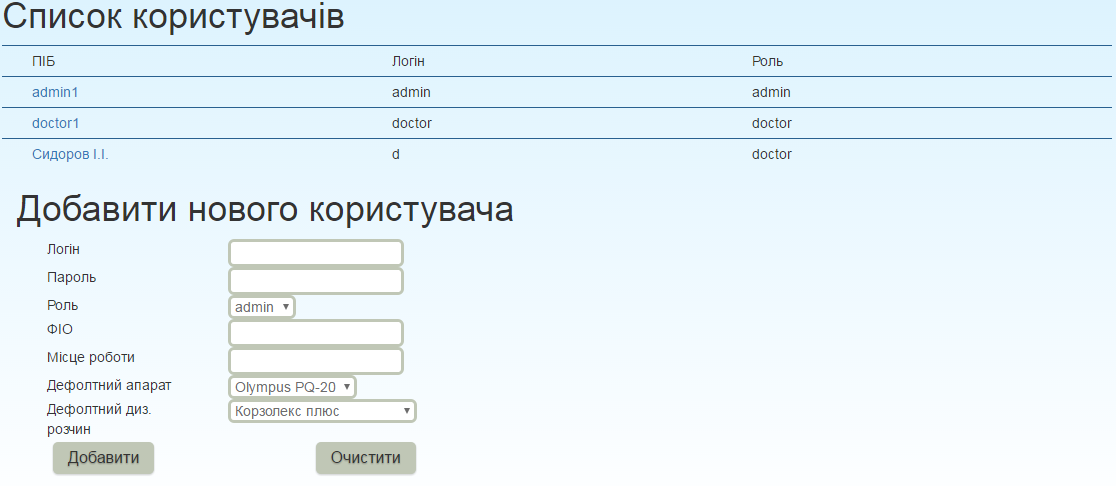


Рисунок 5.11 — Сторінка адміністратора

Для редагування чи детального перегляду інформації користувача потрібно натиснути на його імені в списку контактів. Щоб добавити нового користувача необхідно заповнити поля у відповідному розділі і натиснути кнопку добавити.

**5.6. Робота з словниками**

Для роботи із мінімальною стандартною термінологією потрібно використовувати відповідні словники: типів обстежень, органів, ділянок, термінів, ознак, характеристик, заключень, маніпуляцій, операцій, рекомендацій. Для переходу на сторінку необхідно вибрати потрібний підпункт меню “словники”.

Кожен словник це список даних, що зберігаються в базі даних. Словник типів обстежень, органів, ділянок, термінів, ознак та характеристик дещо відрізняються. На сторінці кожно із цих словників відображаються елементи, що лежать вище по ієрархії (ліва колонка) та нижче по ієрархії (права колонка). Таким чином можна легко відслідкувати зв’язок між ними. Щоб відредагувати конкретний елемент необхідно клікнути по ньому. На відкритій сторінці доступне редагування всієї інформації про елемент а також його видалення.

Роботи з усіма словниками практично однакова. Додавання нового елемента доступне в кінці кожної сторінки. Також кожен користувач може добавити новий апарат, дезінфекційний розчин та направлення у разі потреби або видалити чи відредагувати існуючий.

висновКИ

Проаналізовано мінімально стандартну термінологію, що використовується для опису виконаного дослідження, розроблено її ієрархію.

Створено програмний продукт, який використовується для обліку результатів ендоскопічних досліджень, використовує мінімальну стандартну термінологію, надає можливості для роботи із нею. Для зручної роботи створено електронні медичні карточки, звіти, що відповідають стандарту, налаштування за замовчуванням та автоматичне формування заключення і рекомендацій.

Враховуючи поставлені вимоги до системи, програмне забезпечення розроблено із дотриманням правил медичної термінології.

Програма розроблена у вигляді веб-сайту, що автоматично робить її багатокористувацькою. Серверна частина розроблена з використанням фреймворку Spring MVC та Hibernate 3. Для сервера бази даних обрано MySQL 5.6. Для розробки зручного користувацького інтерфейсу використано Bootstrap та jQuery.

Вибраний набір компонентів ідеально підходить для вирішення поставленої задачі.

Програмний продукт було виконано на замовлення Борщівської ЦРКЛ та впроваджено у використання в ендоскопічному кабінеті в 2016 році. ЇЇ впровадження значно зменшує витрати часу для запису та систематизації результатів ендоскопічних досліджень і дозволяє швидко та ефективно консультуватись між медиками різних спеціальностей та медичних закладів; порівнювати отримані дані, обмінюючись значними обсягами уніфікованої інформації.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Гонсалвес Э. Изучаем Java EE 7 / Энтони Гонсалвес. — Санкт-Петербург: Питер Пресс, 2014. — 640 с.
2. МІНІМАЛЬНА СТАНДАРТНА ТЕРМІНОЛОГІЯ [Електронний ресурс]. — 1998. — Режим доступу до ресурсу: [http://endoscopy.com.ua/assets/ files/MstUkr.pdf](http://endoscopy.com.ua/assets/%20files/MstUkr.pdf).
3. Наказ № 393 [Електронний ресурс]. — 2007. — Режим доступу до ресурсу: <http://endoscopy.com.ua/assets/files/nakaz1.pdf>.
4. Joseph O. Beginning Hibernate / Ottinger Joseph., 2014. — 236 с. — (Apress).
5. IntelliJ IDEA Ultimate [Електронний ресурс] — Режим доступу до ресурсу: <http://jetbrains.ru/products/idea/>.
6. Heffelfinger D. Java EE 7 with GlassFish 4 Application Server / David Heffelfinger., 2014. — 348 с. — (Packt Publishing).
7. Bretet A. Spring MVC Cookbook / Alex Bretet., 2016. — 466 с. — (Packt Publishing).
8. Caliskan M. Beginning Spring / Mert Caliskan., 2015. — 480 с. — (Wrox).
9. Wheeler W. Spring in Practice / Willie Wheeler., 2013. — 560 с. — (Manning).
10. Gutierrez F. Introducing Spring Framework / Felipe Gutierrez., 2014. — 352 с. — (Apress).
11. Spring MVC — сохранение объекта в сессии [Електронний ресурс] — Режим доступу до ресурсу: <http://javastudy.ru/spring-mvc/save-object-in-session/>.
12. Nield T. Getting Started with SQL / Thomas Nield., 2016. — 134 с. — (O'Reilly Media).
13. Boronczyk T. Jump Start MySQL / Timothy Boronczyk., 2015. — 150 с. — (SitePoint).
14. Vukotic A. Apache Tomcat 7 / Aleksa Vukotic., 2011. — 296 с. — (Apress).
15. Zambon G. Beginning JSP, JSF and Tomcat, 2nd Edition / Giulio Zambon., 2012. — 436 с. — (Apress).
16. Using Spring's form tag library [Електронний ресурс] — Режим доступу до ресурсу: <http://docs.spring.io/autorepo/docs/spring/3.2.x/spring-framework-reference/html/view.html>.
17. Crockford D. JavaScript: The Good Parts / Douglas Crockford., 2008. — 172 с. — (O'Reilly Media).
18. Ajax-запрос [Електронний ресурс] — Режим доступу до ресурсу: <http://jquery.page2page.ru/index.php5/Ajax-запрос>.
19. Chaudhary M. Practical jQuery / Mukund Chaudhary., 2015. — 320 с. — (Apress).
20. Bhaumik S. Bootstrap Essentials / Snig Bhaumik., 2015. — 166 с. — (Packt Publishing).