МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської та аспірантської підготовки
Кафедра інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка особистого кабінету споживача теплової енергії»

Виконав студент 2 курсу групи МК- 2
спеціальності 122 Комп'ютерні науки

Нєцик Руслан Ігорович

Керівники к.геог.н., доц.
Кузніченко Світлана Дмитрівна
к.т.н. Рихлік Анджей

Консультант

Рецензент к.т.н., доц.
Гнатовська Ганна Арнольдівна

Одеса 2018

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

БД – база даних

КП «ТМО» – комунальне підприємство «Теплопостачання міста Одеси»

СКБД – система керування базами даних

CSS – Cascading Style Sheets

JS – JavaScript

ВСТУП

У наш час в більшу частину житлових приміщень проведена система опалення для підтримки тепла в холодні пори року. Всі власники опалювальних приміщень зобов'язані оплачувати провіднику системи опалення фіксовану суму що розраховується за кількістю спожитого тепла. Для того що б знати яку суму необхідно оплачувати споживачеві йому доводиться чекати квитанцію або питати безпосередньо у провідника що на тлі розвитку технологій виглядає дуже незручно.

Мета даної роботи є розробка автоматизованої інформаційної системи обліку теплової енергії для КП «Теплопостачання міста Одеси» енергії, а саме персонального особистого кабінету споживача теплової енергії, який дозволить проводити ідентифікацію користувача і надавати йому необхідну інформацію про кількість спожитого їм тепла за певні проміжки часу і суму яку йому необхідно сплатити.

Особистий кабінет – це особливий розділ інформаційної системи, в якому користувач може отримати доступ до особистої інформації, маніпулювати цією інформацією, також здійснювати операції від свого імені.

В ході розробки особистого кабінету будуть розглянуті доступні на сьогоднішній день веб-технології реалізації особистого кабінету, а саме:

– система керування базами даних (СКБД);

– мова для реалізації програмної логіки на стороні сервера;

– засоби побудови веб-інтерфейсу СКБД Firebird мова розмітки HTML, таблиця стилі CSS та ін.

При розробці особистого кабінету буде реалізован функціонал зв’язування даних різних СКБД, оскільки уся важлива інформація яка буде надаватись користувачеві знаходиться на видаленій закрітій базі даних, яка буде тільки давати інформацію за допомогою збережаних процедур.

Також будуть розглянуті існуючі рішення, визначено загальний функціонал, будуть виявлені переваги і недоліки кожної системи.

Для досягнення поставленої мети в роботі необхідно вирішити наступні завдання:

– провести аналіз предметної області;

– провести проектування підсистеми за допомогою UML-засобів;

– провести аналіз доступних технологій реалізації особистого кабінету в інтернеті;

– розробити функціонал особистого кабінету і забезпечити конфіденційність особистої інформації;

– розробити функціонал особистого кабінету відповідно до вимог предметної області, забезпечити читабельний вигляд і доброзичливий інтерфейс користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості об особистому кабінеті споживача теплової енергії

Особистий кабінет – це особливий розділ сайту, в якому користувач може отримати доступ до даних про стан і статичної інформації особового рахунку, деталей і стан замовлення, стані робіт ведуться за проектом і т. д. Здійснює можливість переглядати і роздруковувати таку документацію як рахунки, сертифікати, свідоцтва.

Особистий кабінет використовується в сайтах будь-якої тематики, таких як інтернет магазин, інтернет банкінг, поставлення послуг інтернету, світла і тепла, замовлення квитків на транспорт, в кінотеатр.

Загальні функції особистого кабінету споживача теплової енергії:

– реєстрація;

– відновлення пароля в разі втрати;

– додавання та зміна загальної інформації про користувача;

– зв'язок з адміністрацією сайту;

– надання інформації про рахунки, діяльності, статистики;

– надання інших послуг і можливостей недоступних незареєстрованим користувачам.

Реєстрація – це створення облікового запису яка буде відрізняти користувача від інших в системі. Для реєстрації необхідно заповнити реєстраційну форму такою інформацією як логін, email, пароль, номер телефону, особиста інформація про користувача (ім'я, прізвище, рік народження, стать), номер рахунку, адреси проживання якщо необхідно.

Посилання на реєстрацію зазвичай знаходиться у верхній частині сайту на видному місці. Процес реєстрації повинен бути максимально простим і вимагати тільки необхідну інформацію у користувача. Також на деяких сайту користувачеві може прийти лист-запрошення від іншого користувача цієї системи в якому буде посилання на форму реєстрації. Після заповнення реєстраційної форми користувачеві зазвичай приходить лист на пошту з посиланням для активації облікового запису або відправляється текстове повідомлення з кодом на мобільний телефон. При завершенні активації користувач може зайти в особистий кабінет і користуватися всіма доступними функціоналом.

Відновлення пароля – це процес при якому користувач втратив свій пароль відновлює його за допомогою допоміжного функціоналу. Важливим є забезпечення безпеки інформації про власника особового кабінету і чіткої ідентифікації власника для запобігання використання особистого кабінету зловмисниками. Зазвичай для цього використовуються секретні питання, відправляється лист на пошту або надсилається смс з кодом на мобільний телефон. Іноді ці методи комбінуються. Після ідентифікації власника особового кабінету зазвичай система пропонує ввести йому новий пароль або видає йому новий тимчасовий.

Надання інформації користувачеві є найголовнішою частиною особистого кабінету споживача теплової енергії. Фактично від зручності швидкого надання необхідної інформації в найбільш читабельному вигляді залежить загальна якість особистого кабінету.

Особистий кабінет споживача теплової енергії повинен видавати інформацію про кількість спожитого тепла і необхідної суми до оплати за кожен обліковий місяць в зручному форматі. При підрахунку необхідної суми враховується кількість осіб які проживають за цією адресою і загальна площа приміщення. Крім цього користувач повинен мати можливість додати кілька особових рахунків з одного аккаунта і отримувати по ним всю необхідну інформацію. Мати можливість роздрукувати квитанцію для оплати.

Існує безліч різних способів зв'язку власника особового кабінету з адміністрацією сайту або інших осіб необхідних для виконання будь-яких операцій. Найпростішим є надання користувачеві телефону або email адреси за якими він сам зв'язується з адміністрацією. Об'єктивно це самий незручний для користувача і застарілий спосіб. Наступним методом є функція зворотного дзвінка, коли користувач створює коротку заявку і йому передзвонюють на телефон. Дуже комфортне і швидко для проблем і операцій для вирішення і здійснення яких необхідна або краща голосовий зв'язок.

Останній метод – це реалізація функції живого чату де він спершу описує свою проблему або питання, а потім один з адміністраторів відповідає на поставлене запитання з можливістю швидкого уточнення проблеми і її швидкого вирішення, загальний вигляд показаний на рис 1.1.

Рисунок 1.1 – Приклад реалізованого живого чату

Вибір способу зв'язку з адміністрацією залежить в першу чергу від готовності підприємства надавати співробітників, які могли б підтримувати постійний зв'язок з клієнтом протягом робочого дня.

1.2 Порівняльний аналіз існуючих рішень

1.2.1 Система онлайн платежів ГЕРЦ

ГЕРЦ – одеська автоматизована система прийому та обліку комунальних платежів з метою забезпечити населенню можливість без простоювання в чергах оплатити за надані комунальні послуги, впорядкування розрахунків за енергоносії та забезпечення ефективного контролю за збором і проходженням платежів за житлово-комунальні послуги.

На сайті присутній весь необхідний функціонал і необхідна інформація в тому числі і довідковий центр в якому описаний процес реєстрації і управління обліковим записом.

На головній сторінці описаний основний перелік доступних можливостей користувачеві оплатити за оперделенной послуги. На сайті пропонують оплатити: СДПТ, кварплату, вивезення сміття, гараж, горище, опалення, газ, ліфт, холодну і гарячу воду, водовідведення, Послуги телефонного зв'язку, інтернету, кабельного телебачення. Всі ці послуги можна оплачувати в одному місці з одного облікового запису що дуже зручно.

На ньому грамотно розташована навігація – користувачеві відразу видно, як зареєструватися і почати роботу в особистому кабінеті. При реєстрації користувач повинен ввести ім'я, палось, електронна адресу та телефон (рис 1.2).

Також відразу пропонується підписатися на повідомлення про рахунки і окремо про новини порталу.

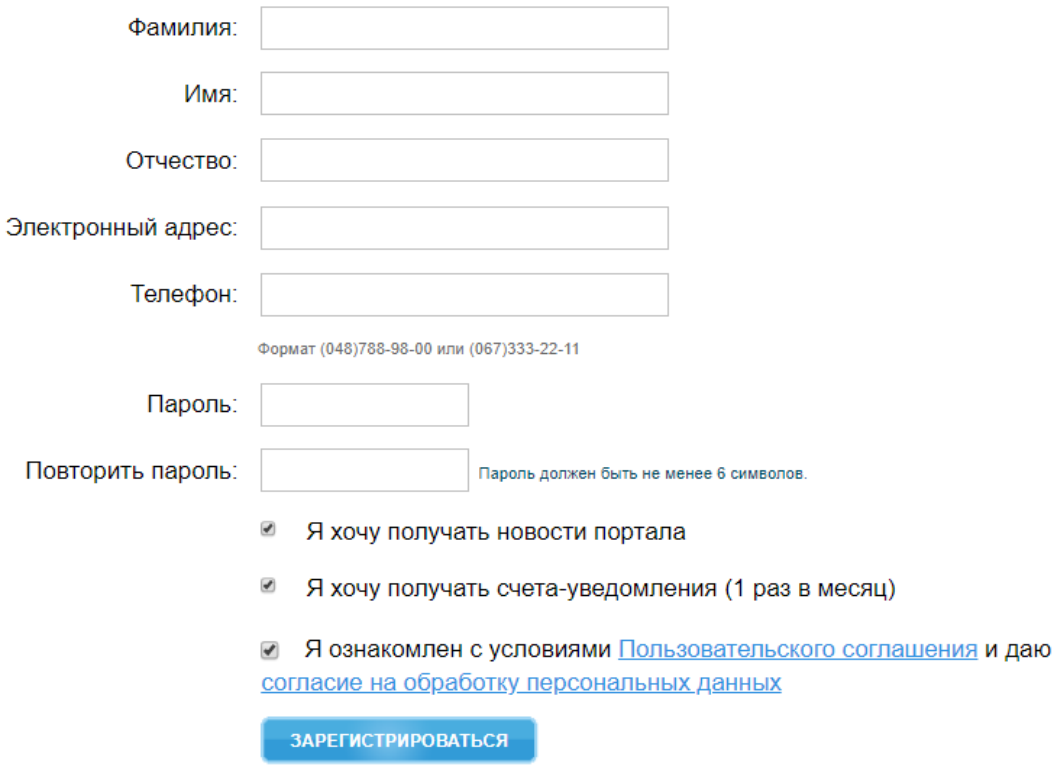


Рисунок 1.2 – Форма реєстрації сайту ГЕРЦ

Загальний вигляд головної сторінки сайту ГЕРЦ представлений на рис 1.3.

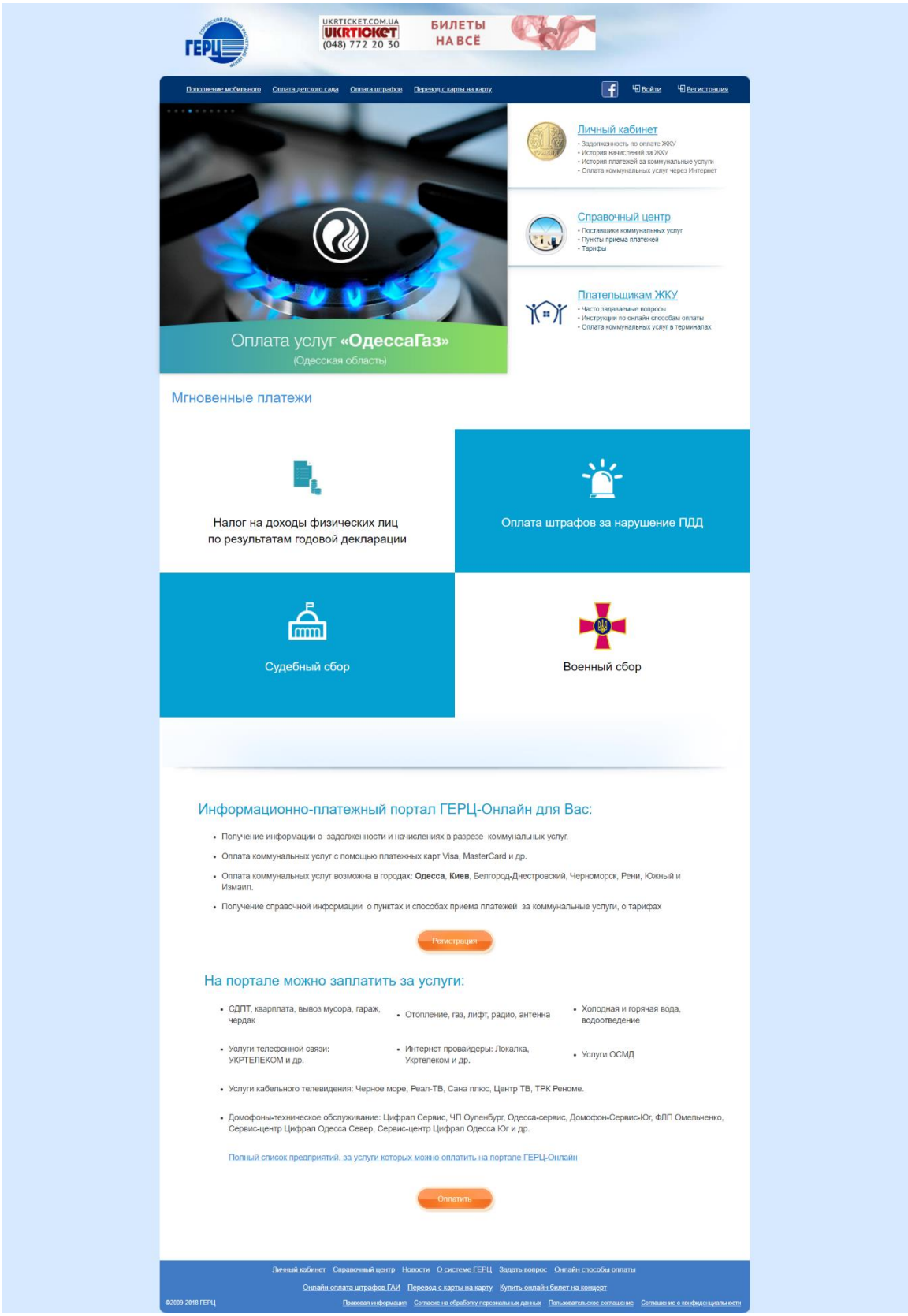


Рисунок 1.3 – Головна сторінка сайту ГЕРЦ

На сайті можна додати житлове приміщення за адресою і побачити все підключені комунальні послуги з цього особового рахунку (рис 1.4).

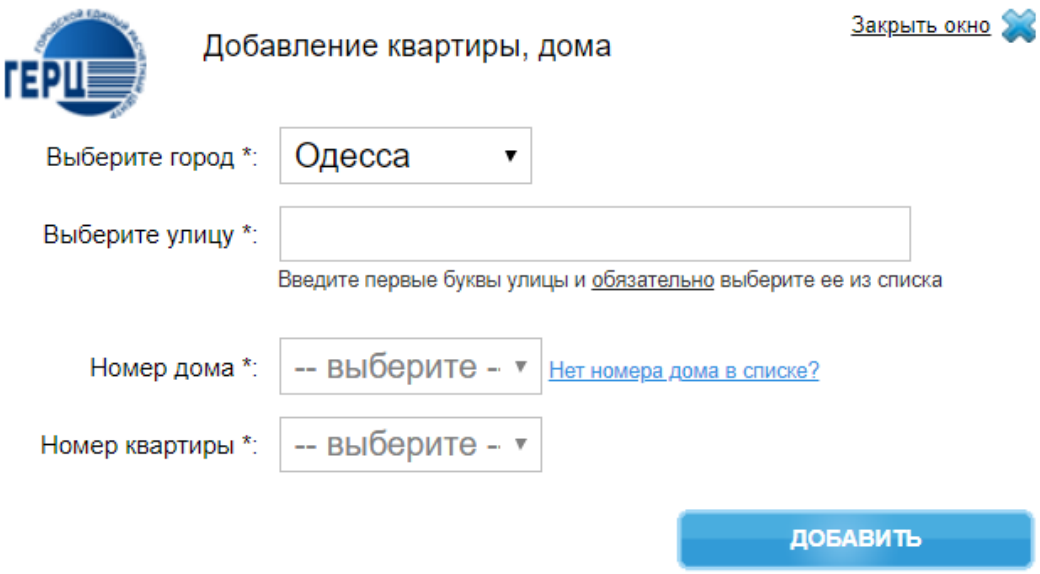


Рисунок 1.4 – Форма додавання жилого будинку за адресою

Біля кожного доданого житлового приміщення відображений загальний борг за всі комунальні послуги (рис 1.5).

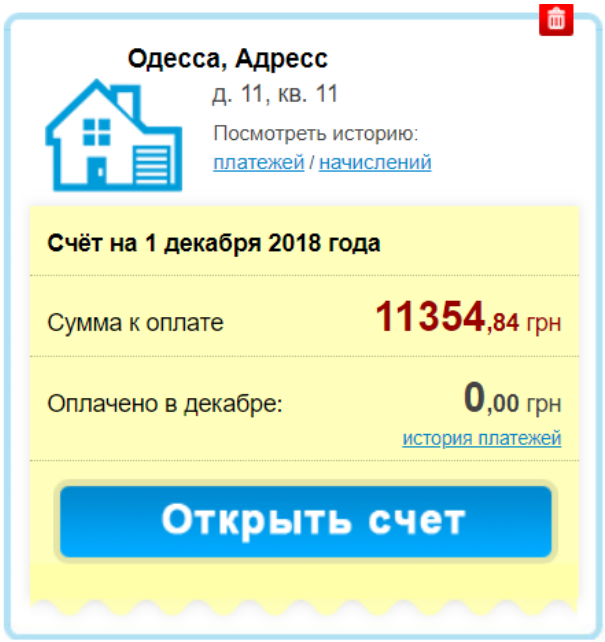


Рисунок 1.5 – Загальний борг по доданому житловому приміщенню

Повний список послуг і боргів / переплат за цими послугами відображається у вигляді таблиці, можна відразу ж оплатити весь борг або за вибрані послуги (рис 1.6).

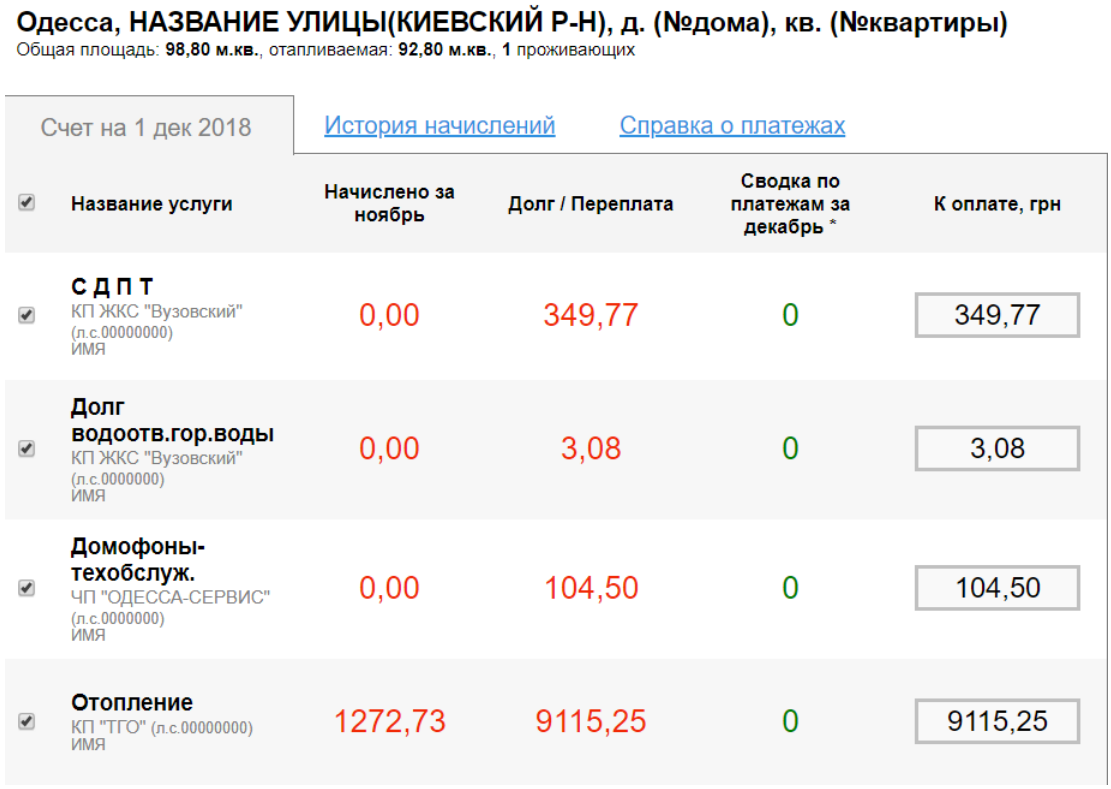


Рисунок 1.6 – Таблиця послуг, що надаються за адресою

Історія платежів зберігається у вигляді таблиці, на якій вказано номер платежу, сума, дата і статус (рис 1.7).

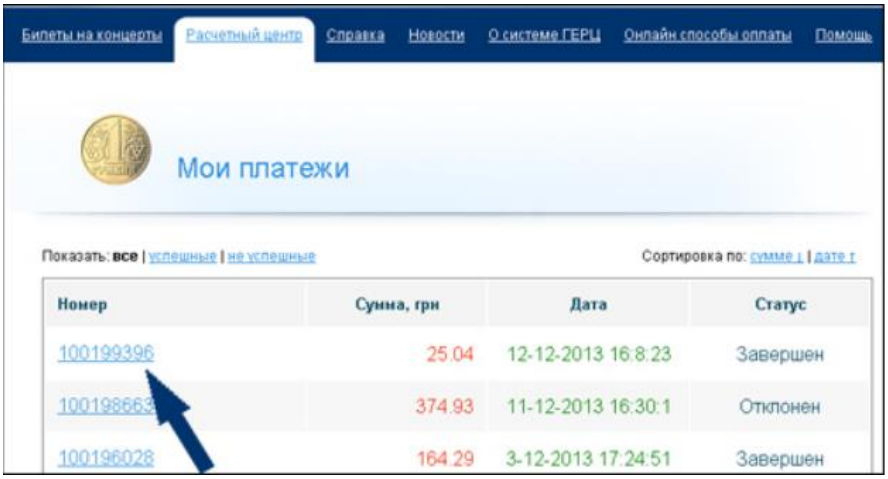


Рисунок 1.7 – Історія платежів

При натисканні на номер платежу в таблиці відкриється детальна інформація про платіж з можливістю друку (рис 1.8).

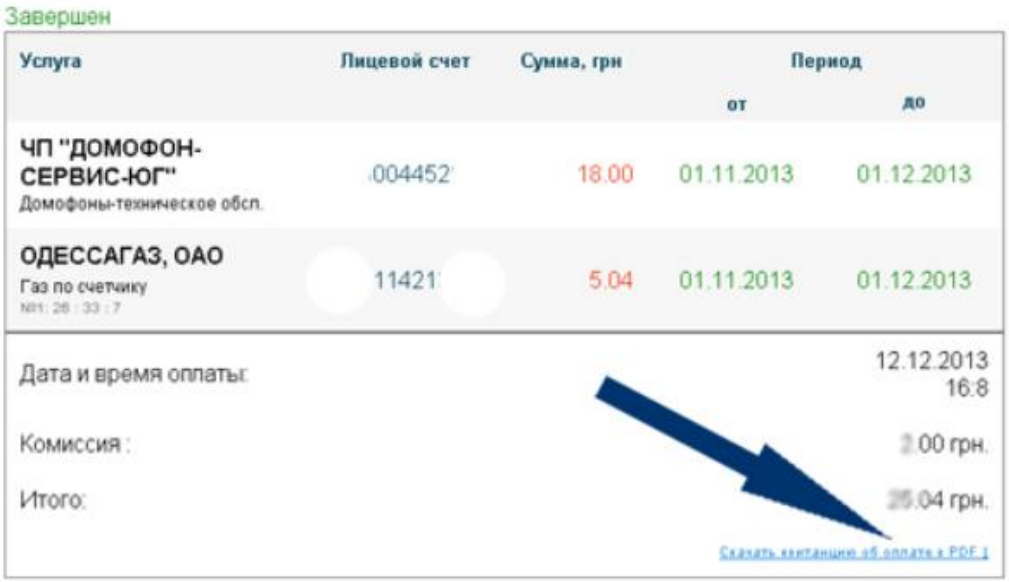


Рисунок 1.8 – Детальна інформация платежу

Крім того, існує можливість оплати штрафів і поповнення рахунку на мобільному телефоні (рис 1.9).

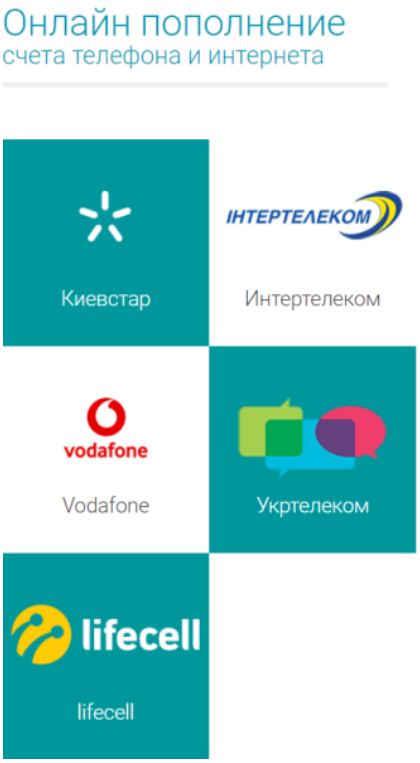


Рисунок 1.9 – Поповнення рахунку для мобільних операторів зв'язку

Особистий кабінет інтернет-порталу ГЕРЦ являє собою великий набір функціоналу, що володіє великими можливостями об'єднаний в одному місці з можливістю онлайн оплати. Користувачеві буде зручно оплачувати всі рахунки з одного аккаунта. Однак з огляду на те що система для загальних розрахунків, в особистому кабінеті мало детальної інформації про кожен вид послуг що надається споживачеві в тому числі і про теплопостачання.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Розробка особистого кабінету буде вестися виходячи з виду та структури бази даних КП «ТМО». Для зображення цього буде використана методологія IDEF0.

Опис системи за допомогою IDEF0 називається функціональною моделлю. Функціональна модель призначена для опису існуючих бізнес-процесів, в якому використовуються як природний, так і графічний мови. Для передачі інформації про конкретну систему джерелом графічного мови є сама методологія IDEF0.

Методологія IDEF0 наказує побудова ієрархічної системи діаграм – одиничних описів фрагментів системи. Спочатку проводиться опис системи в цілому і її взаємодії з навколишнім світом (контекстна діаграма), після чого проводиться функціональна декомпозиція – система розбивається на підсистеми і кожна підсистема описується окремо (діаграми декомпозиції). Потім кожна підсистема розбивається на більш дрібні і так далі до досягнення потрібного ступеня подробиці.

Кожна IDEF0-діаграма містить блоки і дуги. Блоки зображують функції модельованої системи. Дуги пов'язують блоки разом і відображають взаємодії і взаємозв'язку між ними.

Функціональні блоки (роботи) на діаграмах зображуються прямокутниками, які дають зрозуміти зазначені процеси, функції або завдання, які відбуваються протягом певного часу і мають розпізнавані результати. Ім'я роботи повинне бути виражене віддієслівним іменником, що позначає дію.

IDEF0 вимагає, щоб в діаграмі було не менше трьох і не більше шести блоків. Ці обмеження підтримують складність діаграм і моделі на рівні, доступному для читання, розуміння і використання.

Кожна сторона блоку має особливе, цілком певне призначення. Ліва сторона блоку призначена для входів, верхня – для управління, права – для виходів, нижня – для механізмів. Таке позначення відображає певні системні принципи: входи перетворюються у виходи управління обмежує або наказує умови виконання перетворень, механізми показують, що і як виконує функція.

Блоки в IDEF0 розміщуються за ступенем важливості, як її розуміє автор діаграми. Цей відносний порядок називається домінуванням. Домінування розуміється як вплив, яке один блок надає на інші блоки діаграми. Наприклад, самим домінуючим блоком діаграми може бути або перший з необ-

хідної послідовності функцій, або яка планує або контролююча функція, що впливає на всі інші.

При розробці будуть враховані вимоги КП «ТМО», а також загальні переваги користувачів, орієнтир на зручність використання користувачами особистого кабінету. Будуть використані найбільш оптимальні і популярні засоби розробки. Загальна діаграма розробки особистого кабінету в вигляді IDEF0 схеми зображена на рис 2.1.

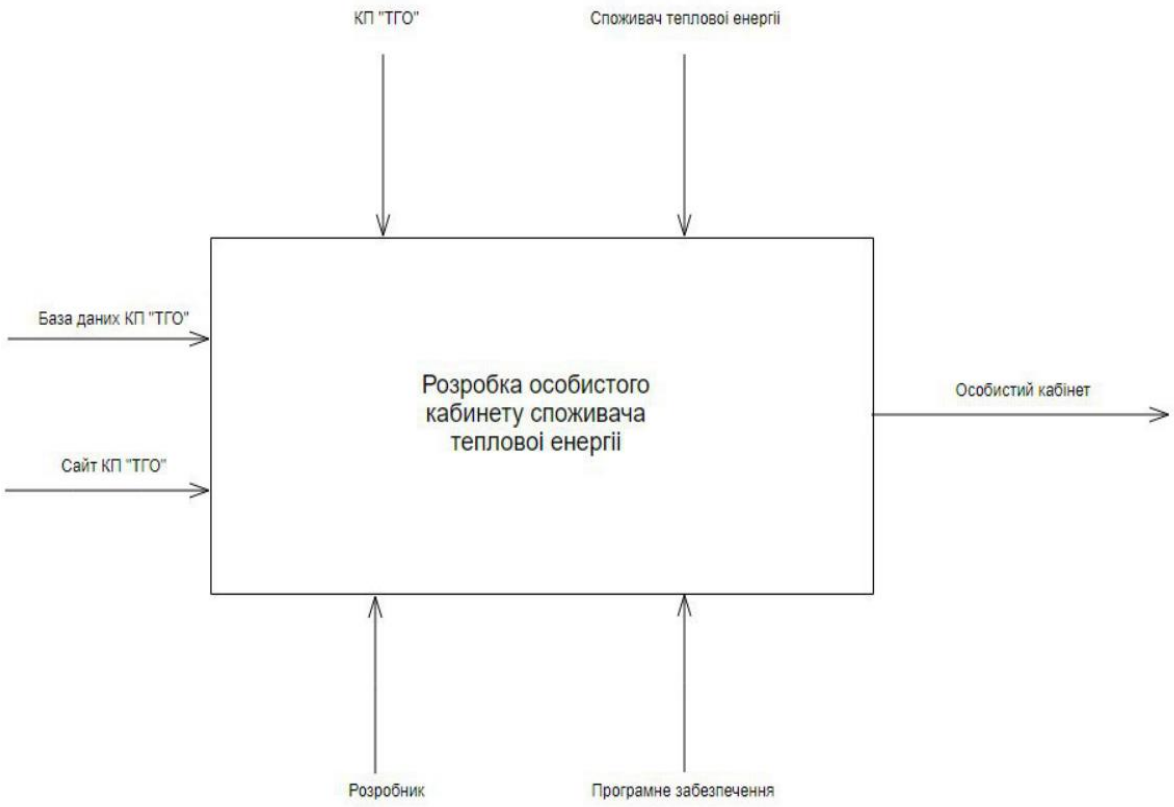


Рисунок 2.1 – Діаграма IDEF0 проекту

Для користування усім доступним функціоналом особистого кабінету користувачеві необхідно буде зареєструватися, які не зареєстровані користувачам буде доступний тільки фукціонал реєстрації відповідно до загальних принципів особистого кабінету.

Візуальне моделювання в UML можна уявити як певний процес поуровневого спуску від найбільш загальної і абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі так званої діаграми варіантів використання (use case diagram), яка опи-

сує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування. Діаграма варіантів використання є вихідним концептуальним поданням або концептуальною моделлю системи в процесі її проектування і розробки.

Розробка діаграми варіантів використання переслідує мети: визначити загальні межі і контекст модельованої предметної області на початкових етапах проектування системи, сформулювати загальні вимоги до функціонального поведінки проектованої системи, розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей, підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми складається в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на моделируемую систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.

У найзагальнішому випадку, діаграма варіантів використання є граф спеціального виду, який є графічної нотації для представлення конкретних варіантів використання, акторів, можливо деяких інтерфейсів, і відносин між цими елементами. При цьому окремі компоненти діаграми можуть бути укладені в прямокутник, який позначає проектовану систему в цілому. Слід зазначити, що відносинами даного графа можуть бути тільки деякі фіксовані типи взаємозв'язків між акторами і варіантами використання, які в сукупності описують сервіси або функціональні вимоги до моделюється системі.

Як було відзначено раціональний уніфікований процес розробки моделі складної системи є розбиття її на складові частини з мінімумом взаємних зв'язків на основі виділення пакетів. У самій мові UML пакет Варіанти використання є підпакету пакету Елементи поведінки. Останній специфицирует поняття, за допомогою яких визначають функціональність модельованих систем. Елементи пакету варіантів використання є первинними по відношенню до тих, за допомогою яких можуть бути описані сутності, такі як системи і

підсистеми. Однак внутрішня структура цих сутностей ніяк не описується. Базові елементи цього пакету – варіант використання і актор. З цих понять ми і приступимо до вивчення діаграм варіантів використання.

Загальний вигляд доступного функціоналу користувача вказано у вигляді UML діаграми прецендентов (рис 2.2).

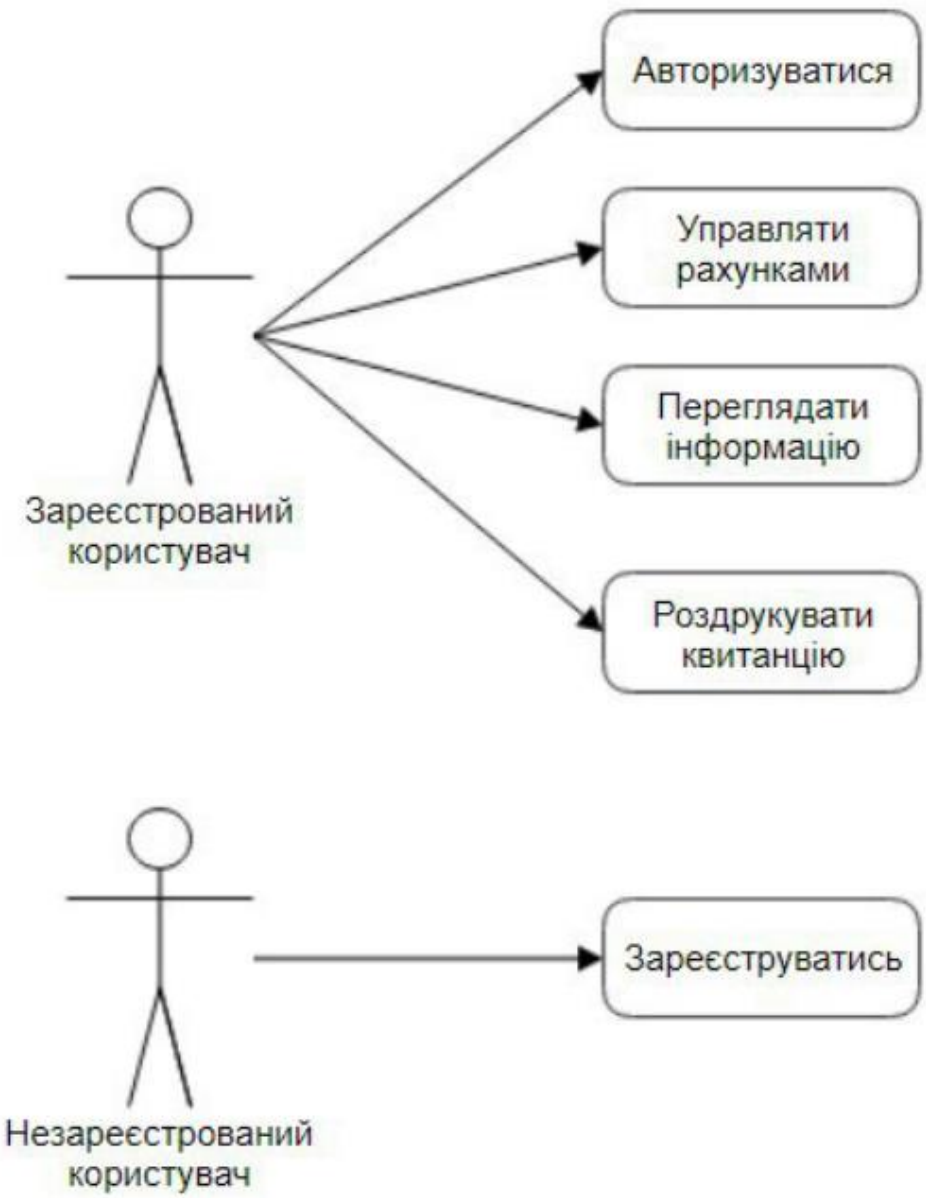


Рисунок 2.2 – Діаграма прецедентів

При підготовці написання функіоналу особистого кабинету було спроектовано частину бази даних що може зберігати усю необхідну інформацію для інтеграції з існуючою віддаленою базою даних, та іншою частиною сайту.

База даних буде зображена у вигляді схеми. Схема бази даних – це її структура, описана на формальній мові, що підтримується СКБД. У реляційних базах даних схема визначає таблиці, поля в кожній таблиці (зазвичай із зазначенням їх назви, типу, обов'язковості), і обмеження цілісності (первинний, потенційні і зовнішні ключі та інші обмеження).

Схеми в загальному випадку зберігаються в словнику даних. Хоча схема визначена на мові бази даних у вигляді тексту, термін часто використовується для позначення графічного представлення структури бази даних.

Основними об'єктами графічного представлення схеми є таблиці і зв'язку, які визначаються зовнішніми ключами.

Для створення схеми бази даних існує безліч інстументів як у СКБД так і у программного забезбечення для створення іншого програмного забезпечення, наприклад Visual Studio.

Оскільки система все написана на реляційній базі даних то і схема буде у вігляді реляційних таблиць.

Реляційна база даних, створена відповідно до проекту канонічної моделі даних предметної області, складається з нормалізованих таблиць, пов'язаних одне-багатозначними відносинами. У такій базі даних забезпечується відсутність дублювання описових даних, їх одноразовий введення, підтримання цілісності даних засобами системи. Зв'язки між таблицями дозволяють виконати об'єднання даних різних таблиць, необхідне для вирішення більшості завдань введення, перегляду і коригування даних, отримання інформації за запитами і виведення звітів.

Зв'язки між таблицями встановлюються відповідно до проекту логічної структури бази даних і запам'ятовуються в схемі даних Access. Схема даних в Access є не тільки засобом графічного відображення логічної структури бази даних, вона активно використовується системою в процесі обробки даних. Створення схеми даних дозволяє спростити конструювання багатотабличних форм, запитів, звітів, а також забезпечити підтримку цілісності взаємозв'язаних даних при введенні і коригування даних в таблицях.

Схема таблиць відображена на рис. 2.3.

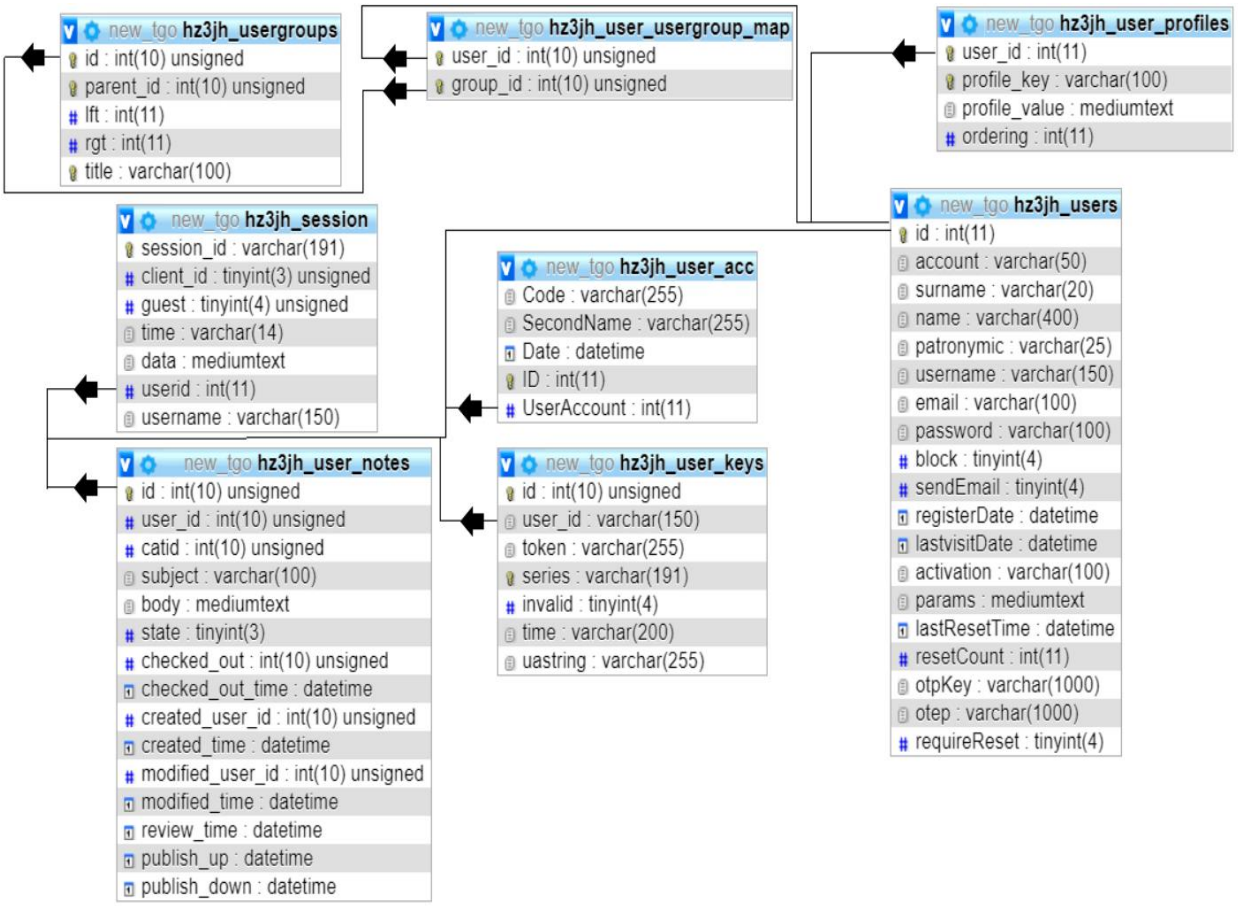


Рисунок 2.3 – Схема таблиць особистого кабінету споживача теплової енергії

Таблиця users зберігає в собі всі дані про всіх користувачів інформаційного порталу (ІП) такі як ім'я, прізвище, логін користувача на ІП, email, пароль (у вигляді хєша), стан блокування користувача на ІП, чи був відправлений email з підтвердженням користувачеві, дата реєстрації користувача, дата останнього візиту користувача на ІП, підтвердив він свій email, поле параметри тільки для адмінів (туди зберігаються настройки аккаунта кожного з адмінів), дата останнього скидання користувачем пароля.

Таблиця usergroups описує різні типи користувачів на інформаційному порталі: адміністратор, редактор новин, редактор тендерів, редактор тарифів, зареєстровані користувачі.

Таблиця user_usergroup_map зберігає в собі id користувача з таблиці users та id його групи з таблиці usergroups.

Таблиця user_notes зберігає в собі всякі замітки, інформація про користувача, яку додав один з адміністраторів сайту.

Таблиця user_keys зберігає в собі ключі cookie.

Таблиця user_profiles зберігає в собі додаткову інформацію яку користувач міг ввестив необов'язкові поля, після реєстрації, такі поля як: адреса, місто, поштовий код, телефон, веб сайт.

Таблиця user_acc зберігає в собі: код користувача, прізвище користувача, і id аккаунта до якого прив'язаний рахунок цього користувача, це таблиця потрібна для того щоб при вході в особистий кабінет отримувати з неї дані які картки повинні бути створені в даному особистому кабінеті.

Для проектування програмного коду буде використана діаграма класів.

Діаграма класів – структурна діаграма мови моделювання UML, що демонструє загальну структуру ієрархії класів системи, їх кооперацій, атрибутів (полів), методів, інтерфейсів і взаємозв'язків між ними. Широко застосовується не тільки для документування та візуалізації, але також для конструювання за допомогою прямого або зворотного проектування. Діаграмма класів представлена на рисунку 2.4.

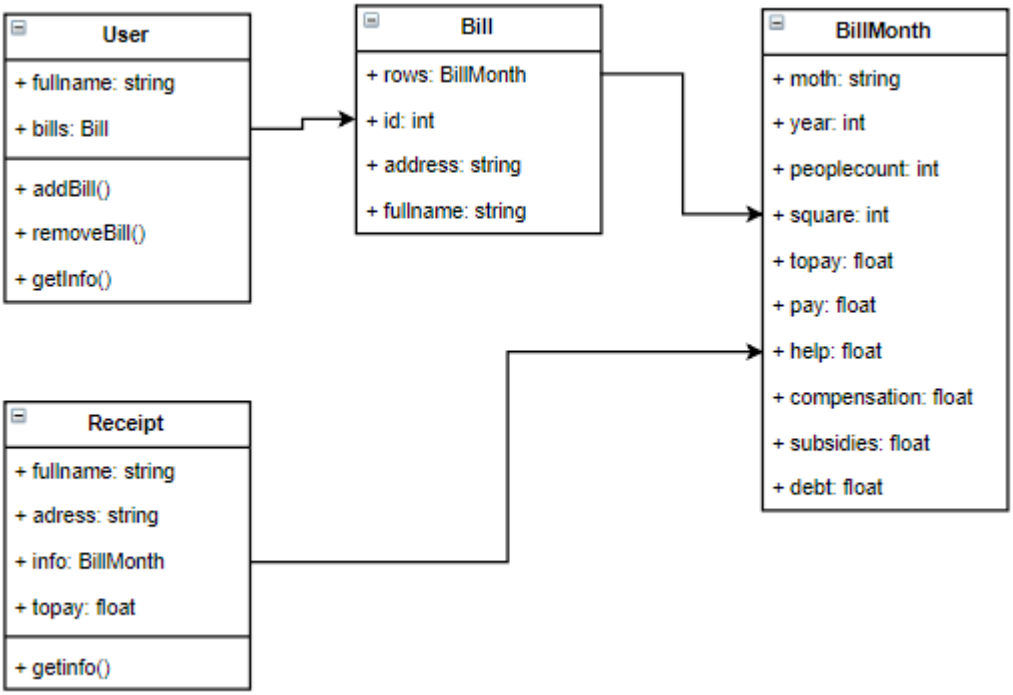


Рисунок 2.4 – Діаграмма класів особистого кабінету споживача теплової енергії

3 ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 СКБД Firebird

3.1.1 Загальний опис СКБД Firebird

Firebird (FirebirdSQL) – кроссплатформенна система управління базами даних з відкритим вихідним кодом підтримуваний багатьма програмістами і фахівцями з інших областей по всьому світу. Його початок було покладено 25 липня 2000 року, коли корпорація Inprise Corp (нині відома як Borland Software Corp) відкрила вихідні коди своєї СКБД Interbase, яка використовувалася в різних інформаційних системах починаючи з 1981 року.

Firebird – це безкоштовна СКБД, вона не вимагає ні реєстрації, ні оплати за підтримку. Вихідний код цієї системи відкритий і будь-який бажаючий може розробляти на його базі власні некомерційні проекти, за умови дотримання вимог ліцензії IDPL, по якій поширюється Firebird.

Firebird повністю підтримує SQL-92 Entry Level 1 і реалізує більшу частину стандарту SQL-99 с деякими дуже корисними доповненнями. Це включає вираження DML / DDL, синтаксис об'єднань FULL / LEFT / RIGHT [OUTER] JOIN, вираження UNION, DISTINCT, підзапити (IN, EXISTS), вбудовані функції (AVG, SUM, MIN, MAX, COALESCE, CASE), обмеження цілісності (PRIMARY KEY, UNIQUE, FOREIGN KEY), та всі загальні типи даних SQL.

Firebird також реалізує обмеження перевірки (check constraints) на рівні доменів і полів, відображення (views), виключення, ролі і управління правами доступу.

3.1.2 Характеристики СКБД Firebird

Відповідність вимогам ACID: Firebird зроблений спеціально, щоб задовольняти вимогам "атомарности, цілісності, ізоляції та надійності" транзакцій ("Atomicity, Consistency, Isolation and Durability").

Версійна архітектура: Основна особливість Firebird – версійна архітектура, що дозволяє сервера обробляти різні версії однієї і тієї ж записи в будь-який час таким чином, що кожна транзакція бачить свою версію даних, не заважаючи сусіднім ("читають транзакції не блокують пишучі, а пишучі не

блокують читаючих"). Це дозволяє використовувати одночасно OLTP і OLAP запити.

Збережені процедури: Використовуючи мову PSQL (процедурний SQL) Firebird, можливо створювати складні збережені процедури для обробки даних повністю на стороні сервера. Для генерації звітів особливо зручні збережені процедури з можливістю вибірки, що повертають дані у вигляді набору записів. Такі процедури можна використовувати в запитах точно так же, як і звичайні таблиці.

Події: Збережені процедури і тригери можуть генерувати події, на які може підписатися клієнт. Після успішного завершення транзакції (COMMIT) він буде сповіщений про події, що відбулися і їх кількості.

Генератори: Ідея генераторів (послідовностей) уможливлює просту реалізацію Автоінкрементний полів, і не тільки їх. Генератори є 64-бітними збереженими в базі даних лічильниками, що працюють незалежно від транзакцій. Вони можуть бути використані для різних цілей, таких як генерація первинних ключів, управління тривалими запитами в сусідніх транзакціях, та ін.

Бази даних тільки для читання: дозволяють поширювати бази даних, приміром, на CD-ROM. Особливо спрощує поширення даних їхнє використання в комбінації з вбудованої версією сервера Firebird (Firebird Embedded).

Повний контроль за транзакціями: Одне клієнтське додаток може виконувати безліч одночасних транзакцій. У різних транзакції можуть бути використані різні рівні ізоляції. Протокол двофазного підтвердження транзакцій забезпечує гарантовану стійкість при роботі з декількома базами даних. Так само доступні оптимістичне блокування даних і точки збереження транзакцій.

Резервне копіювання на льоту: Для резервного копіювання немає потреби зупиняти сервер. Процес резервного копіювання зберігає стан бази даних на момент свого старту, не заважаючи при цьому роботі з базою. Крім того, існує можливість виробляти инкрементальное резервне копіювання БД.

Тригери: Для кожної таблиці можливе призначення декількох тригерів, що спрацьовують до або після вставки, оновлення або видалення записів. Для тригерів використовується мова PSQL, дозволяючи вносити початкові значення, перевіряти цілісність даних, викликати виключення, і т. д. В Firebird 1.5 з'явилися "універсальні" тригери, що дозволяють в одному тригері обробляти вставки, оновлення та видалення записів таблиці.

Зовнішні функції: бібліотеки з UDF (User Defined Function) можуть бути написані на будь-якій мові і легко підключені до сервера у вигляді DLL / SO, дозволяючи розширювати можливості сервера "зсередини".

Декларативне опис посилальної цілісності: Забезпечує несуперечність і цілісність багаторівневих відносин "master-detail" між таблицями.

Набори символів: Firebird підтримує безліч міжнародних наборів символів (включаючи Unicode) з безліччю варіантів сортування [1].

3.1.3 Певеваги та недоліки СКБД Firebird

Firebird – потужна система управління базами даних яка за рахунок своїх привілеїв має великою популярністю серед розробників. Серед всіх привілеїв особливо виділяються:

– вартість – Firebird абсолютно безкоштовний;

– легкість установки – установка Firebird з інсталятора займає не більше хвилини. Але є ще так звана "ручна установка" – в цьому випадку Firebird встановлюється і запускається моментально;

– одночасна робота великої кількості користувачів без будь-яких зависань і блокувань. Firebird в OLTP-системи без праці підтримує роботу одночасно декількох сотень користувачів (максимальне число одночасних підключень залежить від характеристик серверного заліза, можливостей ОС, конфігураційні Firebird, а також від того, наскільки коректно написана клієнтська частина програми бази даних);

– кроссплатформенность. Firebird працює під Windows (32 і 64 бітні), різні версії Linux (32- і 64-бітові), Solaris (Sparc і Intel), HP-UX (PA-Risc) і MacOS X. Клієнтська і серверна частини Firebird можуть працювати під різними операційними системами. Багато хто віддає перевагу використовувати Linux в якості серверної ОС, оскільки таке рішення на практиці виходить більш надійним і дешевим;

– відкритість вихідних кодів. Будь-який бажаючий може створити збірку "під себе", а також доопрацювати цікавить його функціонал;

– зручні засоби адміністрування. Для адміністрування баз даних Firebird призначений чудовий інструмент – IBExpert. Дана програма є безкоштовною для жителів країн СНД, в тому числі для російського користувача. Можна вважати даний інструмент кросплатформним (на Linux'е він без проблем запускається під Wine). На думку автора, IBExpert значно зручніше, ніж "SQL Server Management Studio". Складається відчуття (можливо, трохи

суб'єктивне), що розробників "SQL Server Management Studio" цікавить тільки сам факт наявності тієї чи іншої можливості, але їх зовсім не турбує, наскільки зручно ними користуватися. Не можна сказати, що в IBExpert'е все ідеально, але працювати з ним виходить набагато швидше і зручніше (це все-таки комерційний проект);

– чи не вимогливий до ресурсів. Для роботи з багатогігабайтними (і навіть терабайтними) БД Firebird'у досить средненькой персональні виставки. Зрозуміло, продуктивність СКБД Firebird безпосередньо залежить від характеристик заліза: процесора, пам'яті, жорсткого диска / SDD. При необхідності, Firebird може використовувати більше 2 ГБ ОЗУ, але при цьому потрібна 64-розрядної версії ОС і Firebird'а.

Однак, на тлі СКБД Firebird можна виділити такі переваги у конкурентних СКБД MSSQL і Oracle:

– величезна кількість літератури. Російськомовної літератури по SQLServer'у набагато більше, ніж по всіх інших СКБД (разом узятим);

– вбудована реплікація. За замовчуванням будь-яка таблиця в SQLServer'е містить службову інформацію, необхідну для роботи механізму реплікації;

– вбудований генератор звітів не знайдені у Firebird;

– вибірка з безлічі баз даних. SQLServer дозволяє з одного SELECT-запиту вибрати дані з різних баз даних, розташованих як на цьому ж сервері, так і на віддалених серверах;

– вбудовані засоби шифрування даних;

– повнотекстовий пошук. Дозволяє швидко відшукати документ по заданому текстовому фрагменту (пошук здійснюється в BLOB-полях);

– файлові групи і сегментація таблиць. Дозволяє миттєво переміщати інформацію між таблицями і швидко видаляти величезні обсяги даних, однак вимагає від розробника солідної підготовки [2].

СКБД Firebird має величезну кількість функцій для роботи в проектах будь-якої величини, однак у деяких розробників може виникнути проблеми з її повним засвоєнням на увазі відносно скромною популярності і малої кількості документації. Проте з огляду на те що вона безкоштовна є одним з кращих рішень для середніх проектів де необхідний повний базовий набір функціоналу.

3.2 Опис та характеристики СКБД MySQL

Система управління базами даних (СКБД) MySQL – розробка шведської компанії MySQL AB. СКБД MySQL є програмним забезпеченням з відкритим вихідним кодом, поширюваним за ліцензією GNU (GPL) і комерційної ліцензії для ситуацій, які не підпадають під дію ліцензії GPL. MySQL підтримує реляційну модель даних, тобто є реляційною СКБД.

Починаючи з версії 5.0, СКБД MySQL практично повністю задовольняє стандарту структурованого мови запитів SQL і, отже, сумісна з іншими базами даних [3].

Розглянемо основні переваги СКБД MySQL:

високa якість – MySQL характеризується стійкою роботою;

поряд з Oracle, MySQL вважається однією з найшвидших СКБД в світі;

відкритий код доступний для перегляда та модернізації, що дозволяє постійно покращувати програмний продукт;

СКБД MySQL, розроблена з використанням мов C / C ++, протестована на багатьох платформах, серед яких Windows, Linux, FreeBSD, Mac OS X, OS / 2, Solaris і ін;

MySQL підтримує API (Application Programming Interface, програмний інтерфейс програми) для C, C ++, Eiffel, Java, Perl, PHP, Python, Ruby і Tcl. MySQL можна успішно застосовувати як для побудови Web-сторінок з використанням Perl, PHP і Java, так і для роботи прикладної програми, створеної з використанням Delphi, Builder C ++ або платформи .NET;

СКБД MySQL надає широкий вибір типів таблиць, в тому числі і сторонніх розробників, що дозволяє реалізувати оптимальну для розв'язуваної задачі продуктивність і функціональність;

локалізація в MySQL виконана коректно. У користувача, як правило, не виникає проблем при обробці російського вмісту БД [4].

Ще в версії MySQL 4.1 з'явилися такі важливі нововведення, як повна підтримка вкладених запитів і підтримка транзакцій. А у версії MySQL 5.0 стали доступними такі важливі механізми:

збережені процедури і функції, які об'єднують в собі цілі послідовності запитів;

тригери – збережені процедури, прив'язані до події зміни таблиці;

уявлення – вибірки даних які можна уявити як повноцінні реально існуючі таблиці бази даних;

курсори, що дозволяють циклі переглянути кожен рядок результуючої таблиці запитів;

інформаційна схема – стерпний набір уявлень системної таблиці, в якій зберігається різноманітна внутрішня інформація;

обробники помилок;

безліч нових функцій.

У стандартному дистрибутиві MySQL поставляються клієнтські програми (утиліти), які взаємодіють з MySQL-сервером: mysql (консольний клієнт для доступу до MySQL-сервера, що дозволяє виконувати SQL-запити), mysqladmin (утиліта для виконання адміністративних функцій, таких як створення або видалення баз даних, отримання інформації про процеси, стан сервера і т. п.), mysqldump (утиліта для виведення вмісту бази даних MySQL у вигляді текстового файлу з SQL-операторами), mysqlimport (виконує перенесення інформації з текстового файлу в таблиці баз даних) і mysqlshow (відображає Щоб отримати інформацію про існуючі базах даних, таблицях, полях і індексах).

До утиліт, які можуть функціонувати без підключення до сервера MySQL, відносяться: myisampack (стискає таблиці типу MylSAM, зменшуючи їх в розмірі і роблячи доступними тільки для читання), mysqlcheck (утиліта, яка використовується для опису, перевірки, оптимізації та відновлення таблиць), mysqlbinlog (дана утиліта використовується для читання вмісту журналу двійковій реєстрації при відновленні даних в нештатних ситуаціях) і perror (утиліта, яка виводить розшифровку кодів системних помилок і помилок MySQL).

Створення, заповнення та зміна таблиць в MySQL здійснюються за допомогою стандартних SQL операторів CREATE TABLE, ALTER TABLE, INSERT, UPDATE, DELETE. Визначено також оператор REPLASE, який діє аналогічно INSERT, але таким чином, що якщо значення індексу UNIQUE або PRIMARY KEY в старому записі таблиці таке ж, як і в новій, то стара запис перед занесенням нової буде видалена. Оператор TRUNCATE TABLE призначений для повного очищення таблиці.

Переглянути список таблиць поточної бази даних дозволяє конструкція SHOW TABLES, а отримати опис стовпців конкретної таблиці можна за допомогою оператора DESCRIBE, який не є стандартним і присутній лише в MySQL SQL [5].

СКБД MySQL підтримує кілька типів таблиць. Тип MyISAM призначається за замовчуванням при створенні таблиці. Кожна MyISAM-таблиця зберігається на диску в трьох файлах, імена яких збігаються з назвою таблиці table_name: table_name.frm, table_name.myd і table_name.myi. Перший з них містить структуру таблиці і інформацію про шпальтах і індексах. Другий містить дані таблиці, а третій – її індекси. Таблиці типу MyISAM можна переносити з сервера простим копіюванням файлів.

Таблиці типу InnoDB можуть досягати обсягу в 1 Тбайт. Таблиці цього типу зберігаються в єдиному табличному просторі. Даний тип таблиць підтримує транзакції, блокування на рівні окремих записів і єдиний з типів таблиць MySQL підтримує зовнішні ключі і каскадне видалення (оновлення). Втім, таблиці InnoDB поступаються в швидкості обробки таблиць MyISAM.

Для створення InnoDB-таблиці застосовується ключове слово ENGINE = InnoDB в операторі CREATE TABLE.

Також MySQL підтримує наступні типи таблиць: MERGE (дозволяє згрупувати кілька таблиць MyISAM в одну), MEMORY (таблиця зберігається в оперативній пам'яті), EXAMPLE, BDB, NDB Cluster (для організації кластерів розподілених таблиць), ARCHIVE (для зберігання великих обсягів інформації в стислому вигляді), CSV (формат таблиць Ms Excel), FEDERATED (для зберігання даних на віддаленому комп'ютері), BLACKHOLE.

Під транзакцією розуміють виконання групи SQL-запитів як єдиної операції. Іншими словами, або виконуються всі ці запити, або (якщо на якомусь етапі стався збій) скасовуються всі результати їх роботи. Транзакції виконуються незалежно один від одного, тобто до завершення транзакції блокується доступ до об'єктів бази даних, що піддаються обробці.

Транзакції підтримуються в MySQL для таблиць типу BDB і InnoDB.

Транзакції не можуть бути вкладеними, тому що всякий оператор, який розпочинає нову транзакцію, завершує попередню.

За замовчуванням MySQL працює в режимі автоматичного завершення транзакцій. Після виконання будь-якого запиту, що модифікує дані, результат відразу записується в базі.

MySQL підтримує кілька типів даних.

Числові дані. До них відносять цілі числа, що не містять дробової частини (наприклад, 124), а також речові числа, що мають як цілу, так і дробову частини (наприклад, 56.45). Числові дані діляться на точкові (bit, boolean, integer і decimal) і наближені (float, real і double precision).

MySQL має п'ять цілих типів: TINYINT, SMALLINT, MEDIUMINT, BIGINT. Різниця між ними полягає в діапазоні величин, які можна зберігати в шпальтах такого типу. Тип TINYINT має діапазон від -27 до 27-1, тип SMALLINT – від -215 до 215-1, тип MEDIUMINT – від -223 до 223-1, тип BIGINT – від-263 до 263-1.

Цілі типи даних можуть бути оголошені позитивними. Для цього призначено ключове слово UNSIGNED. В цьому випадку елементам даного стовпця можна буде привласнити негативні значення, а допустимий діапазон значень, які може приймати тип, подвоюється.

При оголошенні цілого типу можна задати кількість відводяться під число розрядів (від 1 до 255).

Тип BIT призначений для зберігання бітових полів.

Тип BOOLEAN є синонімом для TINYINT (1). Значення 1 розглядається як істина (true), а 0 – як брехня (false).

Тип DECIMAL, а також його синоніми NUMERIC і DEC призначені для величин підвищеної точності, наприклад, для грошових даних. Необхідна точність задається при оголошенні даних одного з цих типів, наприклад, DECIMAL (5,2). Тут цифра 5 визначає загальне число символів, що відводяться під число, а цифра 2 задає кількість знаків після коми. При цьому перший параметр може приймати максимальне значення, рівне 64, а другий – максимальне значення, рівне 30.

Для подання дійсних (наближених) типів в СКБД MySQL є типи: FLOAT (діапазон від -3.4E + 38 до 3.4E + 38, точність 1.2E-39), DOUBLE і DOUBLE PRECISION (-1.8E + 308 до 1.8E + 308, точність 2.2E-308). Числові типи даних з плаваючою точкою також можуть мати параметр UNSIGNED. Як і в цілочисельних типах, цей атрибут запобігає зберігання в зазначеному стовпці негативних величин, але, на відміну від цілочисельних типів, максимальний інтервал для величин стовпчика залишається колишнім.

При формуванні структури таблиці необхідно звертати увагу на розмір, займаний тим або іншим типом даних: якщо значення в полях стовпчика ніколи не будуть виходити за межі 100, не слід вибирати тип більше TINYINT. Якщо ж передбачається зберігати тільки цілочисельні дані, то застосування атрибута UNSIGNED дозволить збільшити діапазон в два рази.

Строкові дані – послідовність символів, укладених в одинарні або подвійні лапки: 'Hello world', '123', "MySQL". Оскільки в якості стандарту в SQL визначені одинарні лапки, для сумісності з іншими базами даних рекомендується використовувати саме їх. Розрізняють строкові типи CHAR,

VARCHAR, BLOB, TEXT, MEDIUMTEXT, MEDI-UMBLOB, LONGTEXT, LONGBLOB, ENUM, SET.

Тип CHAR дозволяє зберігати рядок фіксованої довжини; його доповнює тип VARCHAR, що дозволяє зберігати рядки змінної довжини. Довжина рядка може змінюватися від 0 до 65 535. При створенні таблиці можна змішувати стовпці типу CHAR і VARCHAR. В цьому випадку СКБД MySQL змінить тип стовпців відповідно до правила: якщо таблиці присутній хоча б один стовпець змінної довжини, всі стовпці типу CHAR будуть приведені до типа VARCHAR.

Тип TEXT зазвичай використовується для зберігання великих обсягів тексту (до 216-1 символів), в той час як BLOB (також до 216-1 символів) – для великих двійкових об'єктів, таких як електронні документи, зображення, музичні файли і т. п. типи MEDIUMBLOB, MEDIUMTEXT мають максимальний розмір (224-1) символів, а типи LONGTEXT, LONGBLOB – (232-1) символів.

До особливих типів даних відносяться ENUM і SET. Рядки цих типів приймають значення з заздалегідь визначеного списку допустимих значень. Основна відмінність між ними полягає в тому, що значення типу ENUM має містити точно одне значення із зазначеного безлічі, тоді як стовпці SET можуть містити будь-який (або все) елементи заздалегідь заданої множини. Так, значення для стовпця, оголошеного як ENUM ('y', 'n'), можуть набувати значень або 'y', або 'n'.

Для типа SET, так само, як і для типа ENUM, при оголошенні задається список можливих значень, але в комірці таблиці може зберігатися будь-яке значення зі списку, а порожній рядок означає, що жоден з елементів списку не вибрано. Наприклад, значення для стовпця SET ('y', 'n') можуть приймати значення ('y', 'n'), ('y'), ('n') і порожня множина ().

СКБД MySQL має п'ять календарних типів даних: DATE, DATETIME, TIME, TIMESTAMP і YEAR. Тип DATE призначений для зберігання дати, TIME – для часу доби, а TIMESTAMP – для подання та дати, і часу доби. Тип TIMESTAMP призначений для представлення дати і часу доби у вигляді числа секунд, що пройшли з півночі 1 січня 1970 року. Тип даних YEAR дозволяє зберігати тільки рік. Для значень, що мають тип DATE і DATATIME прийнятий формат YYYY-MM-DD або YY-MM-DD. У типах TIME і DATATIME час наводиться в звичному форматі hh: mm: ss.

Якщо час задається в неприпустимому форматі, то в поле записується нульове значення. Нульове значення присвоюється полях тимчасового типу та за замовчуванням, коли їм не присвоюється ініціює значення.

Тип NULL. Якщо поле може приймати значення NULL, то у визначенні стовпця після типу даних слід вказати ключове слово NULL. Якщо ні при яких обставинах поле не повинно приймати значення NULL. Визначення індексу по деякому стовпцю означає, що СКБД створює його копію, яка потім постійно підтримується в відсортованому стані. Пошук по такому стовпцю здійснюється дуже швидко, так як заздалегідь відомо, де необхідно шукати потрібне значення. Індекс може бути визначений по одному або по декількох стовпцях; в одній таблиці може бути кілька індексів.

MySQL підтримує кілька видів індексів: первинний ключ – головний індекс таблиці; звичайний індекс – таких індексів може бути кілька; унікальний індекс – значення в індексованих стовпці не повинні повторюватися (унікальних індексів також може бути кілька), повнотекстовий індекс – спеціальний вид індексу для стовпців типу TEXT, що дозволяє виробляти повнотекстовий пошук. Індекси можуть мати як власні імена, так і імена індексованих стовпців. Один індекс може охоплювати один або декілька стовпців.

У таблиці MySQL може бути не більше 32 індексів, а многостолбцовий індекс може включати не більше 16 стовпців.

В якості первинного ключа може виступати не тільки цілочисельний стовпець, а й текстові дані. В цьому випадку при створенні таблиці необхідно додати в круглих дужках число символів, що входять в індекс PRYMARY KEY (name (10)).

Можна індексувати від 1 до 1000 символів текстового стовпчика. Первинний ключ може бути оголошений не по одному стовпці PRIMARY KEY (id_PC, name (10)).

Цілочисельний первинний ключ може бути забезпечений атрибутом AUTO_INCREMENT, який забезпечує автоматичне створення унікального значення: Id INT (11) NOT NULL PRIMARY KEY AUTO_INCREMENT.

Передача колонки, забезпеченому цим атрибутом, значення NULL або 0 призводить до автоматичного присвоєння йому максимального значення стовпця плюс 1. Даний механізм є досить зручним і дозволяє не турбуватися про створення унікального значення засобами прикладної програми, яка працює з СКБД MySQL. Значення AUTO_INCREMENT можна змінити і почати відлік не з 1, а, наприклад, від 1000.

Оголошення звичайних індексів проводиться за допомогою ключового слова INDEX або KEY.

Для унікальних індексів вводиться додаткове ключове слово UNIQUE, тобто пишеться UNIQUE INDEX і UNIQUE KEY: Створення звичайних індексів, на відміну від первинного ключа і унікального індексу, допустимо для стовпців, забезпечених атрибутом NULL. Для звичайних індексів (на відміну від первинного ключа) неприпустимо їх визначення разом зі стовпцем.

Посилальна цілісність і зовнішні ключі в СКБД MySQL підтримуються, починаючи з версії 3.23.44, але тільки для таблиць типу InnoDB.

Відзначимо, що MySQL підтримує оператори CREATE INDEX і DROP INDEX для створення (видалення) індексів після визначення таблиць [6].

3.3 Мова програмування PHP

3.3.1 Загальний опис PHP та PHP фреймворків

PHP – мова програмування, що використовується на стороні WEB-сервера для динамічної генерації HTML-сторінок. Про це говорить і розшифровка його назви: PHP – Personal HyperText Processor.

Початком PHP можна вважати осінь 1994 року, коли Расмус Лердорф (Rasmus Lerdorf) вирішив розширити можливості своєї Home-page (Домашньої сторінки) і написати невеликий движок для виконання найпростіших завдань. Такий движок був готовий до початка 1995 року і називався Personal Home Page Tools. Умів він не дуже багато – розумів найпростіший мова і всього кілька макросів.

До середини 1995 року з'явилася друга версія, яка називалася PHP / FI Version 2. Приставка FI – приєдналася з іншого пакета Расмуса, що вмів обробляти форми (Form Interpritator). PHP / FI компілювався всередину Apache і використовував стандартний API Apache. PHP скрипти виявилися швидше аналогічних CGI-скриптів, так як сервера не було необхідності породжувати новий процес. Мова PHP по можливостях наблизився до Perl, найпопулярнішому мови для написання CGI-програм.

Була додана підтримка багатьох відомих баз даних (наприклад, MySQL і Oracle). Інтерфейс до GD – бібліотеці, дозволяв генерувати картинки на льоту. З цього моменту почалося широке поширення PHP / FI.

В кінці 1997 Зеев Сураські (Zeev Suraski) і Енді Гутманс (Andi Gutmans) вирішили переписати внутрішній движок, з метою виправити по-

милки інтерпретатора та підвищити швидкість виконання скриптів. Через півроку, 6 червня 1998 року вийшла нова версія, яка була названа PHP 3. До літа 1999 року PHP 3 був включений в кілька комерційних продуктів. За даними NetCraft на листопад 1999 PHP використовувався в більш ніж 1 млн. доменах.

PHP – одна з небагатьох мов програмування, створених спеціально для розробки веб-додатків. Тому вона включає в себе всі функції, необхідні саме для роботи на веб-сервері, і при цьому позбавлена надмірності, властивої багатьом його конкурентам.

Команди включаються в звичайні HTML-сторінки за допомогою спеціальних тегів, які і примушують PHP-машину виконувати на сервері потрібні дії. Програмами на PHP не потрібні спеціальні CGI-директорії з особливими правами доступу. Більш того, на одній сторінці можна довільно чергувати "простий" HTML і PHP-код.

PHP не залежить від платформи. PHP прекрасно інтегрується в усі популярні веб-сервери: Apacce і IIS, Zens і Netscape Enterprise Server, працює під Windows і OS / 2, MacOS і практично всіма UNIX-подібними системами. Як наслідок – PHP працює практично у всіх хостерів, які дозволяють власні виконувані скрипти [7].

PHP інтегрований практично з усіма сучасними інтернет-технологіями. PHP підтримує більшість сучасних веб-протоколів: IMAP, FTP, POP, XML, SNMP і інші. PHP прекрасно працює з базами даних. Важко знайти СКБД, підтримка якої не була б реалізована в PHP. MySQL і MS SQL Server, PostgreSQL та Oracle, Sybase і Interbase. Один тільки перелік баз даних, підтримуваних PHP, займе, напевно, цілий екран.

PHP включає в себе величезну кількість вбудованих функцій: обробки рядків і масивів, роботи з файловою системою і з HTTP, електронної поштою, датою і часом, кирилицею та іншими національними алфавітами. Завдяки їм багато алгоритми, що вимагають в більшості мов написання програмного коду розміром в декілька екранів, реалізуються на PHP однією командою (точніше, викликом однієї функції).

Сучасні тенденції розвитку мов програмування не обійшли стороною і PHP. Засоби об'єктно-орієнтованого програмування з'явилися ще в PHP3. А в об'єктної моделі PHP4 в повному обсязі реалізовані класичні поняття об'єктно-орієнтованого програмування: успадкування, інкапсуляція і поліморфізм [8].

PHP-фреймворк – це основа майбутнього програми, набір добре налагодженого коду для вирішення завдань, найбільш часто стоять перед розробниками сайтів. На основі фреймворків можна розробити не тільки окреме веб-додаток, а й оригінальну CMS, хоча на сучасному етапі це навряд чи має сенс.

Популярні представники: Yii, Zend Framework, Symfony2, Laravel, Phalcon, Codeigniter, Kohana.

PHP фреймворки за останнім часом набрали популярність, і стали базовою платформою для розробки веб-додатків. Іншими словами можна сказати, що вони забезпечують основну структуру програми.

Використання PHP-фреймворків, дозволяє економити велику кількість часу, зменшити навантаження на процес розробки, позбавляючи від проблеми повторюваного коду, і швидко створювати додатки. Без використання PHP-фреймворків, ставати набагато складніше створювати веб-додатки, супроводжувати і модернізувати їх. Тим часом, використання PHP фреймворків робить процес створення програми набагато легшим і функціональним.

Зараз більшість PHP проектів побудовані за допомогою архітектури Model View Controller (MVC). MVC – це архітектурний шаблон проектування, який використовується в більшості мов програмування і дозволяє відокремити бізнес-логіку від призначеного для користувача інтерфейсу, а також виділити область логіки, яка виробляє обмін інформацією між базою даних і призначеним для користувача інтерфейсом. Таким чином можна змінити логіку додатка, не зачіпаючи інтерфейсної частини, або навпаки, що дуже добре для дизайнерів і верстальників. Це дозволяє уникнути плутанини і спрощує весь процес розробки. Коли йдеться про MVC, то мається на увазі: Model – та частина архітектури, яка взаємодіє з базою даних, View – представляє ту частину, яку безпосередньо бачить користувач, тобто графічний інтерфейс, і Controller – це область логіки, яка контролює і управляє всіма її складовими і даними. Більшість сучасних фреймворком беруть за основу саме архітектуру MVC. Так само в сучасному фреймворку використовується шаблон проектування Front Controller, який, в залежності від запиту, перенаправляє його на потрібний контролер. Без Front Controller розробка із застосуванням фреймворка не мала б сенсу.

Для того щоб скористатися всіма можливостями фреймворка, потрібен чималий багаж знань в розробці додатків. PHP-фреймворки можуть допомогти усунути дуже часту помилку при програмуванні додатків, а саме повторення коду, а також систематизувати процес розробки. Фреймворки є потуж-

ним інструментом для динамічного мови програмування як PHP, які допоможуть організувати ваш код.

Кожна людина має різні вподобання і потреби. Для одного розробника використання PHP-фреймворків може допомогти в прискоренні процесу програмування, а для іншого це може здатися марною тратою часу. У більшості випадків це залежить від рівня професіоналізму, але, в загальному, PHP фреймворки призначені, щоб заощадити час і абстрагуватися від рутинних завдань.

В основному, PHP-фреймворки застосовуються для розробки проектів складніше ніж 2-х-3-х сторінковий сайт з текстовими сторінками.

Після проведення невеликого аналізу і читання відгуків про фреймворками від великого числа розробників різних рівнів, з точки зору зручності розробки, швидкості, стабільності, було виділено 6 популярних PHP-фреймворків, які відповідають більшості вимог.

Zend framework – це PHP-фреймворк, створений і підтримуваний компанією Zend, співробітники якої є безпосередніми авторами мови PHP. Тому він наслідує традиції і духу PHP – базується на простоті, об'єктно-орієнтованих принципах, дружній ліцензії і ретельно тестируемом коді із застосуванням agile методів.

CakePHP є швидко розвиваються фреймворком для PHP, який надає расширяемую архітектуру для розробки, обслуговування та розгортки вебдодатків. Використовує відомий шаблон проектування MVC, як і в об'єктно-реляційних фреймворками. Основний парадигмою CakePHP є збільшить продуктивність розробки і допомагає програмісту писати менше коду. Спочатку створювався як клон популярного Ruby on Rails, і багато ідей були запозичені саме звідти.

Проект Kohana був створений як гілка PHP-фреймворку CodeIgniter під ім'ям Blue Flame. Головною причиною відгалуження був перехід до більш відкритої для громадськості моделі розробки, тому, що багато користувачів були незадоволені швидкістю розробки і виправлення помилок в CodeIgniter. Rick Ellis – творець і власник CodeIgniter – підштовхнув новий фреймворк до створення власної документації і порадив перейменувати проект. У липні 2007 Blue Flame був перейменований в Kohana для того, щоб уникнути проблем з авторськими правами в майбутньому.

CodeIgniter – популярний MVC фреймворк з відкритим вихідним кодом, написаний на мові програмування PHP, для розробки повноцінних веб-

систем і додатків. Розроблено компанією EllisLab, а також Ріком Еллісом (Rick Ellis) і Полом Бурдик (Paul Burdick).

Yii – це високоефективний заснований на компонентної структурі PHP-фреймворк для розробки масштабних веб-додатків. Він дозволяє максимально застосувати концепцію повторного використання коду і може істотно прискорити процес веб-розробки. Назва Yii (вимовляється як Yee або [ji:]) означає простий (easy), ефективний (efficient) і розширюваний (extensible). Так само автор фреймворка, Qiang Xue каже, що назва фреймворка спочатку означало Yes It Is.

При виборі PHP фреймворку, можна трохи заплутатися з тим, що він повинен робити, і з тим для чого призначений фреймворк і що він виконує. Не кожен фреймворк підтримує ORM-шар для роботи з базами даних, має якісне співтовариство і хорошу документацію. Це може не перешкодити якщо потрібен простий фреймворк. Однак, якщо необхідний фреймворк який би зручний і простий в освоєнні, то необхідно ретельно підійти до питання вибору фреймворка і зважити всі «за» і «проти».

3.3.2 Переваги та недоліки PHP

Основними перевагами мови PHP є:

– традиційність – синтаксис мови дуже нагадує C / C ++ який є найбільш популярним, що дає можливість легко засвоєння цієї мови;

– простота – сценарій PHP може складатися з 10 000 рядків або з одного рядка – все залежить від специфіки завдання. Механізм PHP просто починає виконувати код після першої екрануючої послідовності (<?) І продовжує виконання до того моменту, коли він зустріне парну екранує послідовність (?>). Крім цього, код PHP може бути вбудований прямо в HTML розмітку;

– ефективність – движок PHP не є ні компілятором, ні інтерпретатором. Він є транслює інтерпретатором. Такий пристрій «движка» PHP дозволяє обробляти сценарії з достатньо високою швидкістю.

– гнучкість – є вбудовуваним (embedded) мовою, він відрізняється винятковою гнучкістю по відношенню до потреб розробника. Хоча PHP зазвичай рекомендується використовувати в поєднанні з HTML, він з таким же успіхом інтегрується і в JavaScript, WML, XML та інші мови. Крім того, добре структуровані додатки PHP легко розширюються в міру необхідності (втім, це відноситься до всіх основних мов програмування);

– безкоштовне розповсюдження – прийняття стратегії Open Source і безкоштовне розповсюдження початкових текстів PHP надало неоціненну послугу користувачам. До того ж, чуйне співтовариство користувачів PHP є свого роду «колективної службою підтримки», і в популярних електронних конференціях можна знайти відповіді навіть на найскладніші питання.

Однак на увазі своїх привілеїв таких як безкоштовність і загальної історії розвитку і початкової задумки мову PHP має такі недоліки:

– загальне низька якість коду – з причини своєї безкоштовності і низького порогу входження нові розробники швидко починають писати код, не замислюючись про правильність принципах написанні коду;

– неузгоджений синтаксис функцій і неортогональної – PHP надає розробникам велику кількість найрізноманітніших функцій, які потрапили в мову з розширень, створених різними групами програмістів. В результаті синтаксис мови не узгоджений, наприклад, частина функцій для роботи з масивами починається з префікса array_, інша частина цим префіксом не володіє. Назви частини строкових функцій починається з префікса str або str_, інші функції таким префіксом не володіють. У тих же строкових функціях обробляється рядок може передаватися як в якості першого, так і в якості останнього аргументу, що викликає плутанину у програмістів, і, отже, вимагає постійного звернення до документації. Деякі завдання, наприклад, розбиття рядка на масив або підрядка, вирішуються кількома функціями;

– відсутність зворотної сумісності між версіями мови – код, створений для більш ранніх версій мови, часто не працює або працює некоректно з більш пізніми версіями мови. У більш пізніх версіях виключаються конструкції, методики, функції, що застосовувалися раніше. В результаті, додатки, створені кілька років тому, практично втрачають працездатність для сучасних версій мови і вимагають значної модифікації. Такі зміни обумовлені двома факторами: усуненням неузгодженого синтаксису і усуненням конструкцій, які заохочують створення небезпечного коду. У версіях лінійки 5.3.x велику кількість функцій було визнано застарілими, їх підтримка не планується в нових версіях мови, що викликає несумісність зі скриптами, які використовують застарілі функції;

– відсутність підтримки мультибайтних символів в ядрі мови – Підтримка рядків з багатобайтове кодування, такими, як UTF-8, реалізується через окремі розширення mbstring і iconv, на рівні ядра підтримка відсутня, проте з версії PHP 4.2.0 є можливість перевизначати стандартні функції роботи зі рядками, підміняючи їх на аналоги з mbstring. Підтримка мультибайт-

них символів у всіх строкових функціях стала доступна з версії 5.4.0 (березень 2012 року);

– відсутність підтримки багатопоточності – в мові не передбачена можливість створення багатопоточних додатків і відсутня підтримка синхронізованого доступу до ресурсів, проте реалізувати за допомогою розширення PCNTL. Однак слід зазначити, що використання PCNTL або ProcessControl не підходить для вирішення специфічних завдань.

Недоліки PHP-фреймворків:

– складність вивчення. Як мінімум, необхідно добре знання PHP;

– висока вартість розробки. Будувати будинок з цеглинок набагато довше і складніше, ніж з готових блоків;

– відсутність адміністративного модуля. Необхідно самостійно створювати сторінки для керування вмістом, авторизації, текстові редактори і т. д;

– дороге подальше обслуговування. Розвиток або супровід готового сайту є трудомісткою завданням навіть для його розробника. Із залученням інших програмістів іноді простіше розробити новий сайт, ніж доопрацювати вже існуючий.

Переваги PHP-фреймворків:

– висока продуктивність коду. Швидше працювати можуть тільки сайти, повністю написані на PHP;

– безпека – фреймворки пишуться одними досвідченими програмістами для інших програмістів і ретельно тестуються вся спільнота. Це дозволяє вчасно помітити недоліки коду з точки зору безпеки і усунути помилки;

– гнучкість – фреймворки дозволяють вирішувати практично будь-які завдання. Є можливість використання готових класів та бібліотек, написаних іншими програмістами.

PHP розвиваюча мова програмування в якої є величезна кількість функцій і технологій «з коробки», проте вона поступається в швидкості роботи компільованим мовам програмування що особливо відчутно у великих системах, однак для невеликих проектів php є легковажним і швидким рішенням.

3.4 Технології побудови інтерфейсу

3.4.1 Використання css фреймворків для побудови інтерфейсу

CSS – технологія опису зовнішнього вигляду документа, оформленого мовою розмітки.

Переважно використовується як засіб оформлення веб-сторінок в форматі HTML і XHTML, але може застосовуватися з будь-якими видами документів в форматі XML, включаючи SVG і XUL.

Каскадні таблиці стилів використовуються творцями веб-сторінок для завдання кольорів, шрифтів, розташування і інших аспектів представлення веб-документа. Основною метою розробки CSS було розділення вмісту (написаного на HTML або іншій мові розмітки) і оформлення документа (написаного на CSS). Це поділ може збільшити доступність документа, надати велику гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне уявлення, друк, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), або при виведенні пристроями, що використовують шрифт Брайля [9].

Каскадні таблиці стилів – це стандарт, який визначає уявлення даних в браузері. Якщо HTML надає інформацію про структуру документа, то таблиці стилів повідомляють як він повинен виглядати.

Стиль – це сукупність правил, що застосовуються до елементу гіпертексту і визначають спосіб його відображення. Стиль включає всі типи елементів дизайну: шрифт, фон, текст, кольори посилань, поля і розташування об'єктів на сторінці.

Таблиця стилів – це сукупність стилів, які можна застосувати до гіпертекстових документів [10].

Каскадування – це порядок застосування різних стилів до веб-сторінці. Браузер, що підтримує таблиці стилів, буде послідовно застосовувати їх відповідно до пріоритетом: спочатку пов'язані, потім впроваджені і, нарешті, вбудовані стилі. Інший аспект каскадирования – успадкування (inheritance) означає, що якщо не вказано інше, то конкретний стиль буде застосований до всіх дочірнім елементами гіпертекстового документа. Наприклад, якщо ви застосуєте певний колір тексту в тезі , то все теги всередині цього блоку будуть відображатися цим же кольором.

Використання каскадних таблиць дає можливість розділити вміст і його уявлення і гнучко управляти відображенням гіпертекстових документів шляхом зміни стилів [11].

CSS-фреймворк – фреймворк, створений для спрощення роботи верстальника, швидкості розробки і виключення максимально можливого числа помилок верстки (проблеми сумісності різних версій браузерів і т. д.). Як і бібліотеки скриптових мов програмування, CSS-бібліотеки, зазвичай мають вигляд зовнішнього css-файлу, «підключаються» до проекту (додаються в заголовок веб-сторінки).

Умовно можна виділити два типи css фреймворків: всеосяжні і обмежені. Третім варіантом може бути розробка власного фреймворка. Цей варіант віддає перевагу більшість розробників, так як це дає вигоди персонального рішення і зменшує негативні моменти залежності від використання сторонніх рішень.

Всеосяжні фреймворки намагаються охопити більшість речей, які можуть знадобитися розробнику. До цього типу належать бібліотеки, які включають CSS для верстки і скидання (або якусь основу).

Обмежені фреймворки охоплюють лише обмежена кількість речей необхідних розробнику для вирішення конкретних завдань.

Як css фреймворка був вибрав всеоб'емлюющій фреймворк Bootstrap містить великий набір функціоналу необхідний для реалізації інтерфейсу особистого кабінету, також має великий набір прикладів і детальну документацію [12].

Bootstrap – вільний набір інструментів для створення сайтів і вебдодатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів вебінтерфейсу, включаючи JavaScript-розширення.

Основні інструменти Bootstrap:

– сітки – заздалегідь задані розміри колонок, які можна відразу ж використовувати, мають ширину, яка визначається батьківським елементом (рис 3.1);

– шаблони – фіксований або гумовий шаблон документа;

– типографіка – опису шрифтів, визначення деяких класів для шрифтів, таких як код, цитати тощо;

– медіа – представляє деякий спосіб упорядкування зображень та відео;

– таблиці – кошти оформлення таблиць, додавання сортування;

– форми – класи для оформлення форм і деяких подій, що відбуваються з ними;

– навігація – класи оформлення для панелей, вкладок, переходу по сторінках, меню і панелі інструментів;

– алерти – оформлення діалогових вікон, підказок і спливаючих вікон.

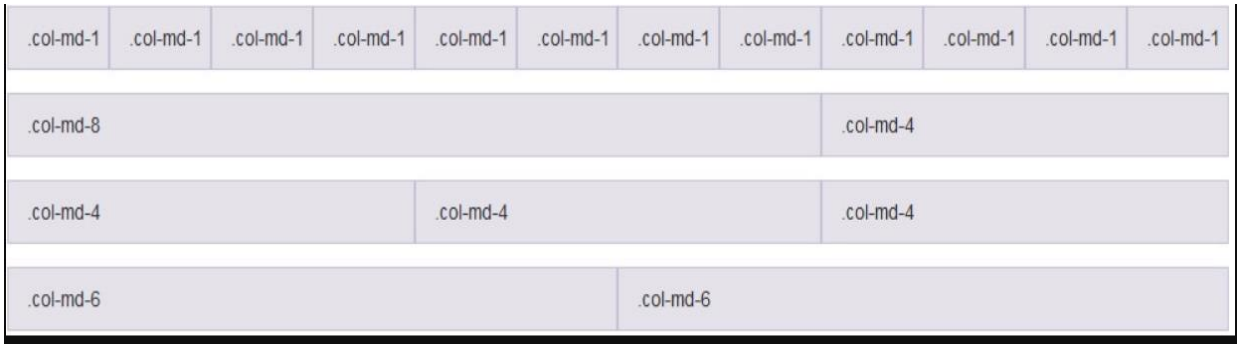


Рисунок 3.1 – Система сіток Bootstrap

Ця бібліотека почала розроблятися як внутрішня бібліотека компанії Twitter під назвою Twitter Blueprint. Після кількох місяців розробки він був відкритий під назвою Bootstrap 19 серпня 2011 року.

Основними нововведеннями другої версії, що з'явилася 31 січня 2012 року, стали 12-колоночная сітка і підтримка адаптивності.

Третя версія випущена 19 серпня 2013 року. У ній адаптивність отримала подальший розвиток, був здійснений перехід до концепції mobile first, оптимізації перш за все під мобільні пристрої. Дизайн за замовчуванням став плоским.

Робота над четвертою версією розпочато 29 жовтня 2014 року. Альфа версія вийшла 19 серпня 2015 року. Перша бета версія випущена 10 серпня 2017. Друга бета версія випущена 19 жовтня 2017. 18 січня 2018 випущена перша стабільна версія Bootstrap 4.

В ході реалізації особистого кабінету була обрана версія Bootstrap 3, яка не є самою останньою, однак найбільш стабільна і давно стандартизована.

3.4.2 Програмування елементів інтерфейсу на мові javascript

Javascript – це мова веб-програмування для надання інтерактивності веб сторінок. З його допомогою створюються додатки, які включаються в HTML-

код (наприклад, анкети або форми реєстрації, які заповнюються користувачем).

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

За допомогою Javascript можна змінювати сторінку, змінювати стилі елементів, видаляти або додавати теги. Дає можливість дізнатися про будь-яких маніпуляціях користувача на сторінці (прокрутка сторінки, натискання будь-якої клавіші, кліки мишкою, збільшення або зменшення робочої області екрану). Надає можливості доступу до будь-якого елементу HTML-коду і робити з цим елементом безліч маніпуляцій. Дозволяє завантажувати дані без перезавантаження сторінки, виводити повідомленням, зчитувати або встановлювати cookie і виконувати безліч інших дій.

Вся унікальність даного мови програмування полягає в тому, що він підтримується практично всіма браузерами і повністю інтегрується з ними, а все що можна зробити з його допомогою – робиться дуже просто. Жодна інша технологія не вміщає в собі всі ці переваги разом. Наприклад, є такі, що не кросбраузерну (тобто підтримуються не всіма браузерами). Це – VBScript, ActiveX, XUL. А є такі, які з браузером не інтегровані в потрібному ступені, це – Java, Flash, Silverlight. На сьогоднішній день дана технологія активно розвівається, розробляється мову програмування Javascript [13].

Назва "JavaScript" є власністю Netscape. Реалізація мови, здійснена розробниками Microsoft, офіційно називається Jscript. Версії JScript сумісні (якщо бути зовсім точним, то не до кінця) з відповідними версіями JavaScript, тобто JavaScript є підмножиною мови JScript.

JavaScript стандартизований ECMA (European Computer Manufacturers Association – Асоціація європейських виробників комп'ютерів). Відповідні стандарти носять назви ECMA-262 і ISO-16262. Ці стандарти вимагають визначається мова ECMAScript, який приблизно еквівалентний JavaScript 1.1. Відзначимо, що не всі реалізації JavaScript на сьогодні повністю відповідають стандарту ECMA. В рамках даного курсу ми у всіх випадках будемо використовувати назву JavaScript.

Об'єктна модель документа надає Web-розробникам воістину величезні можливості. Однак, щоб користуватися об'єктною моделлю документа, Web-розробник повинен вміти писати сценарії, особливо на мові JavaScript [14].

Причина в тому, що об'єктна модель документа є всього лише відображенням Web-сторінки і сама по собі нічого не означає. Щось схоже відбу-

ється зі змінними. Змінна може зберігати в собі ту чи іншу значення, але без сценарію JavaScript ні на що не придатна.

Остання версія об'єктної моделі документа містить чотири ключові нововведення:

– доступ до всіх елементів сторінки;

– постійне оновлення сторінки;

– повна і всеосяжна модель подій;

– оперативну зміну вмісту сторінки.

І ще одна головна особливість – зміни на Web-сторінці можуть відбуватися будь-який час. До, під час і після завантаження сторінки, коли користувач натискає кнопку, клацає кнопкою миші, переміщує курсор і т. д.

JS-framework'i – це інструменти для побудови динамічних веб / мобільних / настільних додатків на мові Javascript. Як і до будь-яких інших інструментів, розробники вдаються до використання js-фреймворків там, де неможливо, дуже складно або дуже довго виконувати завдання звичайними засобами. У переважній більшості випадків, фреймворки використовуються для написання, так званих, Single Page Applications. Тобто все, що проиходит на сайті, проиходит на одній сторінці, без прямого перехода з неї.

З їх допомогою можна розробляти як повноцінні сайти, так і функціональні модулі (різні онлайн-інструменти). Звичайно, повноцінні фреймворки краще підходять для першого завдання, а для другої рекомендується використовувати більш легковагі фреймворки або бібліотеки.

Переваги побудови програми на JS-фреймворку:

– можна легко реалізувати SPA (Single Page Application);

використання js-фреймворка зобов'язує нас мати структуру програми (скажімо рішуче «ні» спагетті-коду);

– код стає помітно менше, і він чистіше, що позитивно відбивається на швидкості розробки, а також підтримки та усунення помилок в коді програми;

– наявність структури має на увазі модульність додатков, а це дає можливість простіше працювати над додатком кількох розробникам одночасно;

– наступне перевагу більше випливає з використання самого javascript, але значно посилюється при використанні фреймворка: можливість швидко створити мобільне і / або настільний кроссплатформне додаток з веб-версії з поміччю систем типу PhoneGap або Apache Cordova.

jQuery – бібліотека JavaScript, що фокусується на взаємодії JavaScript і HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого

елементу DOM, звертатися до атрибутів і вмісту елементів DOM, манипул вати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX.

Автор бібліотеки Джон Резіг вперше представив своє творіння в січні 2006 року на комп'ютерній конференції в Нью-Йорку, а в серпні того ж року була випущена перша стабільна версія бібліотеки.

За минулі роки бібліотека зазнала безліч змін і на поточний день містить функціонал, корисний для максимально широкого кола завдань.

Функції jQuery:

– звернення до будь-якого елементу DOM (об'єктній моделі документа) і не тільки звернення, а й маніпуліляція;

– робота з подіями;

– здійснення різних візуальних ефектів;

– робота з AJAX (дуже корисна технологія, що дозволяє спілкуватися з сервером без перезавантаження сторінки);

– величезна кількість JavaScript плагінів, призначених для створення елементів призначених для користувача інтерфейсів.

Так само, як і CSS відокремлює візуалізацію від структури HTML, JQuery відокремлює поведінку від структури HTML. Наприклад, замість прямої вказівки на обробнику події натискання кнопки, управління передається JQuery, який ідентифікує кнопки і потім перетворює його в обробник події кліка. Такий поділ поведінки і структури також називається принципом ненавьязного JavaScript [15].

Бібліотека jQuery містить функціональність, корисну для максимально широкого кола завдань. Однак, розробниками бібліотеки не ставилася завдання суміщення в jQuery функцій, які підійшли б усюди, оскільки це призвело б до великого коду, велика частина якого не затребувана. Тому була реалізована архітектура компактного універсального ядра бібліотеки і плагінів. Це дозволяє зібрати для ресурса саме ту JavaScript-функціональність, яка на ньому була б затребувана.

Основну функціональність бібліотеки jQuery виконує функція jQuery, яка має псевдонім $. Це єдині ідентифікатори, які доступні в глобальному контексті, всі інші функції знаходяться в області імен бібліотеки jQuery.

З істотних недоліків можна виділити тільки тимчасово неповну підтримку пошуковими системами, але ця задача рідко збігається з завданням по реалізації SPA (Single Page Application), тим більше, що провідні пошукові системи (як мінімум, Google), вже практично повністю вирішили цю проблему.

На сьогоднішній день використання javascript, його великих можливостей, велика кількість інструментів і готових бібліотек для пожвавлення веб-сторінки і забезпечення додаткової зручності користувачеві є фактично обов'язковим при розробці будь-якого веб-сайту.

4 РОЗРОБКА ОСОБИСТОГО КАБІНЕТУ СПОЖИВАЧА ТЕПЛОВОЇ ЕНЕРГІЇ

4.1 Реалізація процесу реєстрації користувача в системі

Для реєстрації користувача йому необхідно вказати основну інформацію, за допомогою якої він буде ідентифікований в системі і здійснює процес авторизації. Форма реєстрації вказана на рис. 4.1.

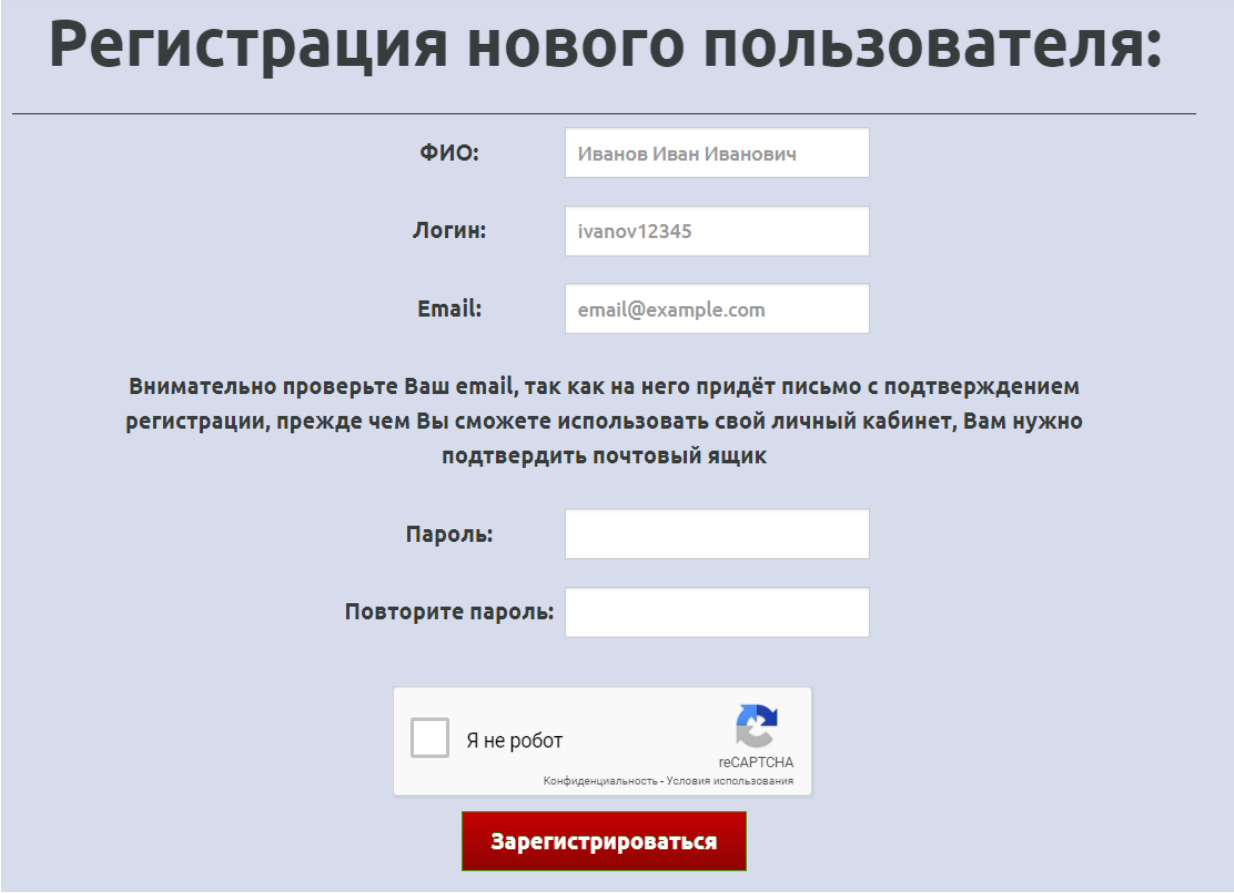


Рис. 4.1 – Форма реєстрації користувача

Для проходження процесу реєстрації необхідно вказати: ПІБ для звернення до користувачеві в системі, логін по якому буде здійснюватися авторизація, є унікальним на всю систему (рис 4.2), email на який прийде лист для підтвердження інформації про користувача і завершення процесу реєстрації, пароль і підтвердження пароля для запобігання помилок при вказівки пароля користувачем, капча для запобігання заповнення форми реєстрації сторонніми програмами, в тому числі запобігання навантажень на сайт цими програ-

мами. Приклад правильно заповненої форми з проходженням капчи вказано на рис 4.3.

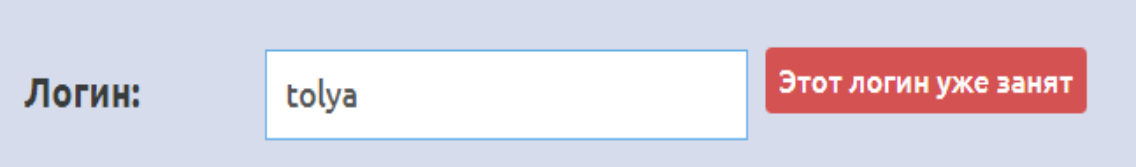


Рис. 4.2 – Запобігання повторного використання логіна в системі

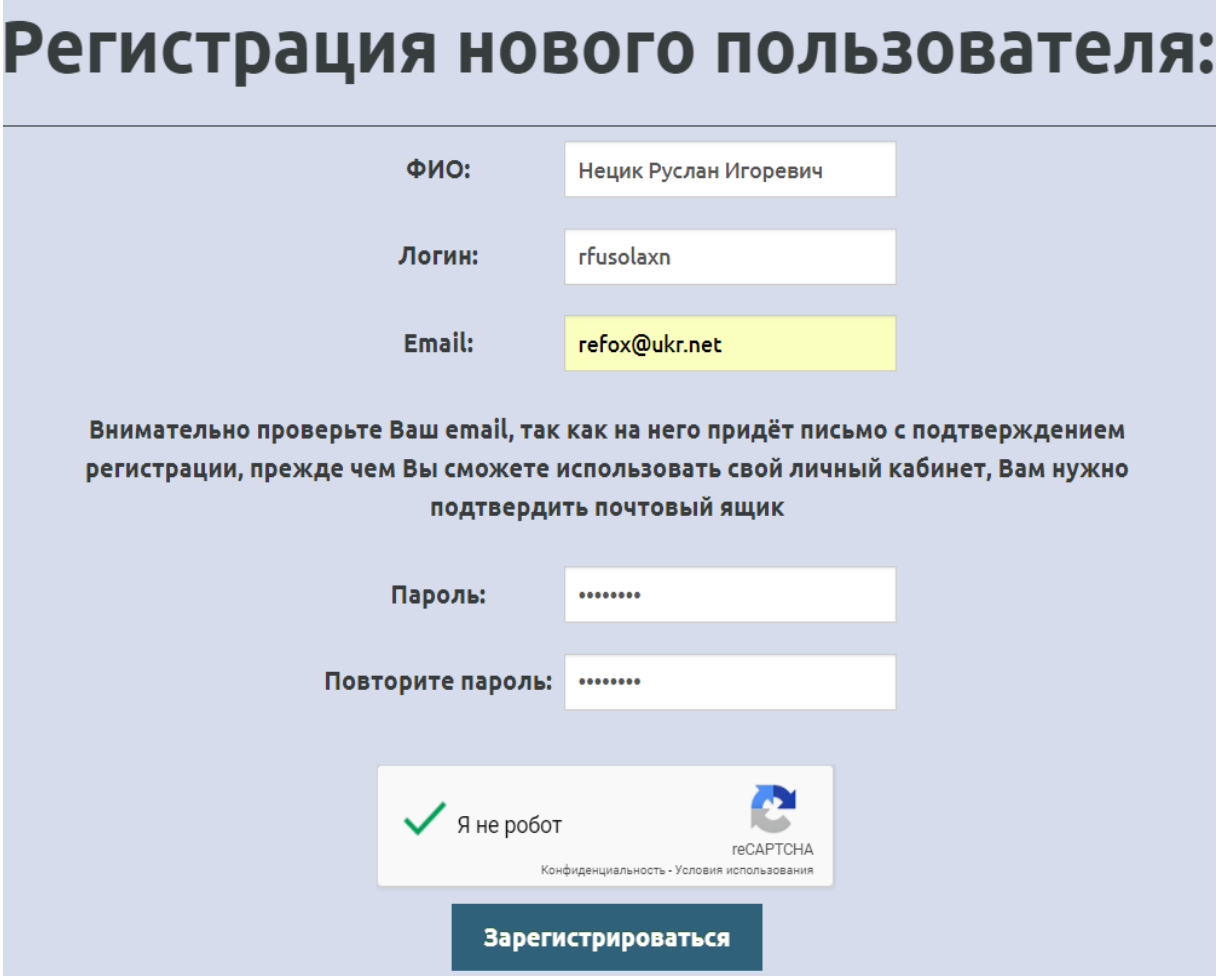


Рис. 4.3 – Заповнена форма реєстрації

Після заповнення форми реєстрації для підтвердження email на нього приходить лист містить посилання для активації облікового запису (рис 4.4).

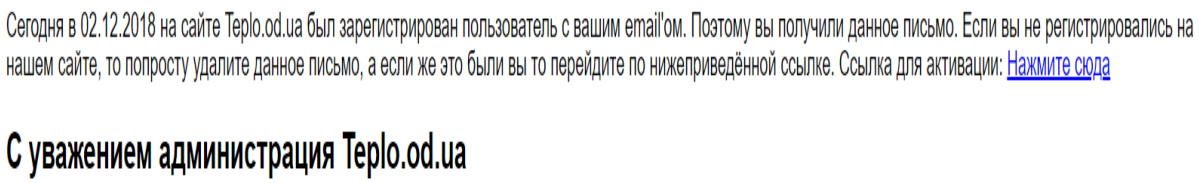


Рисунок 4.4 – Лист для підтвердження реєстрації

Після натискання на посилання користувачеві стає доступним процес аторізації в особистий кабінет.

Процес авторизації представляє собою введення логіна і пароля які були вказані в процесі реєстрації (рис 4.5).

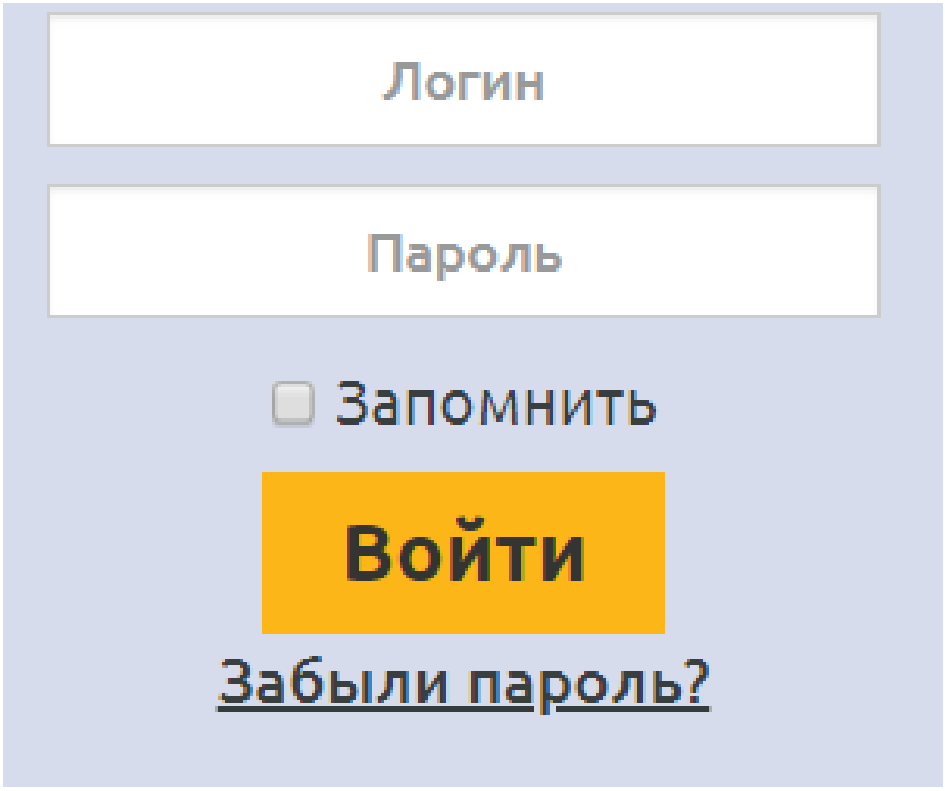


Рисунок 4.5 – Форма авторизації

Для користувачів які забули логін або пароль реалізований функціонал відновлення даних для авторизації. Для відновлення своїх ідентифікаційних даних користувачеві досить заповнити форму відновлення пароля складається з одного поля email (рис 4.6). Після заповнення форми користувачеві прийде лист на вказаний в формі email з його даними.

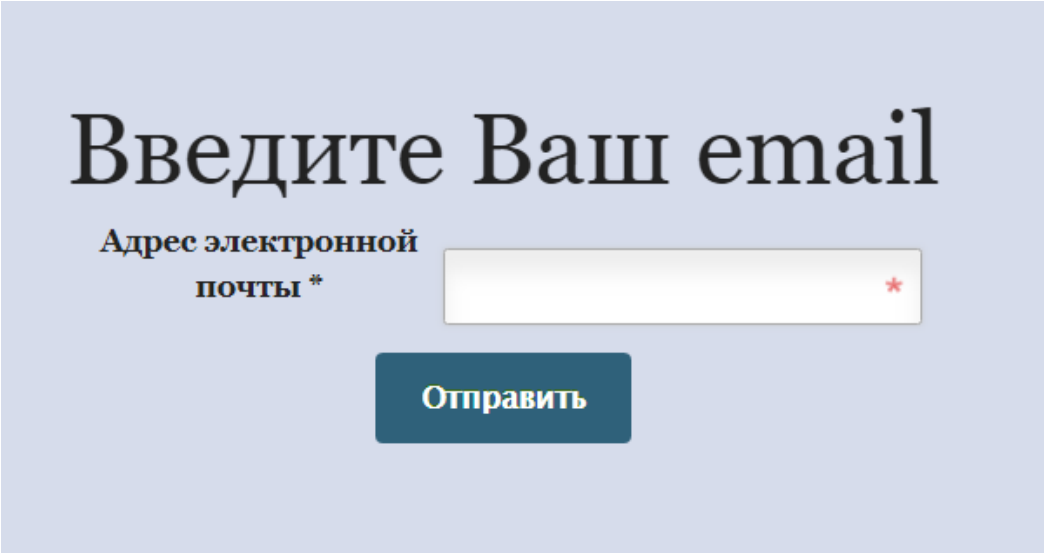


Рисунок 4.6 – Форма відновлення пароля

4.2 Реалізація особистого кабінету споживача теплової енергії

Після завершення реєстрації та авторизації користувачеві доступний весь закритий раніше функціонал особистого кабінету: додавання або видалення особового рахунку квартири, перегляд інформації про особовий рахунок, стан рахунку на кожен місяць, кожен рік у вигляді таблиць, друк квитанції станом заборгованості на особовому рахунку. Загальний вигляд інтерфейсу особистого кабінету вказано на рис 4.7.

Для початку роботи в особистому кабінеті користувачеві необхідно додати свій особовий рахунок. Кількість рахунків у користувача необмежену. Користувач у будь-який момент часу може додати або видалити особовий рахунок. Для додавання нового особового рахунку користувачеві необхідно ввести номер особового рахунку квартири, а також прізвище на якій цей рахунок оформлений рис 4.8.

Після заповнення форми додавання особового рахунку відбудеться пошук зазначеного рахунку в базі даних. Якщо рахунок вказаний правильно то користувачеві стане доступний перегляд інформації з цього особовому рахунку.

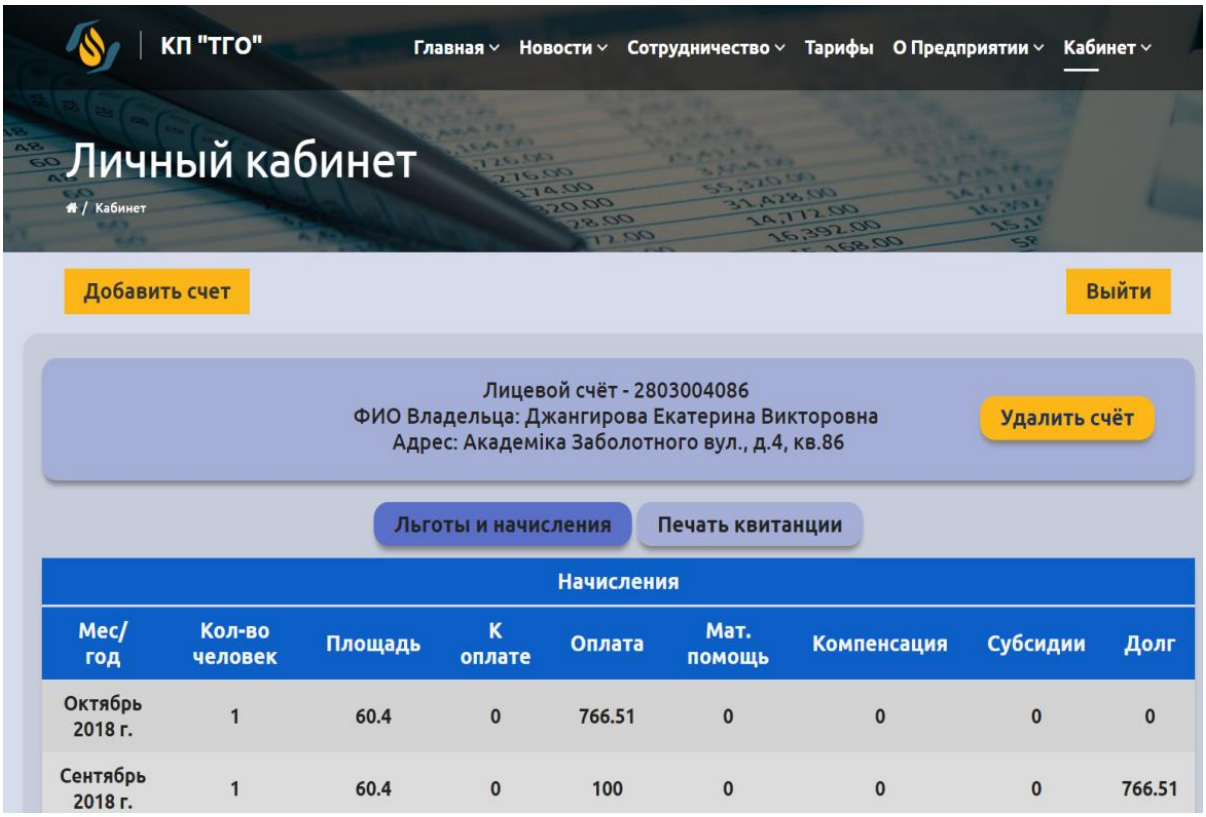


Рисунок 4.7 – Інтерфейс особистого кабінету

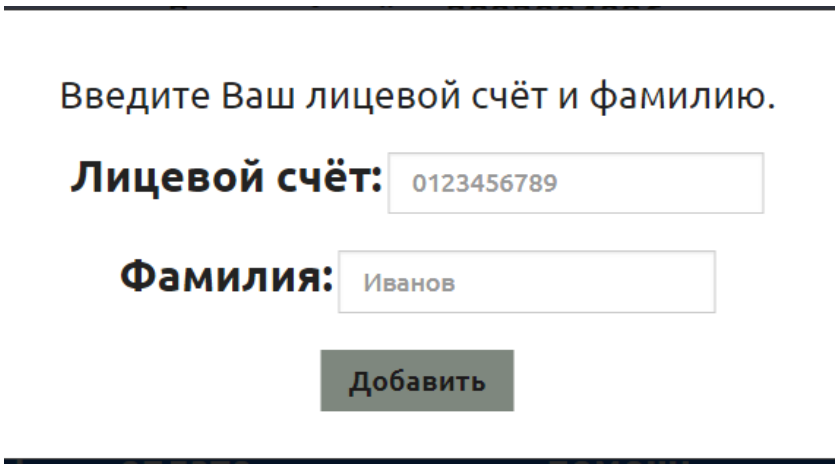


Рисунок 4.8 – Форма додавання особового рахунку

Загальний вигляд доданого особового рахунку вказаний на рис 4.9.

Верхня частина особового рахунку містить інформацію про власника (ПІБ), адреса на який посилається особовий рахунок, а також сам номер особового рахунку.

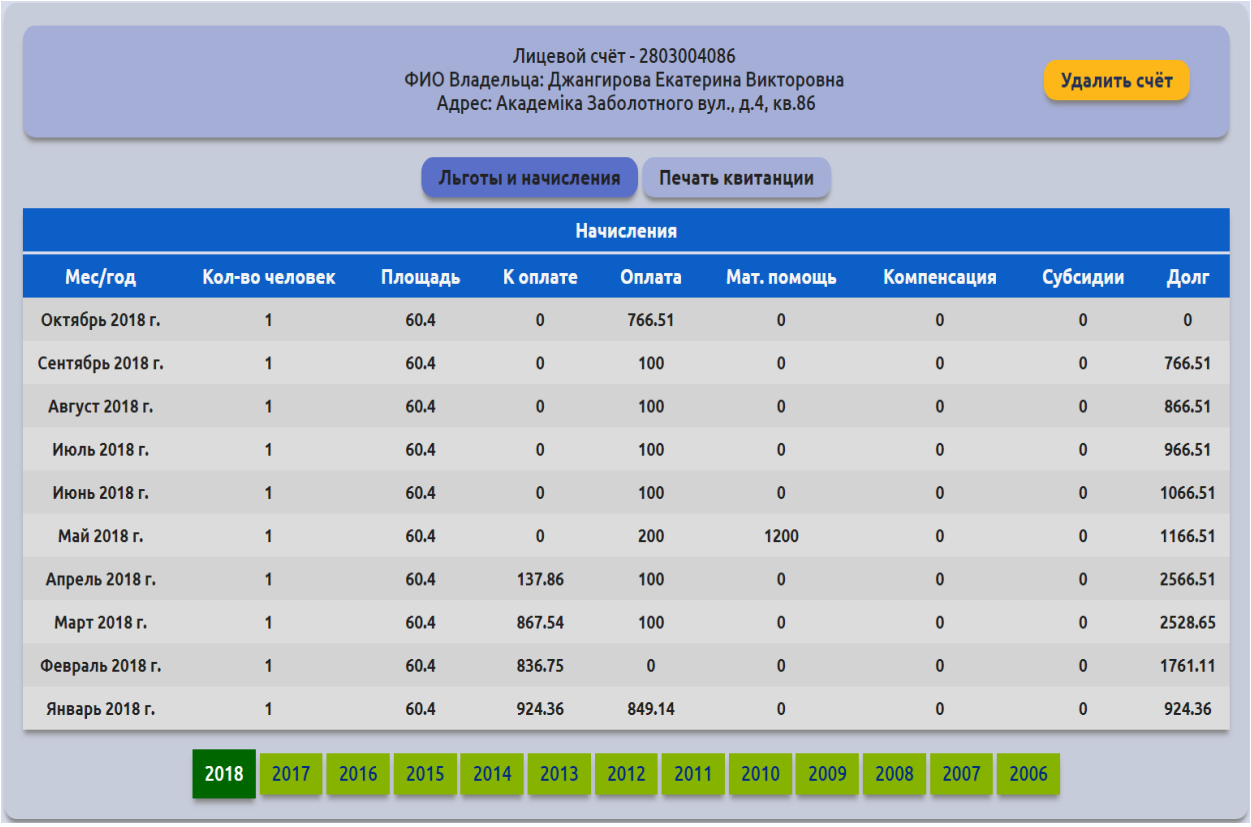


Рисунок 4.9 – Відображення особового рахунку в особистому кабінеті

Також містить кнопку видалення особового рахунку в разі якщо власник особистого кабінету більше не має відношення до цього особового рахунку (рис 4.10).

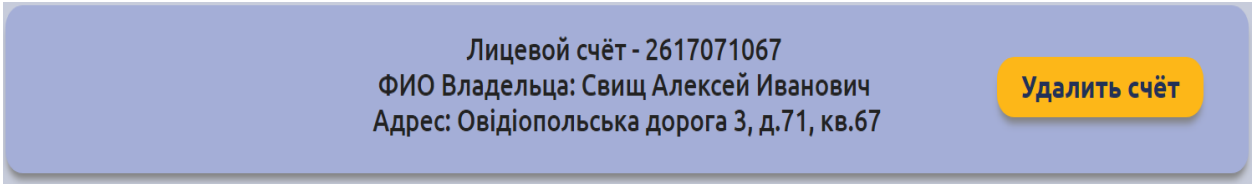


Рисунок 4.10 – Відображення загальної інформації по особовому рахунку

Нижче йде таблиця помісячних нарахувань по даному особовому рахунку поділена на колонки: назва місяця, кількість осіб, площа приміщення, необхідна сума до оплати, сума сплачена за цей місяць, загальна сума боргу з цього рахунку станом на кінець Етоо місяці, а також особлива інформація така як матеріальна допомога, компенсація, субсидії. Для зручність користувача нарахування поділені на року. Список всіх доступних для перегляду років

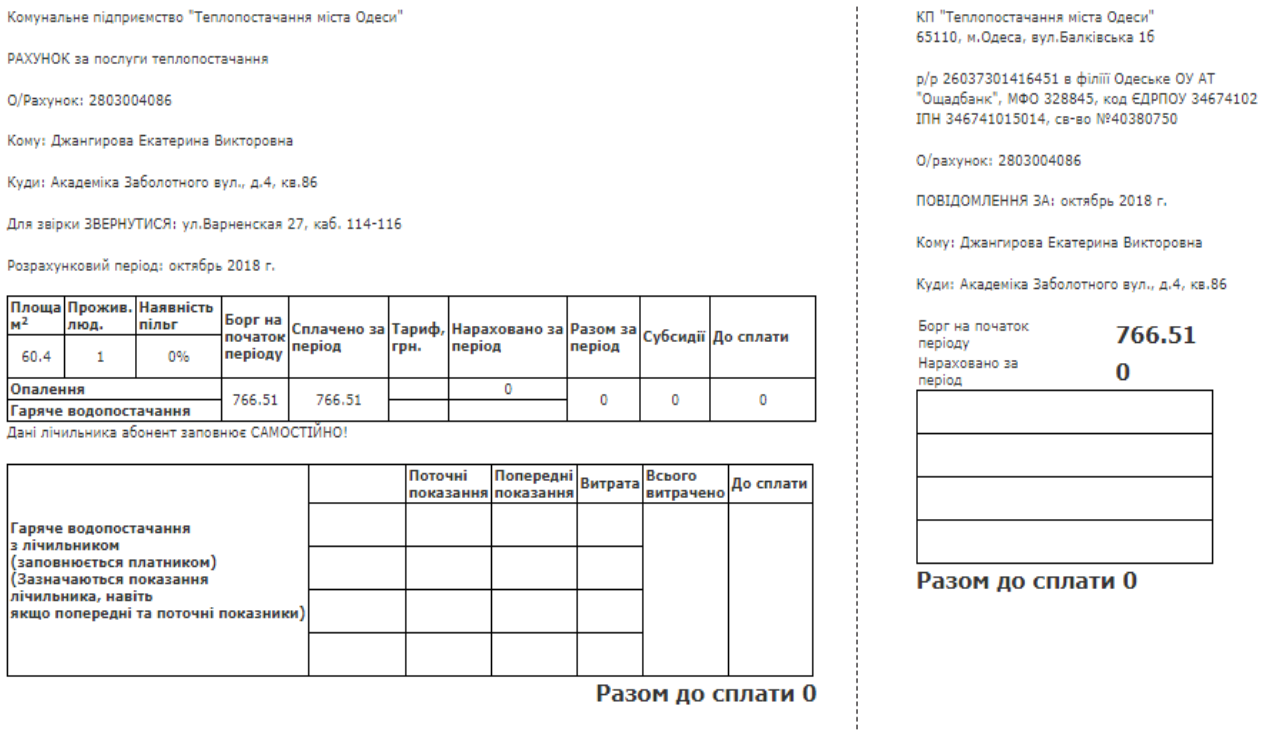
представлений у вигляді пагінацію, при натисканні на яку відкривається список нарахувань за відповідний рік без перезавантаження сторінки (рис. 4.11).

Рисунок 4.11 – Пагінація облікових періодів по роках

Кнопка «друк квитанції» являє собою функціонал побудови квитанції на основі суми загального боргу по даному особовому рахунку.

В рахунку вказана вся необхідна інформація про споживача тепла, його адреса, повне ім'я, номер особового рахунку, а також про постачальника тепла в тому числі і реквізити. Роздрукувавши дану квитанцію, споживач може без зусиль оформити банківський переказ на погашення боргу по своєму особистому рахунку.

Загальний вигляд квитанції в доступному для друку вигляді представлений на рис. 4.12.



Комунальне підприємство "Теплопостачання міста Одеси"

РАХУНОК за послуги теплопостачання

О/Рахунок: 2803004086

Кому: Джангирова Екатерина Викторовна

Куди: Академіка Заболотного вул., д.4, кв.86

Для звірки ЗВЕРНУТИСЯ: ул.Варненская 27, каб. 114-116

Розрахунковий період: октябрь 2018 г.

Площа м²	Прожив. люд.	Наявність пільг	Борг на початок періоду	Сплачено за період	Тариф, грн.	Нараховано за період	Разом за період	Субсидії	До сплати
60.4	1	0%							
Опалення			766.51	766.51		0	0	0	0
Гаряче водопостачання									

Дані лічильника абонент заповнює САМОСТІЙНО!

		Поточні показання	Попередні показання	Витрата	Всього витрачено	До сплати
Гаряче водопостачання з лічильником (заповнюється платником) (Зазначаються показання лічильника, навіть якщо попередні та поточні показники)						

Разом до сплати 0

КП "Теплопостачання міста Одеси"
65110, м.Одеса, вул.Балківська 16

р/р 26037301416451 в філії Одеське ОУ АТ "Ощадбанк", МФО 328845, код ЄДРПОУ 34674102
ІПН 346741015014, св-во №40380750

О/рахунок: 2803004086

ПОВІДОМЛЕННЯ ЗА: октябрь 2018 г.

Кому: Джангирова Екатерина Викторовна

Куди: Академіка Заболотного вул., д.4, кв.86

Борг на початок періоду 766.51

Нараховано за період 0

Разом до сплати 0

Рисунок 4.12 – Вигляд квитанції для оплати

Уся інформація що відображена у квитацнції генеруеться за допомогою збережаних процедур у віддаленій базі даних. Квитанція будується за допомогою HTML та CSS. Функціонал що написаний для цієї квитанції та інформація (ПІБ та номер особистого рахунку) потім буде використаний для онлайн оплати, що буде реалізовано у майбутньому часі.

ВИСНОВКИ

В ході виконання даної магістерської роботи був розроблений особистий кабінет споживача теплової енергії. При розробці були враховані вимоги КП «ТМО», проведено аналіз предметної області. Були виділені загальні риси, переваги і недоліки існуючих порталів з аналогічним функціоналом. Були оцінені можливості реалізації особистого кабінету з використанням доступних на сьогоднішній день технологій. Були описані їх переваги та недоліки. Деякі вибрані технології були обумовлені вимогами КП «ТМО», інші технології обрані для найкращого взаємодії з вже існуючими.

Розроблений особистий кабінет повністю відповідає опису особистого кабінету і вимогам своєї предметної області. Була реалізована можливість реєстрації, повністю підтримана можливість авторизації і захисту даних. Особистий кабінет відображає необхідну інформацію в доступному вигляді, а так само має можливість маніпулювати їй. У майбутньому в особистому кабінеті буде реалізована можливість онлайн-оплати, історія здійснених транзакцій та можливість їх друку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Хелен Б. Firebird. Руководство разработчика баз данных = The Firebird Book: A Reference for Database Developers. – СПб.: «БХВ-Петербург», 2007. – 1104 с.
2. Коннолли Т., Базы данных. Проектирование, реализация и сопровождение. Теория и практика– 3-е изд. – М.: «Вильямс», 2003. – 1436 с.
3. Компания MySQL AB. MySQL. Справочник по языку – М.: Издательский дом "Вильямс", 2005. – 432 с.
4. Васвани. В. А. «MySQL: использование и администрирование MySQL Database Usage & Administration.» – М.: «Питер», 2011. – 368 с.
5. Ульман Л. MySQL – М.: ДМК Пресс; СПб.: Питер, 2004. – 352с.
6. Кузнецов М.В., Симдянов И. В. MySQL5. – СПб.: БХВ-Петербург, 2006. – 1024 с.
7. Котеров Д. В., PHP5. – СПб.: БХВ-Петербург, 2005. – 1120 с.
8. Кухарчик А., PHP: обучение на примерах. – М.: Новое знание, 2004 г. – 240 с.
9. Дуванов А. А. Web-конструирования. – СПб.: БХВ-Петербург, 2009. – 384 с.
10. Крамер Е. HTML: наглядный курс Web-дизайна. – Киев: 2009. – 304 с.
11. Шафер С. HTML, XHTML и CSS. Библия пользователя, пятое издание = HTML, XHTML, and CSS Bible, 5th Edition. – М.: Диалектика, 2010. – 656 с.
12. Смирнова И. Е. Начала web-дизайна. – СПб.: БХВ-Петербург: 2010. – 491 с.
13. Ратбон Е., JavaScript для чайников. – М.: Диалектика, 1995. – 320 с.
14. Рейсиг. Д., Инструменты отладки и тестирования JavaScript. Профессиональные приемы программирования – Pro JavaScript ™ Techniques / Перевод Н. Волчанский. – СПб.: БХВ-Петербург, 2008. – 629с.
15. Фримен А. jQuery для профессионалов = Pro jQuery. – М.: «Вильямс», 2012. – 960 с.

Д О Д А Т К И

ДОДАТОК А
ПРОГРАМНИЙ КОД

```
<?php header("Content-Type: text/html; charset=utf-8");?>
<?php setlocale(LC_ALL, 'ru_RU.65001', 'rus_RUS.65001', 'Russian_Russia. 65001', 'russian');?>
<?php
$user=JFactory::getUser();
$us_id=$user->id;
$us_name = $user->name;
$email = $user->email;
if(!isset($_SESSION))
    {
  session_start();
            }
$TGOCALC =
  "(DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.18.5)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = TGOCALC)
    )
  )";
  $charSet = "AL32UTF8";  //UTF8: AL32UTF8 win-1251: CL8MSWIN1251
  $conn = oci_connect('hu_web_load', 'elfktyyjtcjtlbytybt', $TGOCALC, $charSet);
?>
<script type="text/javascript" src="jquery-1.11.0.js"></script>
<script type="text/javascript" src="/registerLdap/jquery-ui.js"></script>
<?php header("Content-Type: text/html; charset=utf-8");?>
<?php setlocale(LC_ALL, 'ru_RU.65001', 'rus_RUS.65001', 'Russian_Russia. 65001', 'russian');?>
<?php
if (!$conn) {
  //$e = oci_error();
  //trigger_error(htmlentities($e['message'], ENT_QUOTES), E_USER_ERROR);
  echo "Нет связи с источником данных";
}
else{
 //$idNumber = '3701131085'; $secondName = 'Савицкая'; //test values
 //$_SESSION['idNumber'] = '2617071067';
 //$_SESSION['secondName'] = 'Свищ';
//start MySQL query
 $link = mysqli_connect("localhost", "new_tgo", "V1rtu@l", "new_tgo");
/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}
 mysqli_set_charset($link, 'utf8');

$query_activ = mysqli_query($link, "SELECT block FROM hz3jh_users WHERE id=$us_id");
$activ = mysqli_fetch_row($query_activ);
```

```
if($activ[0]=='1')
{
        echo "<div class='User_data'><p>Здравствуйте, $us_name</p><p>Ваш e-mail: $email не был подтвержден, проверьте почту для подтверждения и получения доступу к сайту</p></div><br/><br/><br/>";
}
else
{
$query = "SELECT Code,SecondName FROM hz3jh_user_acc WHERE UserAccount=$us_id";
 ?>
 <div id='kepka'>
 <a class='button_add' href="#win1" >Добавить счет</a><br/>
 <!--<a class='button_add' href = '/index.php' >Вернуться на сайт</a><br/>-->
 <form class="submission small style" action="<?php echo JRoute::_('index.php?option=com_users&task=user.logout'); ?>" method="post">
<a class='button_add'><button type="submit" class="exit_but">Выйти из личного кабинета</button></a>
<input type="hidden" name="return" value="<?php echo base64_encode($this->params->get('logout_redirect_url', $this->form->getValue('return'))); ?>" />
                <?php echo JHtml::_('form.token'); ?>
</form>
 </div>
<a href="#x" class="overlay" id="win1"></a>
    <div class="popup">
      <h2>Введите Ваш лицевой счёт и фамилию.</h2>
                        <form id="form_add" method="POST" action="#">

                        <p><strong>Лицевой счёт:</strong>
      <input class='input_add_check' type="text" id="civ_add" name="civ_add" size="10" placeholder="0123456789"/></p>
                        <p><strong>Фамилия:</strong>
                        <input class='input_add_check' type="text" id="sur_add" name="sur_add" size="18" placeholder="Иванов"/></p>
                        <p style="text-align: center;">
                                <a class='button' id="sub_add">Добавить</a>
                        </p>
                        </form>
                        <div id="response_add"></div>
                        <a class="close" title="Закрыть" href="#close"></a>
    </div>
 <?php
   echo "<div class='User_data'><p>Здравствуйте, $us_name</p><p>Ваш e-mail: $email</p></div><br/><br/><br/>";
        ?>
        <?php
        echo "<div id='main_cont'>";
if ($result = mysqli_query($link, $query)) {
   /* fetch associative array */
   while ($row = mysqli_fetch_row($result)) {
    $idNumber = $row[0];
    $secondName = $row[1];
    $stidCheckPass = oci_parse($conn, "select api_web_load.CheckPass(:idNumber, :secondName) from dual");
    oci_bind_by_name($stidCheckPass, ':idNumber', $idNumber);
    oci_bind_by_name($stidCheckPass, ':secondName', $secondName);
```

```
    oci_execute($stidCheckPass);
    while(oci_fetch($stidCheckPass)){
      if(oci_result($stidCheckPass, 1) == 0){
        ?><script>
          alert("Неверно указан лицевой счет или фамилия");
          document.location.href = '/index.php';
          </script>
       <?php
        exit;
      }
      else{
        $_SESSION['idNumber'] = $idNumber;
        $_SESSION['secondName'] = $secondName;
      }
    }
    $stidPrivs = oci_parse($conn, "select * from table(heat.api_web_load.getPrivs(:idNumber, :secondName))");
//show privs
    $stid = oci_parse($conn, "select * from table(heat.api_web_load.getCalcs(:idNumber, :secondName))");//show
calcs for year period
   // $stidcheck = oci_parse($conn, "select * from table(api_web_load.getCalcs(:m_civ_code, :m_pass,
:m_period))");//new calcs 15.03.16
     $stidcheck = oci_parse($conn, "select * from table(api_web_load.getCalcs(:m_civ_code, :m_pass,
999999))");//new calcs 15.03.16
  $stidCheckVod = oci_parse($conn, "select * from table(api_web_load.getmetrs(:m_civ_code, :m_pass))");
    // $testbdvod = oci_parse($conn, "select * from table(md_data.api_web_load.getmetrs(:m_civ_code, :m_pass))");
    //$stid = oci_parse($conn, "select * from table(api_web_load.getMetrs(:idNumber, :secondName))");
    if (!$stid || !$stidcheck || !$stidCheckVod) {
      $e = oci_error($conn);
      trigger_error(htmlentities($e['message'], ENT_QUOTES), E_USER_ERROR);
    }
    oci_bind_by_name($stid, ':idNumber', $_SESSION['idNumber']);
    oci_bind_by_name($stid, ':secondName', $_SESSION['secondName']);
     oci_bind_by_name($stidCheckVod, ':m_civ_code', $_SESSION['idNumber']);
    oci_bind_by_name($stidCheckVod, ':m_pass', $_SESSION['secondName']);
    oci_bind_by_name($stidcheck, ':m_civ_code', $_SESSION['idNumber']);
    oci_bind_by_name($stidcheck, ':m_pass', $_SESSION['secondName']);
    oci_bind_by_name($stidPrivs, ':idNumber', $_SESSION['idNumber']);
    oci_bind_by_name($stidPrivs, ':secondName', $_SESSION['secondName']);
    $r = oci_execute($stid);
    $y = oci_execute($stidCheckVod);
    $x = oci_execute($stidcheck);
    oci_execute($stidPrivs);
    echo "<br>";
    if (!$r) {
      $e = oci_error($stid);
      trigger_error(htmlentities($e['message'], ENT_QUOTES), E_USER_ERROR);
    }
   if (!$x) {
      $e = oci_error($stidcheck);
      trigger_error(htmlentities($e['message'], ENT_QUOTES), E_USER_ERROR);
    }
    if (!$y) {
```

```
    $e = oci_error($stidCheckVod);
    trigger_error(htmlentities($e['message'], ENT_QUOTES), E_USER_ERROR);
  }
  $CIV_CODE = $row[0];
  while ($row = oci_fetch_array($stid)) {
  $CIV_NAME = $row['CIV_NAME'];
  $ADDRESS = $row['ADDRESS'];
  $_SESSION['CIV_CODE'] = $row['CIV_CODE'];
  $_SESSION['CIV_NAME'] = $CIV_NAME;
  $_SESSION['ADDRESS'] = $ADDRESS;
  }
  oci_execute($stid);
  oci_execute($stidcheck);
    oci_execute($stidCheckVod);
  oci_execute($stidPrivs);
        echo "<div class='table_header'>Лицевой счёт - $CIV_CODE <br/>ФИО Владельца: $CIV_NAME
<br/>Адрес: $ADDRESS<br/></div>";
  print "<table class = 'table-border simple-little-table' id = 'fr'>";
  echo "<tr><td class='history_but td-none-bg'><a class='button' target='_blank'
href='/lichnii_cab/function_print.php?CIV_CODE=$CIV_CODE&id=$us_id' >Печать квитанции</a></td>
        <td class='print_but td-none-bg'><!--<a class='button' href = '#'
onclick='showPayments();return(false);'>История платежей</a>--></td>
                    <td class='delete_but td-none-bg' colspan = '3' align = 'center'><td class='delete_but td-
none-bg'><a class='button_del' href = '#' onclick='delete_acc($CIV_CODE, $us_id)'>Удалить
счёт</a></td></tr>";
  while($rowPriws = oci_fetch_array($stidPrivs)){
        echo "<tr><td colspan = '6' align = 'center'>Льготы</td></tr>";
    echo"<tr><th>Код</th><th>%</th><th>Чел</th><th colspan = '3'>Наименование льготы</th></tr>";
    echo"<tr><td align = 'center'>$rowPriws[PRIV_CODE]</td><td align =
'center'>$rowPriws[PRIV_PERC]</td><td align = 'center'>$rowPriws[PERS]</td><td colspan =
'3'>$rowPriws[PRIV_NAME]</td></tr>";
  }
  echo "<tr><td colspan = '6' align = 'center'>Начисления</td></tr>";
  echo "<tr ><th>Мес.год</th><th>КЧЛ</th><th>Площадь</th><th>К опла-
те</th><th>Оплата</th><th>Остаток</th></tr>";
  $SUM_TOPAY = 0;
  $SUM_PAY_BANK = 0;
  $currentDate = date("Ym");
  while ($row1 = oci_fetch_array($stid)) {
    $year = substr($row1['PERIOD_NAME'], -8);
    $month = substr($row1['PERIOD_NAME'], 0, -9);
    print "<tr>";
    echo "<td>$month<br>$year</td>";
    echo "<td align = 'center'>$row1[PERS]</td>";
    echo "<td align = 'center'>$row1[SQ]</td>";
    echo "<td align = 'right'>$row1[SUM_TOPAY]</td>";
    echo "<td align = 'right'>$row1[SUM_PAY_BANK]</td>";
    echo "<td align = 'right'>$row1[SALDOK]</td>";
    $SUM_TOPAY = $SUM_TOPAY + $row1['SUM_TOPAY'];
    $SUM_PAY_BANK = $SUM_PAY_BANK + $row1['SUM_PAY_BANK'];
    $SALDOC = $row1['SALDOK'];
  print "</tr>";
```

```
  }
    echo "<tr><th>Итого</th><td colspan = '4'><td align = 'right'>$SALDOC</td></tr>";
    echo "</table><br/><br/>";
   }
   /* free result set */
   mysqli_free_result($result);
}
/* close connection */
mysqli_close($link);
   }
   if(isset($stid))
                oci_free_statement($stid);
oci_close($conn);
?>
</div>
</body>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
function delete_acc(civ,id){
        if (confirm("Вы уверенны что хотите удалить счёт: " + civ + ", id: " + id))
        {
        $.ajax({
                type: 'POST',
                url: '/templates/shaper_oneclip/html/com_users/profile/del_acc.php',
                data: 'civ=' + civ + '&id=' + id,
                success: function(data){
                location.reload();
                }
                });
        }
}
var z=1;
$("#sub_add").click(function(){
var sur_add=$("#sur_add").val();
var civ_add=$("#civ_add").val();
var us_id=<?php echo $us_id?>;

        $.ajax({
                type: 'POST',
                url: '/lichnii_cab/function_add.php',
                data: 'sur_add=' + sur_add + '&civ_add=' + civ_add + '&us_id=' + us_id,
                success: function(data){
                $('#response_add').html(data);
                $('#response_add').append(z);
                z++;
                if($("#response_from_serv").data("hidden") === true)
                        {
                                location.reload();
                        }

                }
```

```
                    });

});
var div = $('#kepka');
      var start = $(div).offset().top;

      $.event.add(window, "scroll", function() {
         var p = $(window).scrollTop();
         $(div).css('position',((p)>start) ? 'fixed' : 'absolute');
         $(div).css('right','5px');
         $(div).css('top',((p)>start) ? '80px' : '');
                              $(div).css('opacity',((p)>start) ? '0.3' : '1');
                              $(div).css('z-index','99');
      });
</script>
<?php }?>
```