Національний університет «Львівська політехніка»

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Бази даних»**

на тему:

«База даних для обліку інтернет-провайдера»

Виконав: студент

спеціальності 121

«Інженерія програмного забезпечення»

групи ПЗ-34

Бовчалюк О.В.

Керівник:

доцент кафедри програмного забезпечення, к.т.н., доцент Павич Н.Я.

Оцінка:

Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів\_\_\_\_\_\_\_Оцінка ECTS\_\_\_\_

|  |  |  |
| --- | --- | --- |
| Члени комісії |  |  |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ | Павич Н.Я. |
|  | (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) | Квятковський Б.О. |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ | Цимбалюк Т.М. |
|  | (підпис) |  |

Львів – 2020 рік

**ЗМІСТ**

[РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ОБЛІКУ ІНТЕРНЕТ-ПРОВАЙДЕРА 3](#_Toc512156745)

[1.1. Опис предметної області 3](#_Toc512156746)

[1.2. Постановка завдання 4](#_Toc512156747)

[1.3. Специфікація вимог до системи обліку інтернет-провайдера 6](#_Toc512156748)

[1.3.1. Вступ 6](#_Toc512156749)

[1.3.2. Загальний опис 7](#_Toc512156750)

[1.3.3. Характеристики системи 9](#_Toc512156751)

[1.3.4. Вимоги зовнішніх інтерфейсів 13](#_Toc512156752)

[1.3.5. Інші нефункціональні вимоги 4](#_Toc512156753)

РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ 14

2.1. Концептуальне моделювання предметної області 14

2.1.1. Опис сутностей 14

2.1.2. Опис зв’язків між сутностями 20

2.2. Логічне проектування БД 22

2.3. Проектування типових запитів і транзакцій 25

[РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ РОБОТИ З БАЗОЮ ДАНИХ 2](#_Toc512156760)6

[3.1. Реалізація доступу до бази даних 2](#_Toc512156761)6

[3.2. Реалізація функціональних характеристик системи](#_Toc512156762) 27

[3.3. Опис роботи програми](#_Toc512156768) 40

[ВИСНОВКИ 4](#_Toc512156769)8

[СПИСОК ЛІТЕРАТУРИ 4](#_Toc512156770)9

[Додаток А. Скрипт створення БД 50](#_Toc512156771)

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ОБЛІКУ РОБОТИ ІНТЕРНЕТ-ПРОВАЙДЕРА

* 1. Опис предметної області

Існує безліч підприємств, що надають провайдерські послуги. У кожного з них можуть бути філії в різних регіонах .

Доступ до інтернету надається за різними технологіями, яким відповідають різні тарифи. Крім доступу в інтернет, провайдери можуть надавати додаткові послуги. Наприклад безкоштовний роутер для користування інтернетом.

База даних займає одне з чільних місць в системі обліку і контролю інтернет-провайдера і вимагає особливої ​​уваги до своєї працездатності.

Таким чином, зростає складність та обсяг інформації, яку потрібно зберігати та аналізувати для забезпечення оперативного контролю.

У всіх інтернет-провайдерів існує кілька загальних проблем, до числа яких належать: облік клієнтів, які користуються послугами, облік працівників і ведення бухгалтерії. Для вирішення цих проблем адміністрація змушена утримувати значний штат бухгалтерів, обліковців, менеджерів різних рівнів і т. д. Однак помилки в обліку не виключаються, а можливості аналізу залишаються обмеженими.

Ефективний засіб для вдосконалення роботи – застосування інформаційних технологій. Для того, щоб оперативно працювати з клієнтами та з персоналом та відповідали динаміці сучасного бізнесу.

Підписання контракту з клієнтом – це початкова стадія процесу надання послуги. Він включає визначення складу послуги(тарифний план, період використання, надання додаткових послуг).

Наступна стадія процесу надання послуг – підключення інтернету для клієнтів.

Інформаційна система повинна бути розроблена на основі сучасних інформаційних технологій, що означає: використання системи управління базами даних, яка забезпечує надійну роботу з великими обсягами даних, високу швидкість доступу до даних, безпеку їх зберігання; можливість роботи в локальній обчислювальній мережі з необмеженою кількістю робочих місць, що забезпечує формування бази даних в режимі реального часу; повну інтеграцію даних, які вводяться в систему, що дозволяє здійснювати всебічний аналіз діяльності інтернет-провайдера.

Таким чином, для створення інформаційної системи станції технічного обслуговування, ми повинні забезпечити внесення таких даних:

* дані про співробітників;
* дані про послуги, які надає інтернет-провайдер;
* дані про клієнтів (як фізичних так і юридичних);
* дані про оплату послуг;
* дані про підписані договори;
  1. Постановка завдання

Для фірм, які займаються наданням послуг інтернет-провайдерів товарів велике значення має формування бази даних, оскільки обсяги інформації є великими і потребують достатніх ресурсів для їх зберігання.

База даних міститиме усю інформацію про договори, клієнтів, працівників.

У базі даних необхідно зберігати інформацію про договір, який був підписаний між клієнтом та фірмою. Відповідно інформація повинна складатися з:

* клієнта, який безпосередньо бере участь у замовленні послуг;
* працівника, який оформив замовлення;
* послугу;
* дати підписання договору;

Доступ до бази даних повинен здійснюватися за допомогою форми авторизації. Авторизація має обмежувати доступ до властивостей додатку.

Користувачі додатку(працівники) мають мати можливість формувати звіти, які показують поточну ситуацію компанії.

Програма повинна попереджати користувачів у випадку виникнення помилок чи збоїв, бути стійкою та розрахованою на постійне внесення змін.

Сценарії взаємодії між користувачами системи та випадками використання, що описують функціональні аспекти системи, зображено на діаграмі прецедентів (рис.1.1) [1].

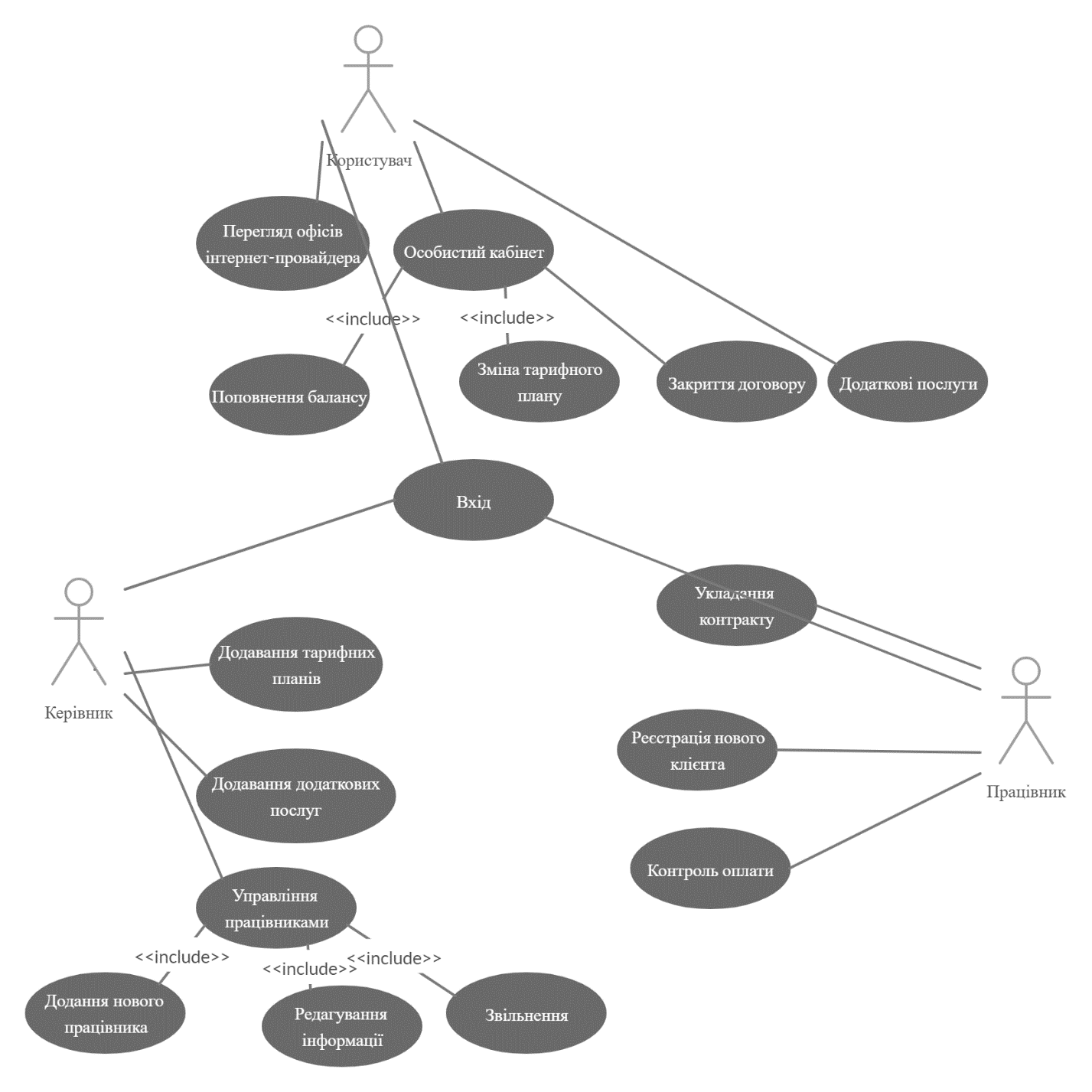


Рис.1.1. Діаграма прецедентів

* 1. Специфікація вимог до системи обліку інтернет-провайдера
     1. Вступ
        1. Призначення, мета

Даний програмний продукт призначений для обліку інтернет-провайдера з метою автоматизації обліку наданих послуг. Система забезпечує оперативний контроль, дозволяє зберігати інформацію про клієнтів, надані послуги та працівників.

* + 1. Загальний опис
       1. Перспективи продукту

Програмний продукт дозволяє розв’язати проблему організації роботи інтернет-провайдера. Його функціональні можливості полягають у забезпеченні процесу автоматизації надання послуг. Система повинна бути простою в користуванні із інтерфейсом, який зрозумілий навіть користувачу-початківцю.

* + - 1. Характеристики продукту

Продукт повинен виконувати наступні функції:

* додавання тарифних планів;
* додавання працівників;
* укладання контракту;
* закриття контракту;
* надання додаткових послуг;
* оплата послуг;
* додавання клієнтів;
  + - 1. Класи користувачів та їх характеристики

В даному програмному продукті будуть наявні три класи користувачів:

1. Працівник – даний клас користувачів матиме змогу керувати клієнтами та укладати договори.
2. Керівник – даний клас користувачів матиме змогу адмініструвати систему в цілому, перевіряти коректність виконання її функціональної частини, керувати працівниками, добавляти нові послуги.
3. Користувач – даний клас користувачів матиме змогу відслідковувати закріплені за ним послуги, проводити оплату, керувати власною інформацією.
   * + 1. Середовище функціонування

Даний програмний продукт буде функціонувати в операційних системах Windows (Windows 7 – Windows 10).

Таблиця 1.1. Вимоги до платформи

|  |  |
| --- | --- |
| Процесор | x64 cумісний процесор з частотою 1.4 Ghz (рекомендовано 2 Ghz і вище) |
| Пам’ять | 1 Gb (рекомендовано не менше 2 Gb із можливістю подальшого збільшення об’єму відповідно до збільшення бази даних. |
| Операційна система | ОС Windows (Windows 7 – Windows 10). |
| Додаткові програмні компоненти | Microsoft .NET Framework 3.5, а також Microsoft SQL Server 2012 і вище для організації бази даних, а також роботи з системою керування базою |
| Інтернет | Наявність доступу до мережі інтернет є бажаною для своєчасного оновлення компонентів системи. |

* + - 1. Обмеження проектування та реалізації

Продукт повинен бути реалізований впродовж 6 місяців після старту розробки. Система не повинна вимагати наявності додаткових програмних компонентів крім Microsoft .NET Framework, а також Microsoft SQL Server.

* + - 1. Документація користувача

Для працівників буде доступна наступна документація:

* інструкції використання – покрокове встановлення програмного продукту та його налаштування;
* документація про використання, в якій буде представлено приклади використання основних функцій даного програмного продукту.
  + - 1. Припущення та залежності

Для коректної роботи продукту потрібно мати встановленим екземпляр SQL Server версії 2008 і вище, а також потрібно встановити Microsoft .NET Framework 4.5 або новішу версію.

* + 1. Характеристики системи
       1. Авторизація користувача
          1. Опис і пріоритет

Авторизація користувача відповідно до його типу.

Пріоритет – високий.

* + - * 1. Послідовності дія/відгук

Під час авторизації користувач повинен заповнити форму, де потрібно вказати логін та пароль. Якщо дані внесено правильно – повідомлення про успіх, в іншому випадку користувач отримає повідомлення з описом помилки.

**Користувач**

**Форма для авторизації**

**База даних**

* + - * 1. Функціональні вимоги

REQ-1: Перевірка коректності введених даних в полях авторизації.

REQ-2: У разі виникнення помилки при введенні даних, показується повідомлення з помилкою.

REQ-3: Приховування даних входу.

* + - 1. Укладання контракту
         1. Опис і пріоритет

Укладання контракту виконується наступним чином: працівник працює з клієнтом (вибирають послуги які хоче отримати клієнт).

Пріоритет – високий.

* + - * 1. Послідовності дія/відгук

Під час укладання договору заповнюється форма, в якій потрібно вказати інформацію про працівника та клієнта які укладають договір та послуги, які мають бути підключені. Якщо дані внесено правильно – повідомлення про успіх, у протилежному випадку – повідомлення з описом помилки.

**Працівник**

**Форма для укладання контракту**

**База даних**

* + - * 1. Функціональні вимоги

REQ-1: Перевірка коректності введених даних в полях.

REQ-2: У разі виникнення помилки при введені даних, показується повідомлення з помилкою, де вказується, в яких полях введено невірні дані.

* + - 1. Керування послугами
         1. Опис і пріоритет

Можливість підключення нових послуг або редагувати вже наявні.

Пріоритет – високий.

* + - * 1. Послідовності дія/відгук

Під час редагування послуг змінюються відповідні дані. Для цього у відповідній формі вносяться зміни.

**Користувач**

**Форма для редагування послуг**

**База даних**

* + - * 1. Функціональні вимоги

REQ-1: Редагування даних про послуги. Передбачити неправильне введення даних у наведені вище поля.

REQ-2: У разі виникнення помилки при введені даних, показується повідомлення з помилкою, де вказується, в яких полях введено невірні дані.

REQ-3: Додавання додаткової нової послуги. Передбачити неправильне введення даних у наведені вище поля.

REQ-5: Скасування/видалення послуги. Передбачити запит на видалення із повторним підтвердженням операції.

* + - 1. Додавання нового клієнта, працівника
         1. Опис і пріоритет

Створюється новий клієнт який хоче укласти контракт, або додається новий працівник.

Пріоритет – середній.

* + - * 1. Послідовності дія/відгук

Під час додавання нового клієнта(працівника) вносяться відповідні його дані в спеціальній формі.

**Працівник(керівник)**

**Форма для додавання  
клієнта(працівника)**

**База даних**

* + - * 1. Функціональні вимоги

REQ-1: Редагування даних про послуги. Передбачити неправильне введення даних у наведені вище поля.

REQ-2: У разі виникнення помилки при введені даних, показується повідомлення з помилкою, де вказується, в яких полях введено невірні дані.

* + 1. Вимоги зовнішніх інтерфейсів
       1. Користувацькі інтерфейси

Інтерфейс повинен включати вікно входу в систему. В залежності від класу користувача буде представлений різний графічний інтерфейс.

* + - 1. Апаратні інтерфейси

Комп'ютер на якому буде виконуватись дана програма повинен мати наступні апаратні характеристики:

* 32-х бітний або 64-х бітний процесор з тактовою частотою 1 ГГц, або більше;
* 1 ГБ ОЗП (х86) або ж 2 ГБ ОЗП (х64);
* Графічний пристрій з підтримкою DirectX9.
  + - 1. Програмні інтерфейси

Для коректної роботи програми потрібні наступні програмні компоненти:

* ОС Windows (Windows 7 – Windows 10);
* .Net framework 3.5 і вище.
  + - 1. Комунікаційні інтерфейси

Для коректної роботи ПК повинен мати порт RJ-45 або ж модуль Wi-Fi, а локальна мережа підтримувати стандарт Fast Ethernet.

* + 1. Інші нефункціональні вимоги
       1. Вимоги продуктивності

Основними вимогами до продуктивності системи є:

* керування змінами даних в програмі, швидка заміна застарілих даних;
* при заміні даних синхронізація з користувачем має бути оперативною;
* коректне та швидке використання паралельного використання сервера кількома користувачами .
  + - 1. Вимоги надійності

Необхідно забезпечити надійність зберігання даних у базі даних та захист їх від негативних сторонніх впливів, втрат та пошкоджень, оскільки втрата цих даних може спричинити значні збитки. Крім того, повинен бути реалізований захист від випадкових неправильних дій користувачів, зберігання повної історії змін для можливості повернення системи до одного з її минулих станів у будь–який момент часу.

* + - 1. Вимоги безпеки

Необхідне нерозголошення інформації про розробку, підтримку та використання продукту, а також про дані, які будуть занесені у базу даних.

Вимоги аутентифікації користувачів:

• пароль із високим рівнем складності (довжина: не менше 8 знаків,

набір різних символів, таких як літери та цифри);

* + - 1. Атрибути якості програмного продукту
* Коректність – програма повинна бути написана згідно із вимоги замовника та правильно обробляти дані;
* повнота – програмний продукт повинен містити всю необхідну інформацію при оформленні контракту;
* верифікованість – у програмі реалізовані тільки ті вимоги, які потрібні замовнику;
* несуперечливість – поля вводу даних зрозуміло описані, щоб не виникало двозначності;
* супроводжуваність – компанія–розробник повинна протягом зазначеного часу надавати технічну допомогу при супроводі, а також модернізовувати існуючий продукт;
* надійність – програма не «вилітає» при некоректному вводі даних;
* стійкість – програма правильно обробляє різні види помилок;
* зручність використання – інтерфейс програма є інтуїтивно зрозумілим для користувачів програми.

РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ

* 1. Концептуальне моделювання предметної області
     1. Опис сутностей

У БД було створено наступні сутності: «Контракт», «Тарифний план», «Закриття контракту», «Оплата», «Працівники», «Посади», «Заробітна плата», «Працівники», «Фізичні», «Юридичні», «Додаткові сервіси», «Сервіси».

Сутність *Контракт*



Рис.2.1. Таблиця «Контракт»

Дана сутність описує всі необхідні дані, які потрібні для підписання контракту. Таблиця містить усю необхідну інформацію про контракт в тому числі дату підписання, працівник який укладав договір, тарифний план, код контракту. Має зв’язки з такими сутностями: Тарифний план, Закриття договору, Працівники, Клієнти. Містить такі поля:

* ID\_Contract – унікальний номер (ідентифікатор), призначений кожному договору;
* ID\_Customer – унікальний номер (ідентифікатор), призначений кожному клієнту;
* ID\_Employee – унікальний номер (ідентифікатор), призначений кожному працівнику;
* ID\_Tariff\_Plan – унікальний номер (ідентифікатор), призначений для певного тарифного плану;
* Date\_Of\_Signing – дата підписання договору;

Сутність *Тарифний план*

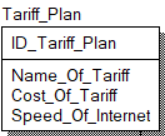


Рис.2.2. Таблиця «Тарифний план»

Дана сутність відображає всі можливі тарифні плани, які доступні. Таблиця містить усю необхідну інформацію про тарифний план в тому числі назву, вартість, код тарифного плану, шкидвість інтернету. Має зв’язки з такими сутностями: Контракт. Містить такі поля:

* ID\_Tariff\_Plan – унікальний номер (ідентифікатор), призначений кожному тарифному плану;
* Name\_Of\_Tariff – назва тарифного плану;
* Cost\_Of\_Tariff – вартість тарифного плану;
* Speed\_Of\_Internet – швидкість тарифного плану;

Сутність *Закриття контракту*

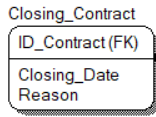


Рис.2.3 Таблиця «Закриття контракту»

Дана сутність відображає інформацію про закриття договору між інтернет провайдером та клієнтом. Таблиця містить таку інформацію як: код договору, дата закриття, причина. Має зв’язки з такими сутностями: Договір. Містить такі поля:

* ID\_Contract – унікальний номер (ідентифікатор), призначений договору;
* Closing\_Date – дата коли договір був розірваний;
* Reason – причина розірвання договору;

Сутність *Оплата*

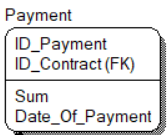


Рис.2.4. Таблиця «Оплата»

Дана сутність описує дату оплати за тариф та суму. Має зв’язки з такими сутностями: Контракт. Містить такі поля:

* ID\_Payment – унікальний номер (ідентифікатор), призначений оплаті тарифного плану;
* ID\_Contract – унікальний номер (ідентифікатор), призначений договору;
* Sum – скільки грошей було заплачено;
* Date\_Of\_Payment – дата здійснення оплати;

Сутність *Клієнти*

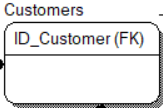


Рис.2.5. Таблиця «Клієнти»

Дана сутність відображає можливих клієнтів. Таблиця містить таку інформацію: код клієнта. Служить для розділення клієнтів на фізичних та юридичних. Має зв’язки з такими сутностями: Договір, Фізичні, Юридичні. Містить такі поля:

* ID\_Customet – унікальний номер (ідентифікатор), призначений кожному клієнту;

Сутність *Фізичні*

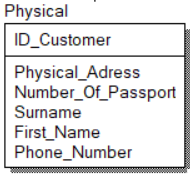


Рис.2.6. Таблиця «Фізичні»

Дана сутність відображає інформацію про фізичних осіб. Таблиця містить таку інформацію: код фізичної особи, ПІБ, адресу, номер паспорта, номер телефону. Має зв’язки з такими сутностями: Клієнти. Містить такі поля:

* ID\_Customer – унікальний номер (ідентифікатор), призначений клієнту;
* First\_Name – ім'я;
* Surname – прізвище;
* Phone\_Number – номер телефону;
* Physical\_Adress – адреса проживання;
* Number\_Of\_Passport – номер паспорта;

Сутність *Юридичні*

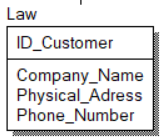


Рис.2.7. Таблиця «Юридичні»

Таблиця “Юридичні” відображає інформацію про юридичні фірми. Таблиця містить таку інформацію: код юридичної фірми, назва фірми, адреса, номер телефону. Має зв’язки з такими сутностями: Клієнти. Містить такі поля:

* ID\_Customer – унікальний номер (ідентифікатор), призначений кожному клієнту;
* Company\_Name – назва фірми;
* Phone\_Number – номер телефону;
* Physical\_Adress – адреса;

Сутність *Посади*

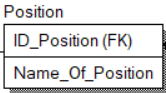


Рис.2.8. Таблиця «Посади»

Дана сутність відображає посади, які можуть мати працівники. Таблиця містить таку інформацію: код посади, назву посади. Має зв’язки з такими сутностями: Працівники, Заробітна плата. Містить такі поля:

* ID\_Position – унікальний номер (ідентифікатор), призначений кожній посаді;
* Name\_Of\_Position – назва посади, яку може мати працівник;

Сутність *Заробітна плата*

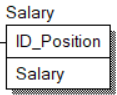


Рис.2.9. Таблиця «Заробітна плата»

Дана сутність відображає заробітну плату для працівників. Таблиця містить таку інформацію як: код посади та заробітну плату. Має зв’язки з такими сутностями: Посади. Містить такі поля:

* ID\_Position – унікальний номер (ідентифікатор), призначений кожній посаді;
* Salary – заробітна плата працівників;

Сутність *Додаткові сервіси*



Рис.2.10. Таблиця «Додаткові сервіси»

Дана сутність відображає додаткові сервіси до контракту. Таблиця містить таку інформацію як: код сервісу, код контракту, дата початку, дата закінчення. Має зв’язки з такими сутностями: Контракт, Сервіси. Містить такі поля:

* ID\_Services – унікальний номер (ідентифікатор), призначений додатковим сервісам;
* ID\_Contract – унікальний номер (ідентифікатор), призначений договору;
* Start\_Date – дата початку;
* End\_Date – дата закінчення;

Сутність *Сервіси*

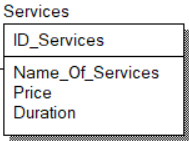
-

Рис.2.11. Таблиця «Сервіси»

Дана сутність відображає інформацію про додаткові сервіси. Таблиця містить таку інформацію як: код сервісу, назва додаткового сервісу, вартість та тривалість. Має зв’язки з такими сутностями: Додаткові сервіси. Містить такі поля:

* ID\_Services – унікальний номер (ідентифікатор), призначений додатковим сервісам;
* Name\_Of\_Services – назва додаткового сервісу;
* Price – вартість;
* Duration – тривалість;
  + 1. Опис зв’язків між сутностями

Між сутностями було встановлено наступні зв’язки:

* Між сутностями Договір та Тарифний план існує зв’язок “один до багатьох”, оскільки кожен договір прив’язаний до певного тарифного плану, який може повторюватись. Також ця сутність пов’язана з сутністю Працівники зв’язком “один до багатьох”, оскільки кожен договір має працівника який складав цей договір, і цей працівник може пересікатись в інших договорах. Також з сутністю Клієнти існує зв'язок “один до багатьох”, оскільки кожен клієнт може підписувати не один договір про надання послуг, а декілька, наприклад для різних будівель. З сутністю Закриття договору існує зв'язок “один до одного”, оскільки один договір може бути лише один раз розірваний.
* Сутність Тарифний план пов’язана зв’язком “один до багатьох” з сутністю Оплата, оскільки кожен тарифний план має оплату, і оплата за тарифні плани може збігатися.
* Між сутностями Контракт та Додаткові сервіси існує зв’язок “один до багатьох”, оскільки додаткових сервісів може бути безліч.
* Сутність Працівники пов’язана з сутність Посади зв’язком “один до багатьох”, оскільки одну й ту саму посаду можуть мати декілька працівників.
* Сутність Посади пов’язана з сутністю Зарплата зв’язком “один до багатьох”, оскільки одну й ту саму зарплату можуть мати різні працівники, які займають різні посади.
* Сутність Клієнти пов’язана з сутністю Фізичні зв’язком “один до багатьох”, оскільки фізичні клієнти характеризуються як клієнти в цілому і їх може бути багато. Також ця сутність пов’язана з сутністю Юридичні зв’язком “один до багатьох”, оскільки юридичні клієнти характеризуються як клієнти в цілому і їх може бути багато.
  1. Логічне проектування БД

В рамках даної роботи було створено логічну (рис.2.12) та фізичну (рис.2.13) моделі БД для обліку роботи інтернет-провайдера.

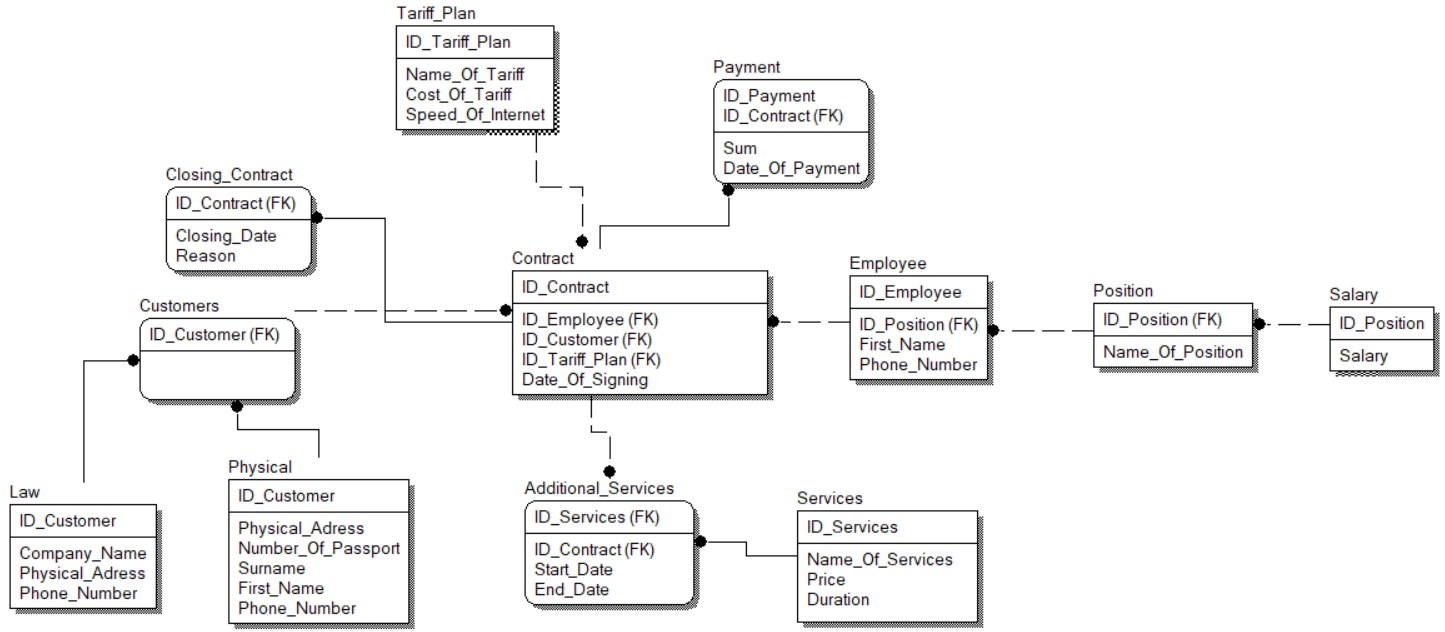


Рис. 2.12. Логічна модель

Після створення логічної моделі потрібно перейти на фізичний рівень. В даній БД обидві моделі вийшлі однакові.

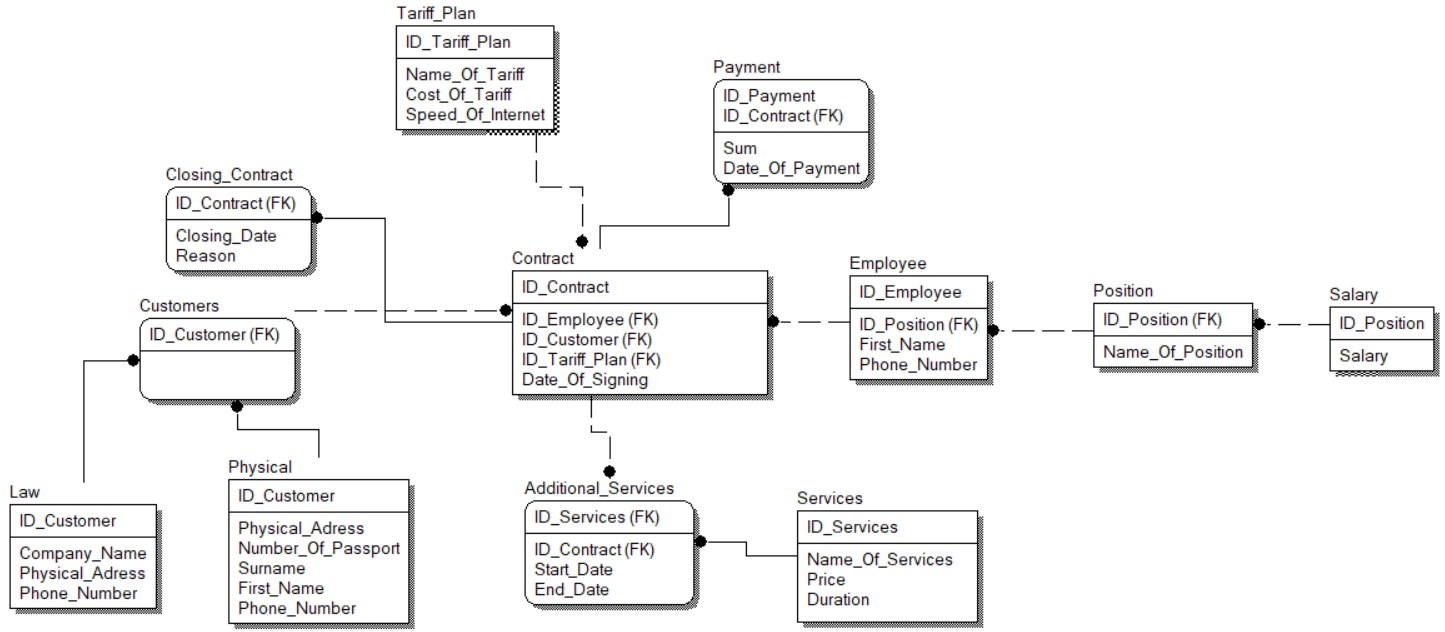


Рис.2.13. Фізична модель

Наступним кроком є генерація скрипта бази даних. Для цього потрібно переключитися на фізичну модель і на панелі інстументів вибрати "Tools | Forward Engineering | Schema Generation". Натиснувши кнопку "Preview", ми отримаємо код скрипта.

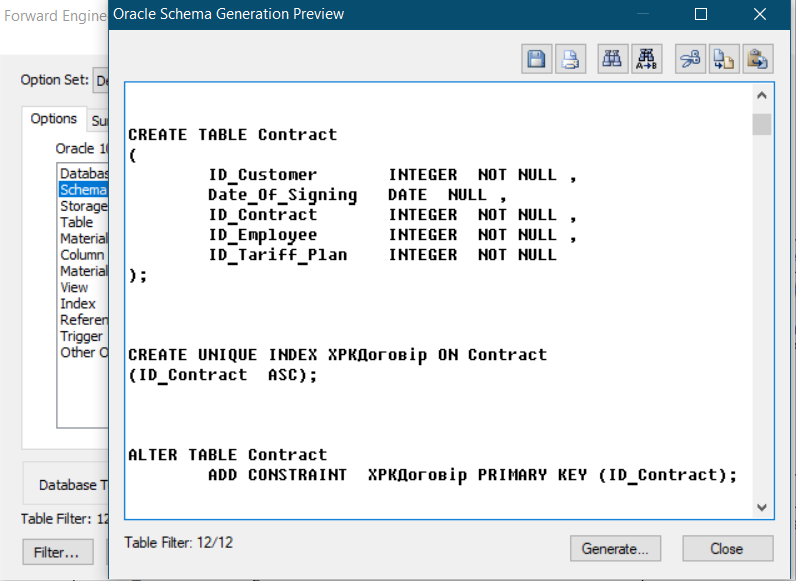


Рис. 2.14. Вікно відображення скрипта створення бази даних

Після генерації скрипта я переніс його у середовище SQL Server Management Studio та з його допомогою створив базу даних.

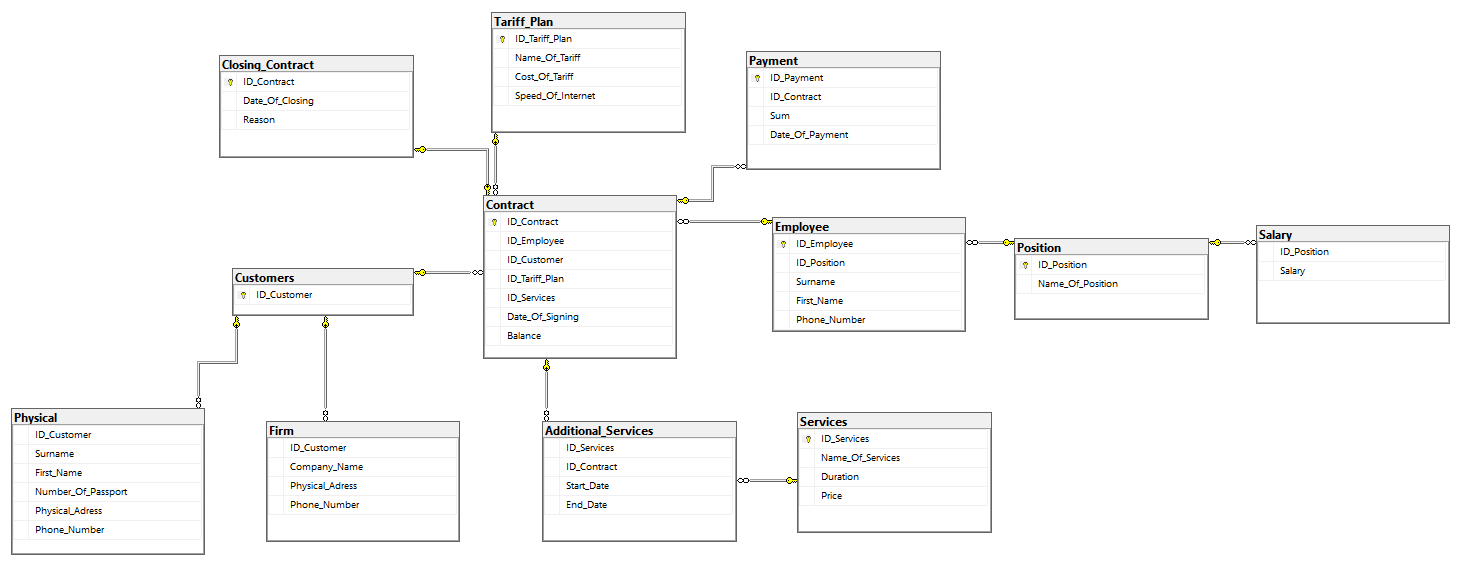


Рис. 2.15. Діаграма бази даних в SQL Server

Для відображення інформації були створені такі віртуальні таблиці:

* Віртуальна таблиця для перегляду інформації про замовлення

Були додані наступні обмеження:

* Усі первинні ключі не можуть бути рівні NULL (обмеження SQL) та менші 0.
* На усі атрибути з вторинними ключами було накладено обмеження з умовою - не рівні NULL.
  1. Проектування типових запитів і транзакцій

В ході проектування системи управління базою даних було, спроектовано ряд запитів, а саме:

* Запит, який виводить інформацію для формування контракту відповідно до замовлення через віртуальну таблицю, формуючи її з таблиць «Customers», «Employee», «Tariff\_Plan», «Additional\_Services.
* Запит, який додає записи в таблицю «Customer».
* Запит, який додає записи в таблицю «Employee».
* Запит, який додає записи в таблицю «Contract», який може включати в себе підзапити з додавання нового клієнта в відповідних таблицях «Physical» і «Firm», якщо не були занесені в базу раніше.
* Запити, що дозволяють видаляти записи з таблиць «Contract», «Customers», «Firm», «Physical», «Employee», «Additional\_Services», «Tariff\_Plan», «Closing\_Contract»
* Запит, який видаляє працівника шляхом видалення його з таблиці «Employee».
* Запит, який виводить інформацію про замовлення конкретного клієнта відповідно до номера клієнта, яке зберігається в таблиці «Customers».
* Вибірка всіх працівників з таблиці «Employee» разом з посадами, що містяться в таблиці «Position», шляхом побудови багатотабличного запиту.
* Запити, що редагують. записи в таблицях: «Contract», «Customers», «Firm», «Physical», «Employee», «Additional\_Services», «Tariff\_Plan», «Closing\_Contract».

Створено такий тригер:

* Тригер, який спрацьовує при додаванні додаткової послуги до контракту (перераховує загальну вартість).

Було створено такі збережені процедури:

* Збережена процедура, яка відповідно до вказаної посади видає список працівників.
* Збережена процедура, яка дозволяє додати нову послугу до існуючих.

Індекси в даній системі будуть використовуватися виключно за замовчуванням середовища SQL Server, через відсутність великих потоків даних для зберігання.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ РОБОТИ З БАЗОЮ ДАНИХ

**3.1.Реалізація доступу до бази даних**

Зв'язок з базою даних у Microsoft SQL Server був реалізований завдяки додатку на платформі .NET Framework 4.6 з використанням мови C# і технології ASP.NET MVC. Для розробки додатку використовувався архітектурний шаблон MVC. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Для взаємодії з базою даних використоється технологія Entity Framework. Вона має низку переваг на фоні інших доступних засобів. Зокрема, автоматична генерація класів, що відповідають таблицям у базі данних, простота роботи, захищеність, транзактивність тощо.

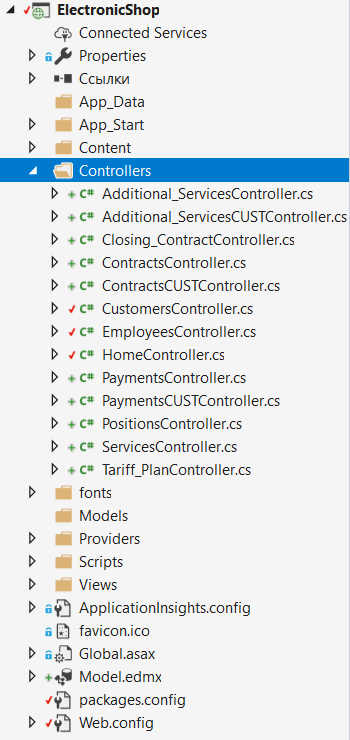


Рис. 3.1. Структура проекту

3.2.Реалізація функціональних характеристик системи

**3.2.1.Реєстрація користувача**

Реєстрація користувача реалізована наступним чином:

Для реєстрації нового користувача в систему має ввійти працівник, для цього він проходить механізм авторизації, для якої потрібно вказати логін та пароль. Якщо авторизація пройшла успішно, працівник потрапляє на головне вікно програми. Він переходить у вкладку «Працівники, натискає відповідну кнопку, та вносить дані про нового працівника у новій формі, що з’явиться. Після заповнення полів перевіряється чи вільний даний логін. Якщо дані відповідають всім умовам – користувача зареєстровано. Фрагмент програми:

private InternetProviderEntities db = new InternetProviderEntities();

// GET: Employees

public ActionResult Index()

{

var employees = db.Employee.Include(c => c.Position);

return View(employees.ToList());

}

// GET: Employees/Create

public ActionResult Create()

{

ViewBag.RoleID = new SelectList(db.Role, "RoleID", "RoleName");

ViewBag.ID\_Position = new SelectList(db.Position, "ID\_Position", "Name\_Of\_Position");

return View();

}

// POST: Employees/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "ID\_Employee,ID\_Position,Surname,First\_Name,Phone\_Number,Password,RoleID")] Employee employee)

{

if (ModelState.IsValid)

{

db.Employee.Add(employee);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(employee);

}

3.2.2.Авторизація користувача

Авторизація відбувається наступним чином:

Під час входу в систему користувач повинен ввести свій логін, який був вибраний при реєстрації, та пароль. Програма підтримує 2 типи користувачів. Тип користувача залежить чи це працівник чи клієнт. В залежності від введених даних, користувач авторизується, та одразу потрапляє на відповідне вікно. Якщо ж користувача з такими даними немає, тоді програма видає повідомлення про помилку.

Фрагмент програми:

public ActionResult Login(Employee user, string returnUrl)

{

var db = new InternetProviderEntities();

var dataItem = db.Customers.FirstOrDefault(x => x.Phone\_Number == user.Phone\_Number

&& x.Password == user.Password);

var dataItem2 = db.Employee.FirstOrDefault(x => x.Phone\_Number == user.Phone\_Number

&& x.Password == user.Password);

if (dataItem != null)

{

FormsAuthentication.SetAuthCookie((dataItem.ID\_Customer).ToString(), false);

string name = User.Identity.Name;

return RedirectToAction("Index", "Home");

}

if(dataItem2 != null)

{

FormsAuthentication.SetAuthCookie((dataItem2.ID\_Employee).ToString(), false);

string name = User.Identity.Name;

return RedirectToAction("Index", "Home");

}

ModelState.AddModelError("", "Некоректні дані. Спробуйте ще раз.");

return View();

}

3.2.3.Відкриття контракту

Для надання послуг клієнту, в програму повинен увійти користувач типу «Працівник». Після авторизації він повинен зайти на вкладку «Контракти», в таблиці обрати потрібного користувача, та обрати з випадаючого списку необхідний тарифний план, після чого натиснути «ОК».

Фрагмент програми:

// GET: Contracts

public ActionResult Index()

{

var contract = db.Contract.Include(c => c.Customers).Include(c => c.Employee).Include(c => c.Tariff\_Plan);

return View(contract.ToList());

}

// GET: Contracts/Create

public ActionResult Create()

{

var idEmp = Convert.ToInt32(HttpContext.User.Identity.Name);

ViewBag.ID\_Customer = new SelectList(db.Customers, "ID\_Customer", "ID\_Customer");

//ViewBag.ID\_Employee = new SelectList(db.Employee, "ID\_Employee", "Surname");

ViewBag.ID\_Employee = new SelectList(db.Employee.Where(c => c.ID\_Employee == idEmp), "ID\_Employee", "Surname");

ViewBag.ID\_Tariff\_Plan = new SelectList(db.Tariff\_Plan, "ID\_Tariff\_Plan", "Name\_Of\_Tariff");

return View();

}

// POST: Contracts/Create

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "ID\_Contract,ID\_Employee,ID\_Customer,ID\_Tariff\_Plan,Date\_Of\_Signing")] Contract contract)

{

if (ModelState.IsValid)

{

db.Contract.Add(contract);

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.ID\_Customer = new SelectList(db.Customers, "ID\_Customer", "Surname", contract.ID\_Customer);

ViewBag.ID\_Employee = new SelectList(db.Employee, "ID\_Employee", "Surname", contract.ID\_Employee);

ViewBag.ID\_Tariff\_Plan = new SelectList(db.Tariff\_Plan, "ID\_Tariff\_Plan", "Name\_Of\_Tariff", contract.ID\_Tariff\_Plan);

return View(contract);

}

<div class="form-horizontal">

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.Label("Працівник", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("ID\_Employee", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.ID\_Employee, "", new { @class = "text-danger" })

</div>

</div>

</div>

<div class="form-group">

@Html.Label("Номер клієнта", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("ID\_Customer", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.ID\_Customer, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.Label("Тарифний план", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("ID\_Tariff\_Plan", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.ID\_Tariff\_Plan, "", new { @class = "text-danger" })

</div>

</div>

3.2.4.Робота з даними про працівників

Для роботи з даними про працівників у систему потрібно увійти користувачу типу «Адміністратор». Після успішного проходження авторизації, він повинен перейти у вкладку «Працівники», та обрати дані про працівника якого потрібно переглянути. Після цього йому потрібно обрати потрібну дію з наданих, серед яких є: додавання нового працівника, редагування та видалення.

Фрагмент коду:

// GET: Employees

public ActionResult Index()

{

var employees = db.Employee.Include(c => c.Position);

return View(employees.ToList());

}

// GET: Employees/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = db.Employee.Find(id);

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

// GET: Employees/Create

public ActionResult Create()

{

ViewBag.RoleID = new SelectList(db.Role, "RoleID", "RoleName");

ViewBag.ID\_Position = new SelectList(db.Position, "ID\_Position", "Name\_Of\_Position");

return View();

}

// POST: Employees/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "ID\_Employee,ID\_Position,Surname,First\_Name,Phone\_Number,Password,RoleID")] Employee employee)

{

if (ModelState.IsValid)

{

db.Employee.Add(employee);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(employee);

}

// GET: Employees/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = db.Employee.Find(id);

if (employee == null)

{

return HttpNotFound();

}

//ViewBag.ContractId = new SelectList(db.Contracts, "ContractId", "ContractId", employee.ContractId);

//ViewBag.RoleId = new SelectList(db.EmployeeRoles, "RoleId", "RoleName", employee.RoleId);

return View(employee);

}

// POST: Employees/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Surname,Name,EnterDate,PhoneNumber,Password,EmployeeId,RoleId")] Employee employee)

{

if (ModelState.IsValid)

{

db.Entry(employee).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

// ViewBag.ContractId = new SelectList(db.Contracts, "ContractId", "ContractId", employee.ContractId);

// ViewBag.RoleId = new SelectList(db.EmployeeRoles, "RoleId", "RoleName", employee.RoleId);

return View(employee);

}

// GET: Employees/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = db.Employee.Find(id);

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

// POST: Employees/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Employee employee = db.Employee.Find(id);

db.Employee.Remove(employee);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

3.2.5.Додавання та видалення нових тарифних планів та додаткових послуг

Для додавання нових послуг та тарифів у систему потрібно увійти користувачу типу «Працівник». Після цього він матиме змогу перейти на вкладку з тарифами та додатковими послугами відповідно. Для видалення потрібного тарифу та додаткової послуги потрібно обрати тариф у таблиці та натиснути відповідну кнопку.

Фрагмент коду:

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "ID\_Tariff\_Plan,Name\_Of\_Tariff,Cost\_Of\_Tariff,Speed\_Of\_Internet")] Tariff\_Plan tariff\_Plan)

{

if (ModelState.IsValid)

{

db.Tariff\_Plan.Add(tariff\_Plan);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(tariff\_Plan);

}

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Tariff\_Plan tariff\_Plan = db.Tariff\_Plan.Find(id);

if (tariff\_Plan == null)

{

return HttpNotFound();

}

return View(tariff\_Plan);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "ID\_Tariff\_Plan,Name\_Of\_Tariff,Cost\_Of\_Tariff,Speed\_Of\_Internet")] Tariff\_Plan tariff\_Plan)

{

if (ModelState.IsValid)

{

db.Entry(tariff\_Plan).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(tariff\_Plan);

}

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Tariff\_Plan tariff\_Plan = db.Tariff\_Plan.Find(id);

if (tariff\_Plan == null)

{

return HttpNotFound();

}

return View(tariff\_Plan);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Tariff\_Plan tariff\_Plan = db.Tariff\_Plan.Find(id);

db.Tariff\_Plan.Remove(tariff\_Plan);

db.SaveChanges();

return RedirectToAction("Index");

}

<div class="form-horizontal">

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.Label("Назва тарифу", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name\_Of\_Tariff, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Name\_Of\_Tariff, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.Label("Вартість", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Cost\_Of\_Tariff, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Cost\_Of\_Tariff, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.Label("Швидкість", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Speed\_Of\_Internet, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Speed\_Of\_Internet, "", new { @class = "text-danger" })

</div>

</div>

3.2.6.Перегляд та редагування посад

Для перегляду посад працівників потрібно увійти користувачу типу «Працівник». Після переходу на вкладку «Посади» користувач має змогу побачити всі доступні посадити та зможе редагувати потрібну посаду.

Фрагмент коду:

public ActionResult Index()

{

var position = db.Position.Include(p => p.Salary);

return View(position.ToList());

}

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Position position = db.Position.Find(id);

if (position == null)

{

return HttpNotFound();

}

return View(position);

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "ID\_Position,Name\_Of\_Position,Salary")] Position position)

{

if (ModelState.IsValid)

{

db.Position.Add(position);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(position);

}

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Position position = db.Position.Find(id);

if (position == null)

{

return HttpNotFound();

}

return View(position);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "ID\_Position,Name\_Of\_Position,Salary")] Position position)

{

if (ModelState.IsValid)

{

db.Entry(position).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(position);

}

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Position position = db.Position.Find(id);

if (position == null)

{

return HttpNotFound();

}

return View(position);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Position position = db.Position.Find(id);

db.Position.Remove(position);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

<div class="form-horizontal">

<h4>Посади</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.Label("Назва посади", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name\_Of\_Position, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Name\_Of\_Position, "", new { @class = "text-danger" })

</div>

</div>

3.2.7.Поповнення балансу

Для поповнення балансу потрібно увійти користувачу типу «Клієнт», та перейти на вкладку «Контракт». Далі клієнт має змогу переглянути деталі свого контракту та поповнити баланс.

Фрагмент коду:

public ActionResult Index()

{

var idEmp = Convert.ToInt32(HttpContext.User.Identity.Name);

var cust = (from s in db.Contract

join cc in db.Customers on s.ID\_Customer equals cc.ID\_Customer

where cc.ID\_Customer == idEmp

select s);

return View(cust.ToList());

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "ID\_Contract,ID\_Employee,ID\_Customer,ID\_Tariff\_Plan,Date\_Of\_Signing")] Contract contract, List<string> names)

{

var summOfAdd = Convert.ToInt32(names[0]);

// string authData = $"Dat: {summ}";

var idEmp = Convert.ToInt32(HttpContext.User.Identity.Name);

var contracts = db.Contract.Include(p => p.Tariff\_Plan).Where(l => l.ID\_Customer == idEmp).FirstOrDefault();

var custBalance = db.Customers.Where(l => l.ID\_Customer == idEmp).FirstOrDefault();

if (ModelState.IsValid)

{

custBalance.Balance = custBalance.Balance + summOfAdd;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(contract);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

<form method="post">

<div class="form-horizontal">

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.Label("Сумма", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

<input id="names" name="names" type="text" value="" />

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Поповнити" class="btn btn-default" />

</div>

</div>

</form>

3.2.8.Оплата за тариф

Для оплати за тариф потрібно увійти користувачу типу «Клієнт», та перейти на вкладку «Оплата». Далі клієнт має змогу переглянути історію оплат.

Фрагмент коду:

public ActionResult Index()

{

var payment = db.Payment.Include(p => p.Contract);

var idEmp = Convert.ToInt32(HttpContext.User.Identity.Name);

var cust = (from pay in db.Payment

join con in db.Contract on pay.ID\_Contract equals con.ID\_Contract

where con.ID\_Customer == idEmp

select pay);

return View(cust.ToList());

}

public ActionResult Create()

{

var idEmp = Convert.ToInt32(HttpContext.User.Identity.Name);

ViewBag.ID\_Contract = new SelectList(db.Contract.Where(c => c.ID\_Customer == idEmp), "ID\_Contract", "ID\_Contract");

var contract = db.Contract.Include(p => p.Tariff\_Plan).Where(l => l.ID\_Customer == idEmp).FirstOrDefault();

ViewBag.Sum = new SelectList(db.Tariff\_Plan.Where(k => k.ID\_Tariff\_Plan == contract.Tariff\_Plan.ID\_Tariff\_Plan), "Cost\_Of\_Tariff", "Cost\_Of\_Tariff");

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "ID\_Payment,ID\_Contract,Sum,Date\_Of\_Payment")] Payment payment)

{

var idEmp = Convert.ToInt32(HttpContext.User.Identity.Name);

var contract = db.Contract.Include(p => p.Tariff\_Plan).Where(l => l.ID\_Customer == idEmp).FirstOrDefault();

var custBalance = db.Customers.Where(l => l.ID\_Customer == idEmp).FirstOrDefault();

using (var transaction = db.Database.BeginTransaction())

{

try

{

db.Database.ExecuteSqlCommand(@"UPDATE Payment SET Sum + 10 WHERE ID\_Payment == 3033");

transaction.Commit();

}

catch(Exception ex)

{

transaction.Rollback();

}

if (ModelState.IsValid)

{

if (custBalance.Balance >= contract.Tariff\_Plan.Cost\_Of\_Tariff)

{

custBalance.Balance = custBalance.Balance - contract.Tariff\_Plan.Cost\_Of\_Tariff;

db.Payment.Add(payment);

db.SaveChanges();

return RedirectToAction("Index");

}

else

{

//return RedirectToAction("Index");

return RedirectToRoute(new { controller = "ContractsCUST", action = "Create" });

}

}

}

ViewBag.ID\_Contract = new SelectList(db.Contract, "ID\_Contract", "ID\_Contract", payment.ID\_Contract);

return View(payment);

}

<div class="form-horizontal">

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.Label("Номер контракту", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("ID\_Contract", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.ID\_Contract, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.Label("Сума", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("Sum", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Sum, "", new { @class = "text-danger" })

</div>

</div>

* 1. Опис роботи програми

Робота з додатком розпочинається з авторизації, за допомогою якої визначається клас користувача та надається йому певний набір функціональних характеристик і обмежень, з якими від буде працювати протягом сеансу.

Авторизація користувача включає інформацію логін та пароль користувача, які містяться в базі даних у таблицях Employee та Customers.

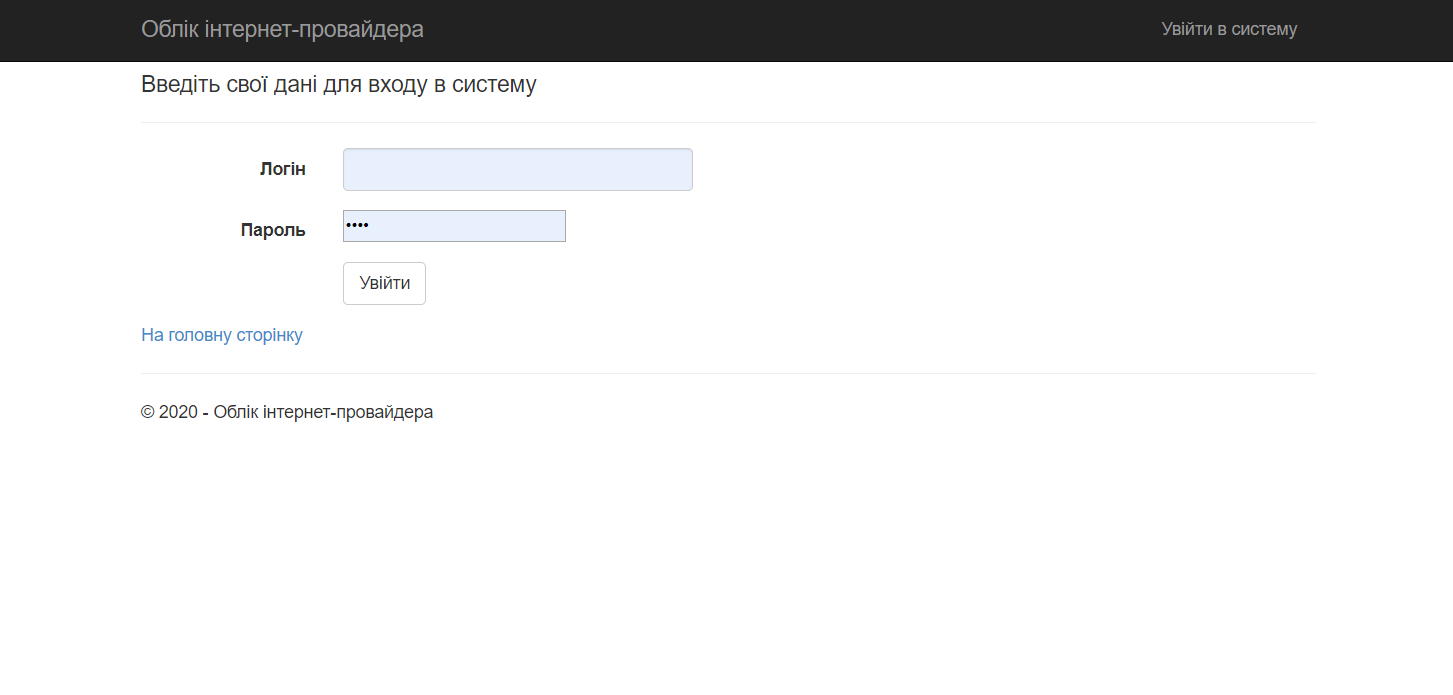


Рис. 3.2. Форма авторизації користувача

У випадку введення недостовірної інформації, користувачу буде подано повідомлення про помилку авторизації.

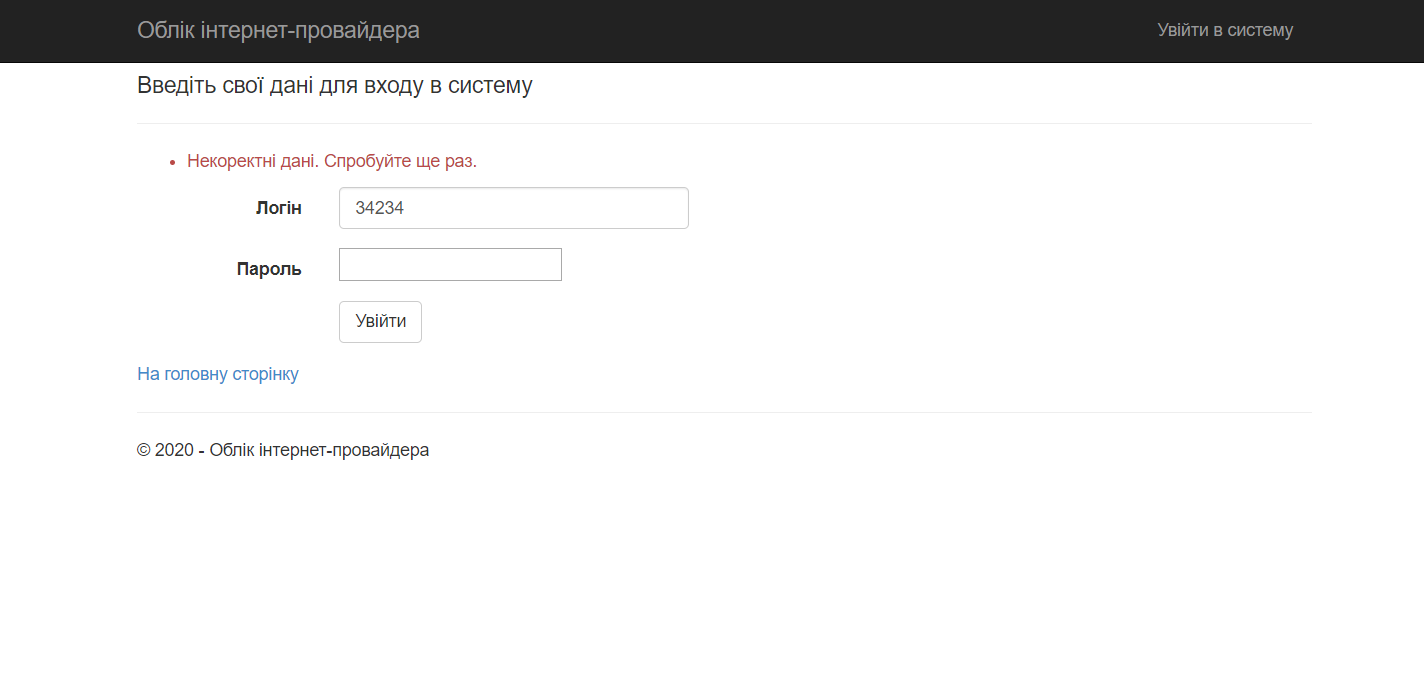


Рис. 3.3. Введення недостовірної інформації

Для авторизованих користувачів з’являється відповідні влкдаки. Також в залежності від ролі користувача стають доступними певні функціональні можливості системи.

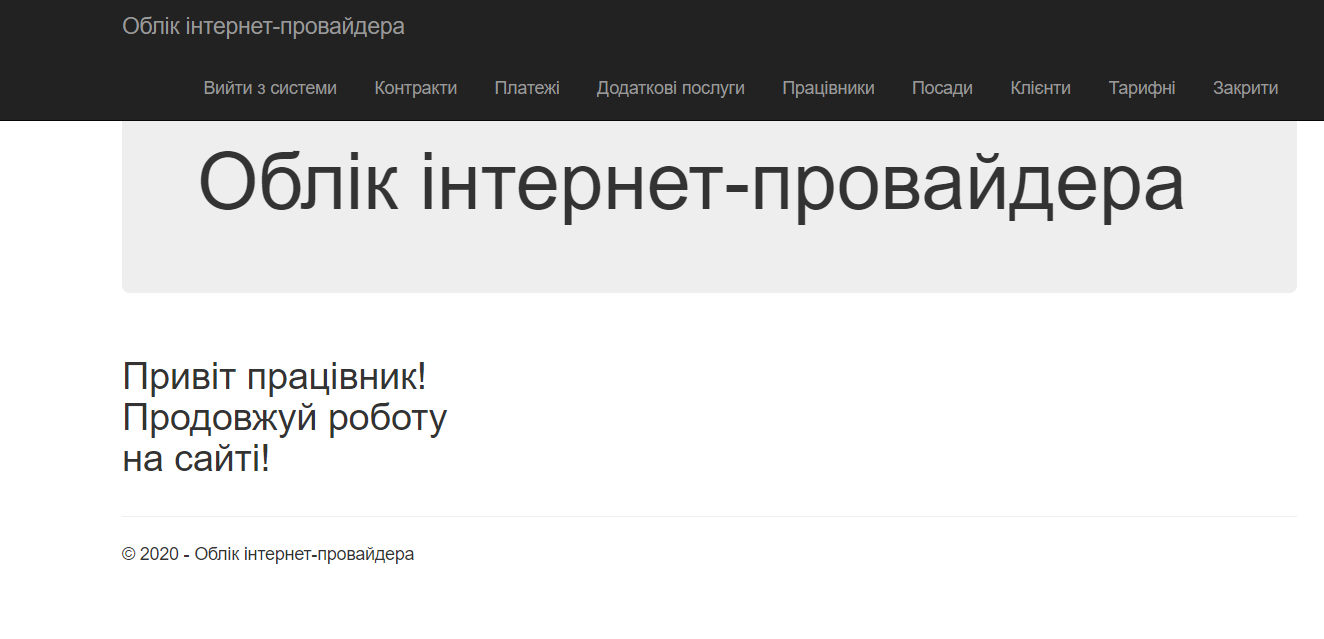


Рис.3.4. Головне вікно програми, для користувача типу «Працівник»

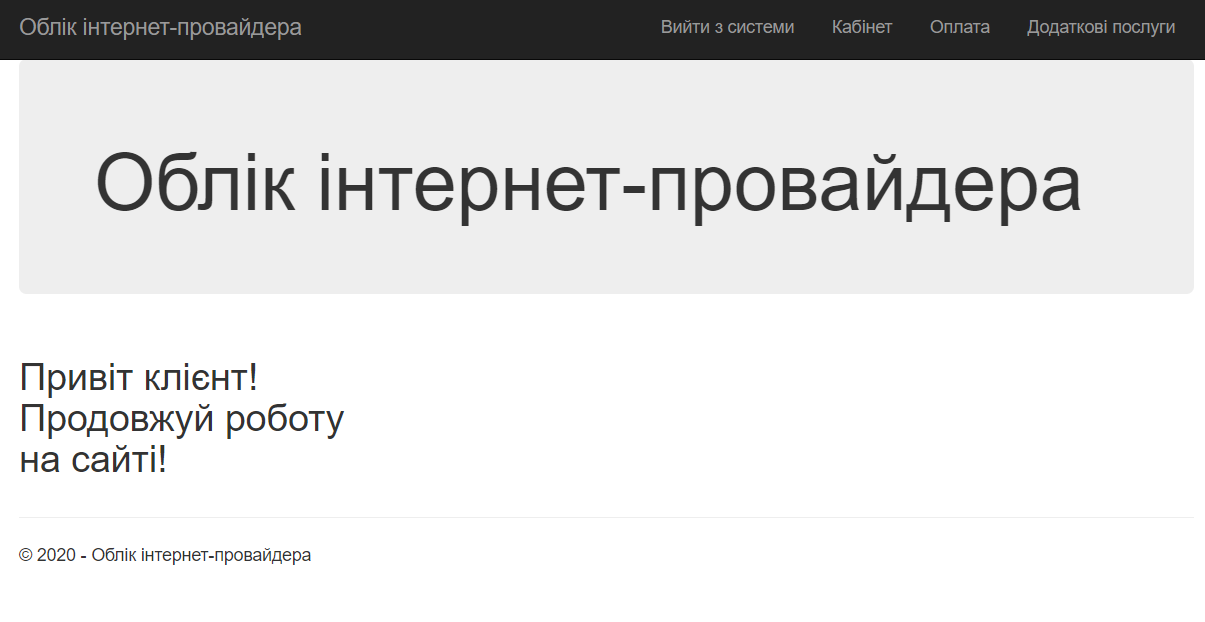


Рис.3.5. Головне вікно програми, для користувача типу «Клієнт»

При авторизації користувача типу «Працівник» йому доступні наступні функції.

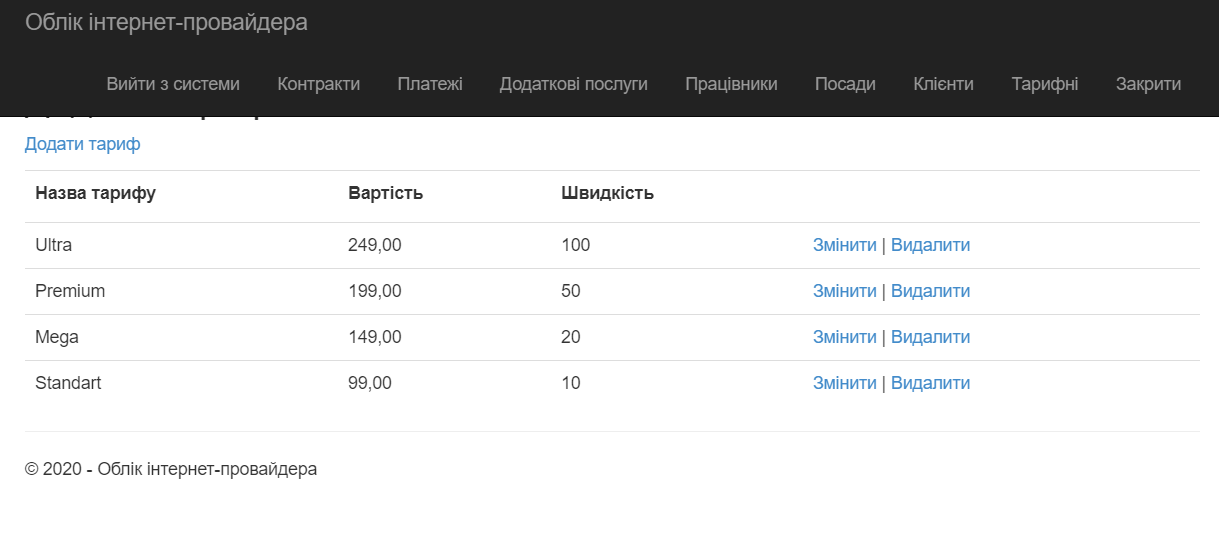


Рис.3.6. Таблиця тарифів

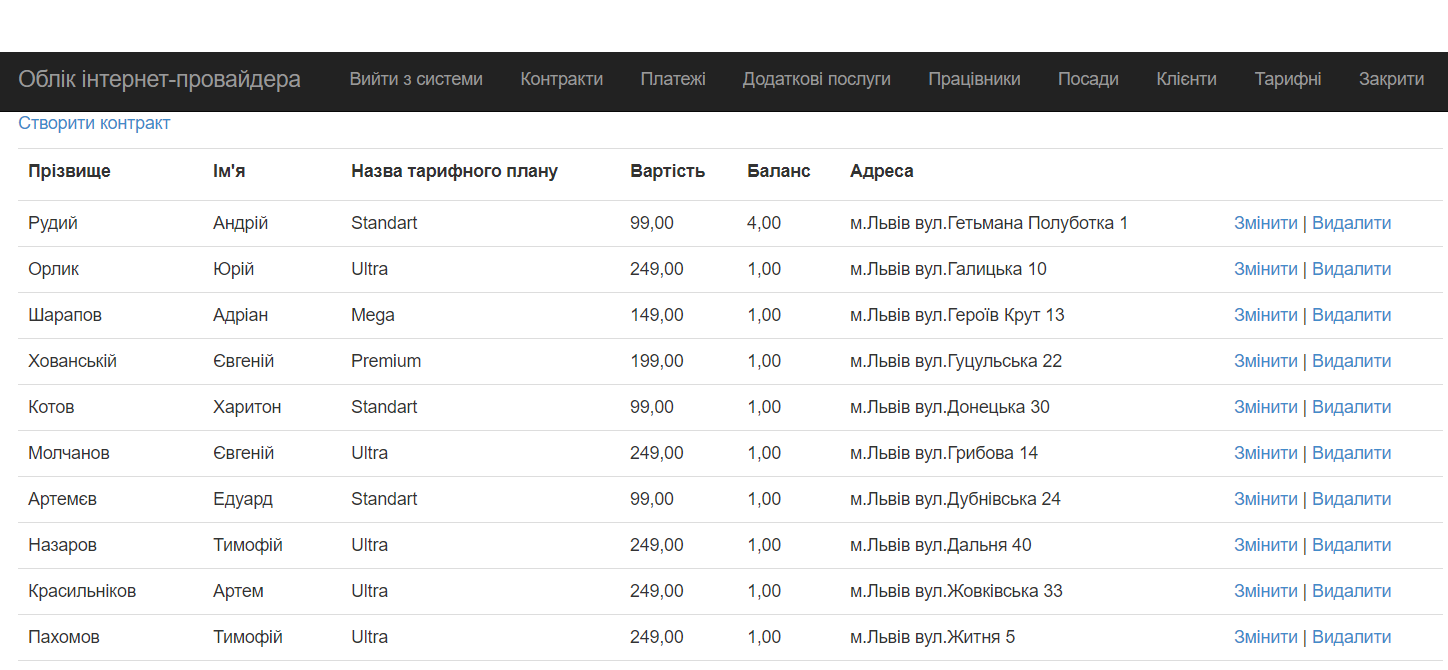


Рис.3.7. Таблиця контрактів

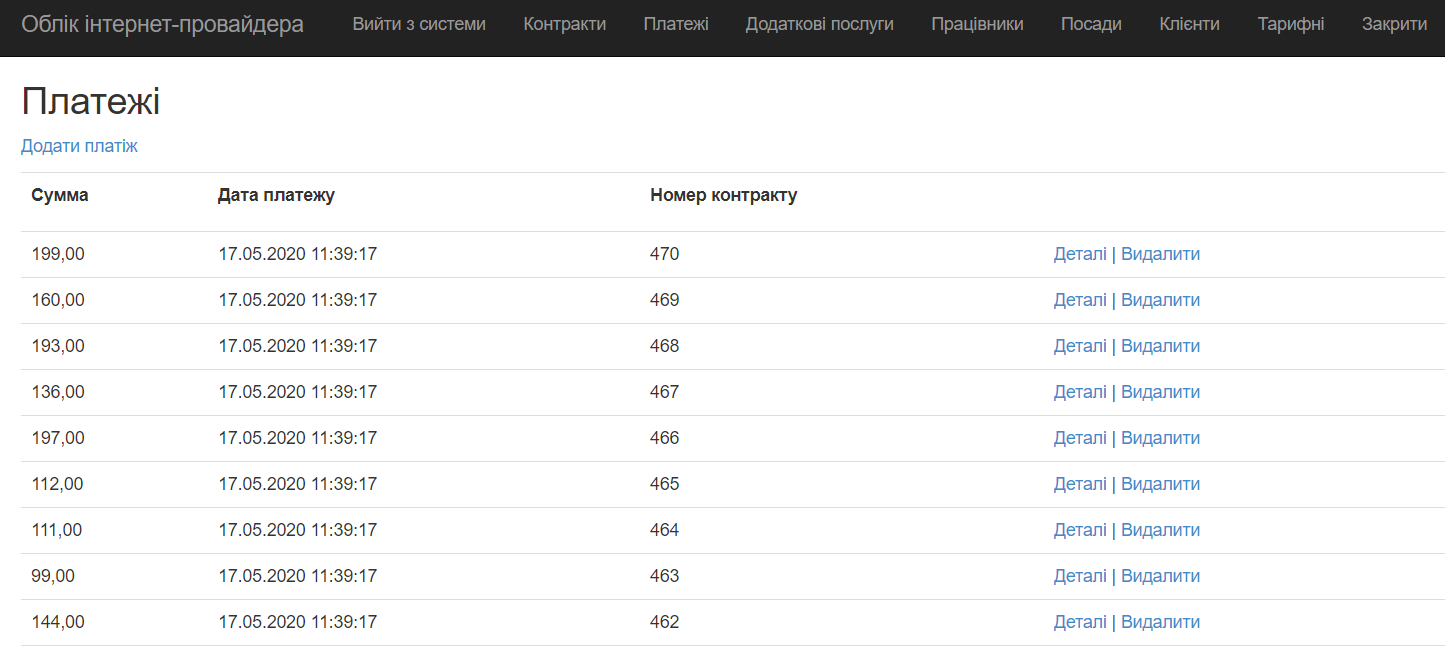


Рис.3.8. Таблиця оплат

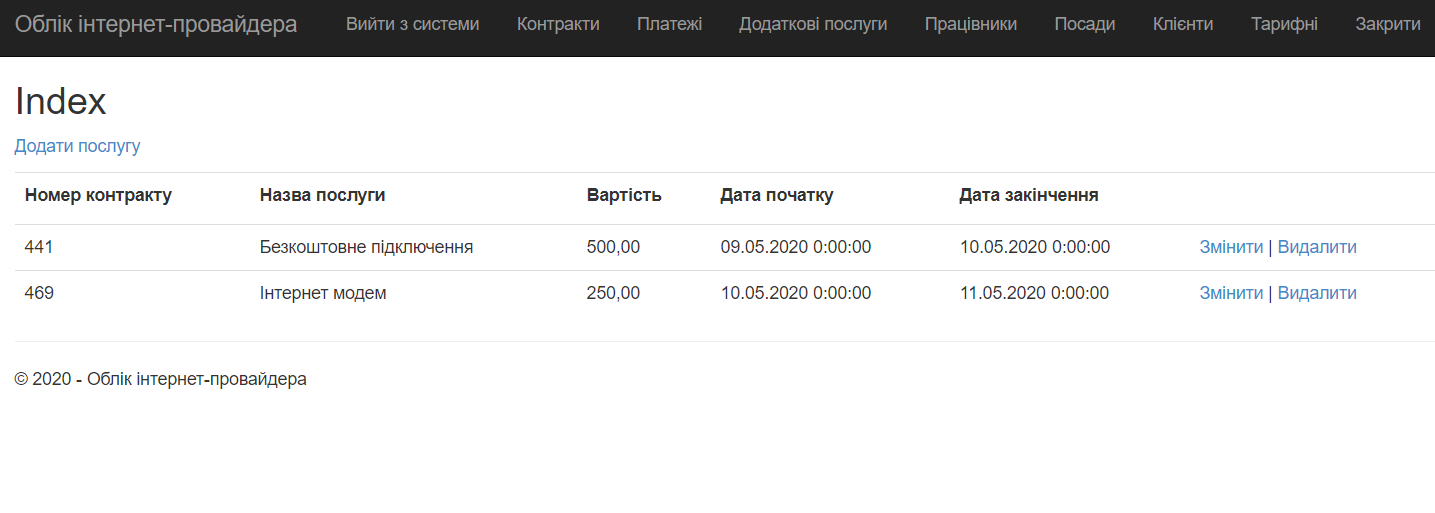


Рис.3.9. Таблиця додаткових послуг

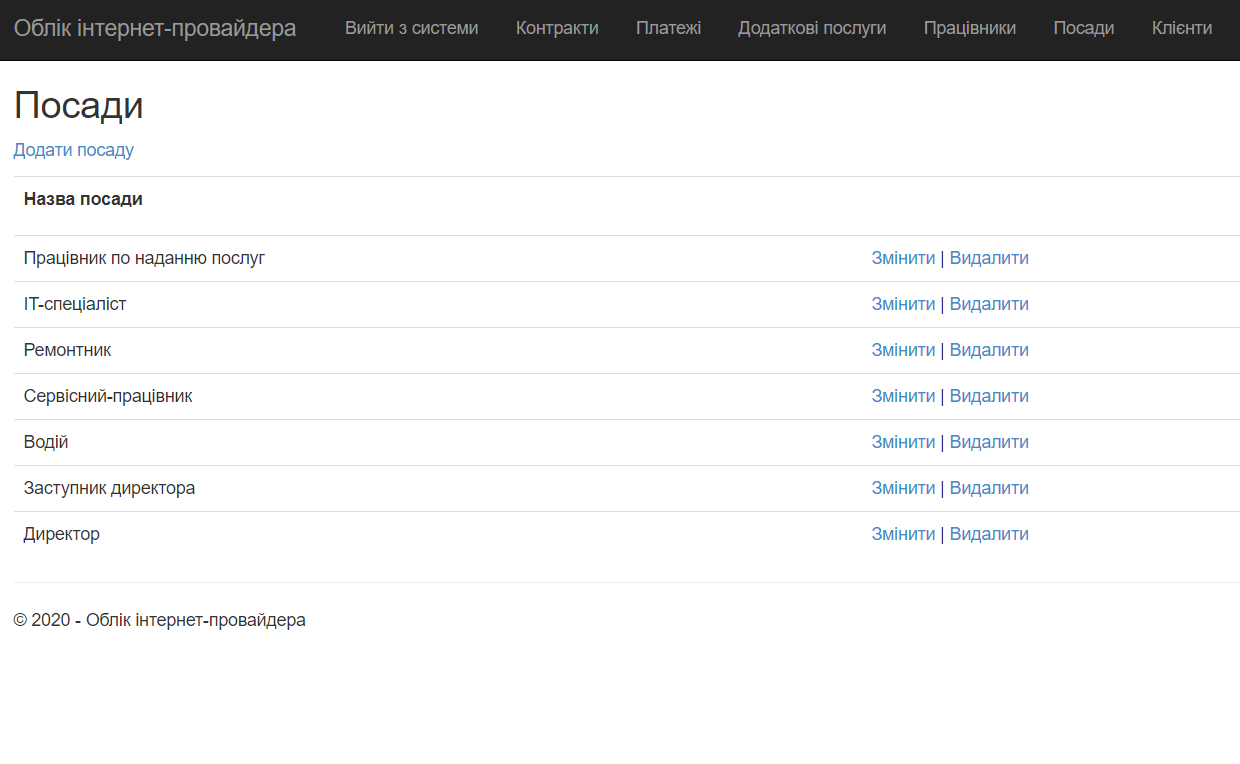


Рис.3.10. Таблиця посад

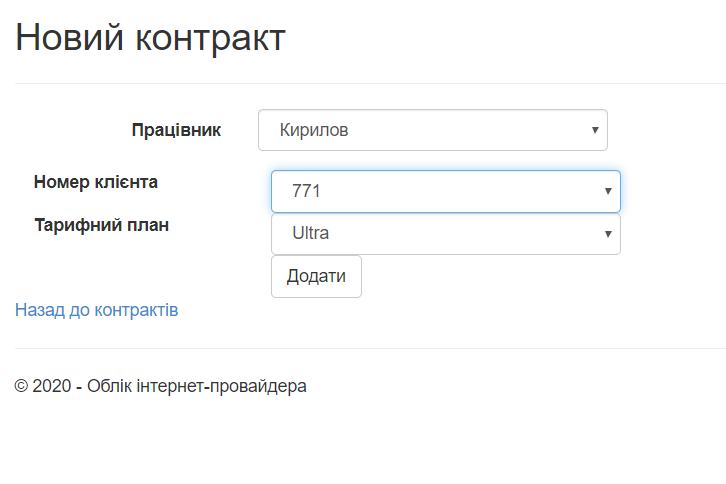


Рис.3.11. Відкриття меню для додавання нового контракту

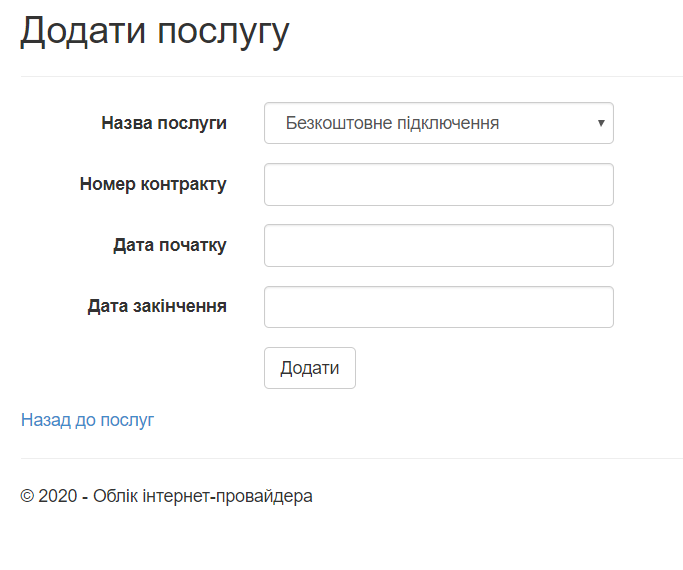


Рис.3.12. Відкриття меню для додавання додаткової послуги

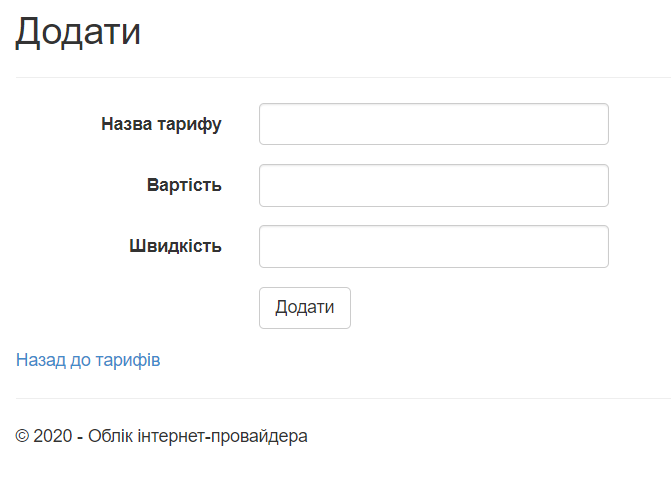


Рис.3.13. Відкриття меню для додавання нового тарифу

При авторизації користувача типу «Клієнт» йому доступні наступні функції.

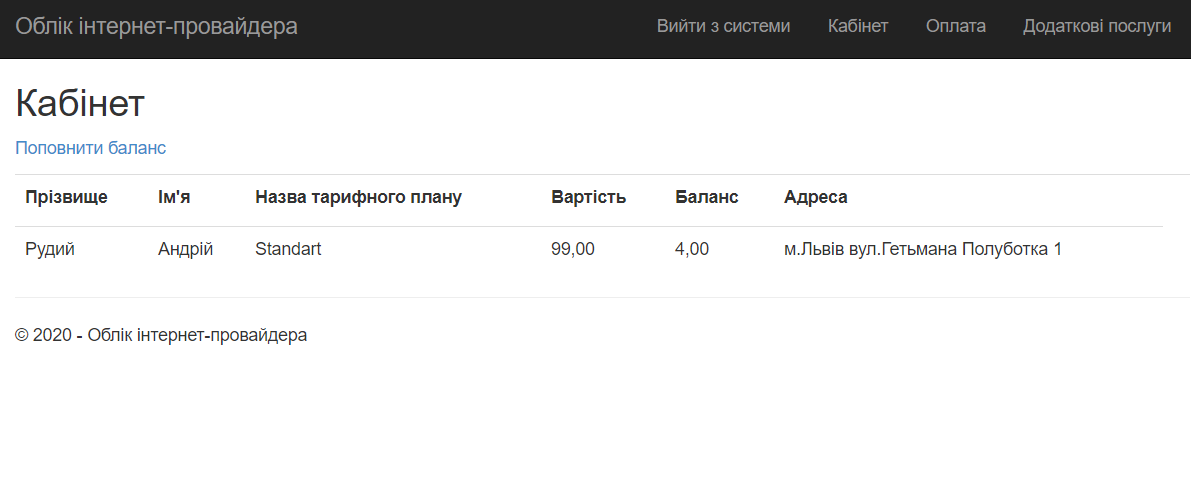


Рис.3.14. Перегляд контракту

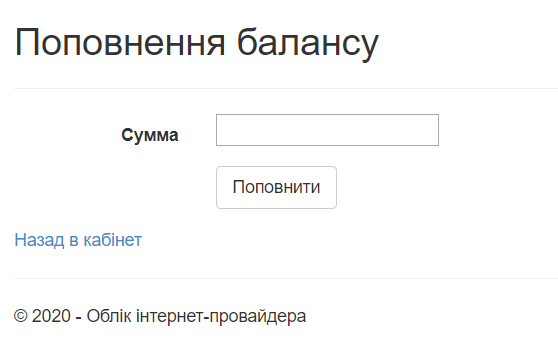


Рис.3.15. Поповнення балансу

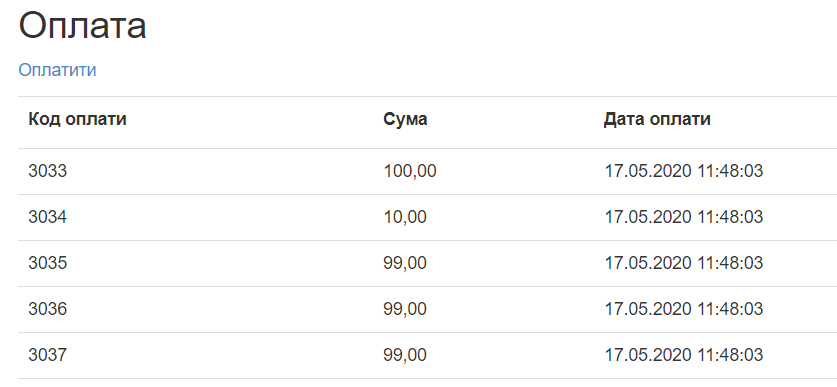


Рис.3.16. Перегляд оплат за тариф

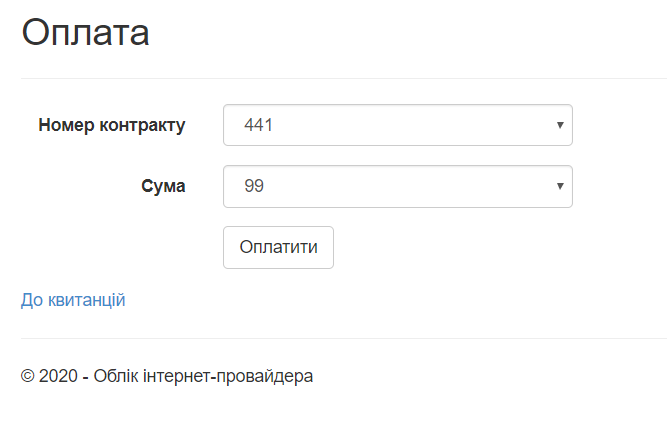


Рис.3.17. Оплата тарифу

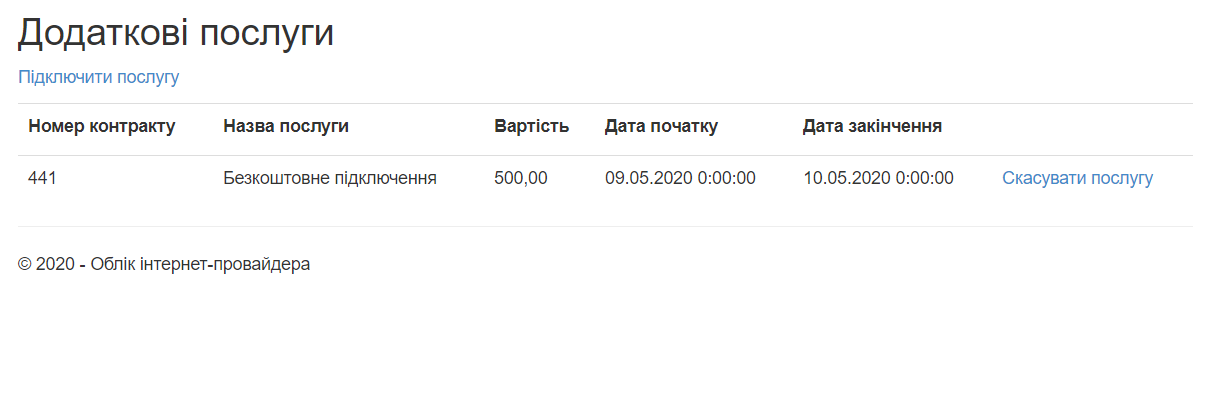


Рис.3.18. Перегляд додаткових послуг

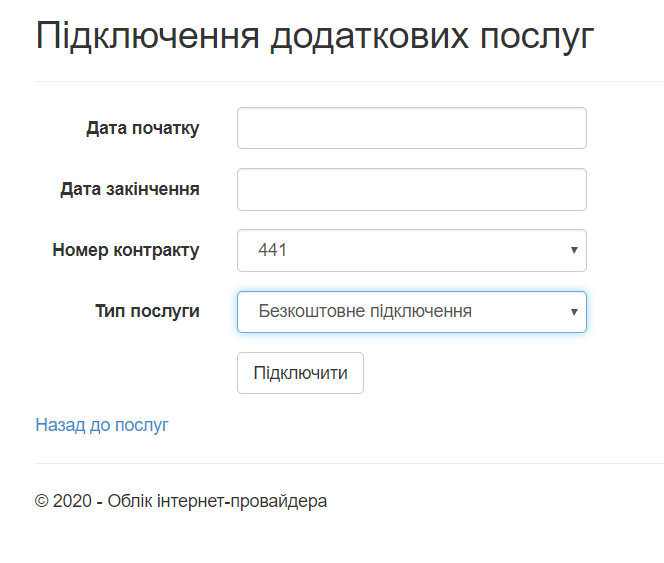


Рис.3.19. Підключення додаткової послуги

ВИСНОВOK

Виконуючи дану курсову роботу головною ціллю було створити програму для управління базою даних магазину електронної техніки.

Для проектування бази даних було використано середовище CA Erwin Data Modeler, з допомогою якого також було згенеровано скрипт для фізичного представлення бази даних.

Для розробки програми було обрано мову програмування C# та технологію ASP.NET MVC.

Для взаємодії додатку з базою даних було обрано технологію Entity Framework.

Інтерфейс користувача було реалізовано в проекті з використанням ASP.NET MVC, Bootstrap, jQuery.

Для побудови чіткої та зрозумілої архітектури було обрано архітектурний шаблон MVC, який є зручним для розробки додатків такого типу.

СПИСОК ЛІТЕРАТУРИ

1. Дудзяний І. М. Об’єктно-орієнтоване моделювання програмних систем. Львів: Видавництво ЛНУ ім. Івана Франка, 2007. 107 с.
2. Мак-Дональд М. WPF 4: Windows Presentation Foundation в NET 4.0 с примерами на C# 2010 для профессионалов. М.: Вильямс, 2011. 1024 с.
3. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. Київ: Видавнича група BHV, 2006. – 384 с.
4. Петкович Д. Microsoft SQL Server 2012. Руководство для начинающих: пер. с англ. СПб.: БХВ-Петербург, 2013. 816 с.
5. Руководство по WPF. *METANIT.COM Сайт о программировании.* URL: [https://metanit.com /sharp/wpf/](https://metanit.com/sharp/wpf/).
6. Entity Framework Database First. *MSDN: Introduction to Entity Framework.* URL: <https://msdn.microsoft.com/en-us/library/jj206878(v=vs.113).aspx>
7. Introduction to WPF *MSDN: Windows Presentation Foundation.* URL: <https://msdn.microsoft.com/ru-ru/library/aa970268(v=vs.100).aspx>
8. SQL Server Documentation URL: <https://docs.microsoft.com/ru-ru/sql/sql-server/sql-server-technical-documentation>
9. Using SQL Views with Entity Framework Code First. URL:

<https://laboremus.ug/?p=1708>

Додаток А. Скрипт створення БД

CREATE TABLE Additional\_Services

(

ID\_Contract INTEGER NOT NULL ,

Start\_Date DATE NULL ,

End\_Date DATE NULL ,

ID\_Services INTEGER NOT NULL

);

CREATE UNIQUE INDEX XPKServices ON Additional\_Services

(ID\_Services ASC);

ALTER TABLE Additional\_Services

ADD CONSTRAINT XPKServices PRIMARY KEY (ID\_Services);

CREATE TABLE Closing\_Contract

(

ID\_Contract INTEGER NOT NULL ,

Closing\_Date DATE NULL ,

Reason VARCHAR2(20) NULL

);

CREATE UNIQUE INDEX XPKЗакриття\_договору ON Closing\_Contract

(ID\_Contract ASC);

ALTER TABLE Closing\_Contract

ADD CONSTRAINT XPKЗакриття\_договору PRIMARY KEY (ID\_Contract);

CREATE TABLE Contract

(

ID\_Customer INTEGER NOT NULL ,

Date\_Of\_Signing DATE NULL ,

ID\_Contract INTEGER NOT NULL ,

ID\_Employee INTEGER NOT NULL ,

ID\_Tariff\_Plan INTEGER NOT NULL

);

CREATE UNIQUE INDEX XPKДоговір ON Contract

(ID\_Contract ASC);

ALTER TABLE Contract

ADD CONSTRAINT XPKДоговір PRIMARY KEY (ID\_Contract);

CREATE TABLE Customers

(

ID\_Customer CHAR(18) NOT NULL

);

CREATE UNIQUE INDEX XPKКлієнти ON Customers

(ID\_Customer ASC);

ALTER TABLE Customers

ADD CONSTRAINT XPKКлієнти PRIMARY KEY (ID\_Customer);

CREATE TABLE Employee

(

ID\_Position CHAR(18) NOT NULL ,

ID\_Employee INTEGER NOT NULL ,

First\_Name VARCHAR2(20) NULL ,

Phone\_Number CHAR(18) NULL

);

CREATE UNIQUE INDEX XPKПрацівники ON Employee

(ID\_Employee ASC);

ALTER TABLE Employee

ADD CONSTRAINT XPKПрацівники PRIMARY KEY (ID\_Employee);

CREATE TABLE Law

(

Company\_Name VARCHAR2(20) NULL ,

Physical\_Adress VARCHAR2(20) NULL ,

Phone\_Number CHAR(18) NULL ,

ID\_Customer CHAR(18) NOT NULL

);

CREATE UNIQUE INDEX XPKЮридичні ON Law

(ID\_Customer ASC);

ALTER TABLE Law

ADD CONSTRAINT XPKЮридичні PRIMARY KEY (ID\_Customer);

CREATE TABLE Payment

(

ID\_Payment INTEGER NOT NULL ,

Sum INTEGER NULL ,

Date\_Of\_Payment DATE NULL ,

ID\_Contract INTEGER NOT NULL

);

CREATE UNIQUE INDEX XPKОплата ON Payment

(ID\_Payment ASC);

ALTER TABLE Payment

ADD CONSTRAINT XPKОплата PRIMARY KEY (ID\_Payment);

CREATE TABLE Physical

(

Physical\_Adress VARCHAR2(20) NULL ,

Number\_Of\_Passport VARCHAR2(20) NULL ,

Surname VARCHAR2(20) NULL ,

First\_Name VARCHAR2(20) NULL ,

Phone\_Number CHAR(18) NULL ,

ID\_Customer CHAR(18) NOT NULL

);

CREATE UNIQUE INDEX XPKФізичні ON Physical

(ID\_Customer ASC);

ALTER TABLE Physical

ADD CONSTRAINT XPKФізичні PRIMARY KEY (ID\_Customer);

CREATE TABLE Position

(

ID\_Position CHAR(18) NOT NULL ,

Name\_Of\_Position CHAR(18) NULL

);

CREATE UNIQUE INDEX XPKПосади ON Position

(ID\_Position ASC);

ALTER TABLE Position

ADD CONSTRAINT XPKПосади PRIMARY KEY (ID\_Position);

CREATE TABLE Salary

(

ID\_Position CHAR(18) NOT NULL ,

Salary CHAR(18) NULL

);

CREATE UNIQUE INDEX XPKЗарплата ON Salary

(ID\_Position ASC);

ALTER TABLE Salary

ADD CONSTRAINT XPKЗарплата PRIMARY KEY (ID\_Position);

CREATE TABLE Services

(

ID\_Services INTEGER NOT NULL ,

Name\_Of\_Services VARCHAR2(20) NULL ,

Price CHAR(18) NULL ,

Duration CHAR(18) NULL

);

CREATE UNIQUE INDEX XPKAdditional\_Service ON Services

(ID\_Services ASC);

ALTER TABLE Services

ADD CONSTRAINT XPKAdditional\_Service PRIMARY KEY (ID\_Services);

CREATE TABLE Tariff\_Plan

(

ID\_Tariff\_Plan CHAR(18) NOT NULL ,

Name\_Of\_Tariff INTEGER NULL ,

Cost\_Of\_Tariff INTEGER NULL ,

Speed\_Of\_Internet VARCHAR2(20) NULL

);

CREATE UNIQUE INDEX XPKТарифний\_план ON Tariff\_Plan

(ID\_Tariff\_Plan ASC);

ALTER TABLE Tariff\_Plan

ADD CONSTRAINT XPKТарифний\_план PRIMARY KEY (ID\_Tariff\_Plan);

ALTER TABLE Additional\_Services

ADD (CONSTRAINT R\_28 FOREIGN KEY (ID\_Contract) REFERENCES Contract(ID\_Contract));

ALTER TABLE Additional\_Services

ADD (CONSTRAINT R\_30 FOREIGN KEY (ID\_Services) REFERENCES Services(ID\_Services));

ALTER TABLE Closing\_Contract

ADD (CONSTRAINT R\_1 FOREIGN KEY (ID\_Contract) REFERENCES Contract(ID\_Contract));

ALTER TABLE Contract

ADD (CONSTRAINT R\_6 FOREIGN KEY (ID\_Customer) REFERENCES Customers(ID\_Customer));

ALTER TABLE Contract

ADD (CONSTRAINT R\_7 FOREIGN KEY (ID\_Tariff\_Plan) REFERENCES Tariff\_Plan(ID\_Tariff\_Plan));

ALTER TABLE Contract

ADD (CONSTRAINT R\_10 FOREIGN KEY (ID\_Employee) REFERENCES Employee(ID\_Employee));

ALTER TABLE Customers

ADD (CONSTRAINT R\_31 FOREIGN KEY (ID\_Customer) REFERENCES Physical(ID\_Customer));

ALTER TABLE Customers

ADD (CONSTRAINT R\_32 FOREIGN KEY (ID\_Customer) REFERENCES Law(ID\_Customer));

ALTER TABLE Employee

ADD (CONSTRAINT R\_11 FOREIGN KEY (ID\_Position) REFERENCES Position(ID\_Position));

ALTER TABLE Payment

ADD (CONSTRAINT R\_22 FOREIGN KEY (ID\_Contract) REFERENCES Contract(ID\_Contract));

ALTER TABLE Position

ADD (CONSTRAINT R\_16 FOREIGN KEY (ID\_Position) REFERENCES Salary(ID\_Position));

CREATE TRIGGER tI\_Additional\_Services BEFORE INSERT ON Additional\_Services for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- INSERT trigger on Additional\_Services

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/28 Additional\_Services on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0002255e", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Additional\_Services"

P2C\_VERB\_PHRASE="R/28", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_28", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(:%New,Contract," = "," AND") \*/

:new.ID\_Contract = Contract.ID\_Contract;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Additional\_Services because Contract does not exist.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Services R/30 Additional\_Services on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Services"

CHILD\_OWNER="", CHILD\_TABLE="Additional\_Services"

P2C\_VERB\_PHRASE="R/30", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_30", FK\_COLUMNS="ID\_Services" \*/

SELECT count(\*) INTO NUMROWS

FROM Services

WHERE

/\* %JoinFKPK(:%New,Services," = "," AND") \*/

:new.ID\_Services = Services.ID\_Services;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Additional\_Services because Services does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Additional\_Services AFTER UPDATE ON Additional\_Services for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Additional\_Services

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/28 Additional\_Services on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00021c1e", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Additional\_Services"

P2C\_VERB\_PHRASE="R/28", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_28", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(:%New,Contract," = "," AND") \*/

:new.ID\_Contract = Contract.ID\_Contract;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Additional\_Services because Contract does not exist.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Services R/30 Additional\_Services on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Services"

CHILD\_OWNER="", CHILD\_TABLE="Additional\_Services"

P2C\_VERB\_PHRASE="R/30", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_30", FK\_COLUMNS="ID\_Services" \*/

SELECT count(\*) INTO NUMROWS

FROM Services

WHERE

/\* %JoinFKPK(:%New,Services," = "," AND") \*/

:new.ID\_Services = Services.ID\_Services;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Additional\_Services because Services does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tI\_Closing\_Contract BEFORE INSERT ON Closing\_Contract for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- INSERT trigger on Closing\_Contract

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/1 Closing\_Contract on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0001004c", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Closing\_Contract"

P2C\_VERB\_PHRASE="R/1", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_1", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(:%New,Contract," = "," AND") \*/

:new.ID\_Contract = Contract.ID\_Contract;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Closing\_Contract because Contract does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Closing\_Contract AFTER UPDATE ON Closing\_Contract for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Closing\_Contract

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/1 Closing\_Contract on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000105dd", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Closing\_Contract"

P2C\_VERB\_PHRASE="R/1", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_1", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(:%New,Contract," = "," AND") \*/

:new.ID\_Contract = Contract.ID\_Contract;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Closing\_Contract because Contract does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tI\_Contract BEFORE INSERT ON Contract for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- INSERT trigger on Contract

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Customers R/6 Contract on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00032651", PARENT\_OWNER="", PARENT\_TABLE="Customers"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/6", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_6", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Customers

WHERE

/\* %JoinFKPK(:%New,Customers," = "," AND") \*/

:new.ID\_Customer = Customers.ID\_Customer;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Contract because Customers does not exist.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Tariff\_Plan R/7 Contract on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Tariff\_Plan"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/7", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_7", FK\_COLUMNS="ID\_Tariff\_Plan" \*/

SELECT count(\*) INTO NUMROWS

FROM Tariff\_Plan

WHERE

/\* %JoinFKPK(:%New,Tariff\_Plan," = "," AND") \*/

:new.ID\_Tariff\_Plan = Tariff\_Plan.ID\_Tariff\_Plan;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Contract because Tariff\_Plan does not exist.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Employee R/10 Contract on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Employee"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/10", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_10", FK\_COLUMNS="ID\_Employee" \*/

SELECT count(\*) INTO NUMROWS

FROM Employee

WHERE

/\* %JoinFKPK(:%New,Employee," = "," AND") \*/

:new.ID\_Employee = Employee.ID\_Employee;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Contract because Employee does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Contract AFTER DELETE ON Contract for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Contract

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/1 Closing\_Contract on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0003065b", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Closing\_Contract"

P2C\_VERB\_PHRASE="R/1", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_1", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Closing\_Contract

WHERE

/\* %JoinFKPK(Closing\_Contract,:%Old," = "," AND") \*/

Closing\_Contract.ID\_Contract = :old.ID\_Contract;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Contract because Closing\_Contract exists.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/22 Payment on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Payment"

P2C\_VERB\_PHRASE="R/22", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_22", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Payment

WHERE

/\* %JoinFKPK(Payment,:%Old," = "," AND") \*/

Payment.ID\_Contract = :old.ID\_Contract;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Contract because Payment exists.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/28 Additional\_Services on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Additional\_Services"

P2C\_VERB\_PHRASE="R/28", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_28", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Additional\_Services

WHERE

/\* %JoinFKPK(Additional\_Services,:%Old," = "," AND") \*/

Additional\_Services.ID\_Contract = :old.ID\_Contract;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Contract because Additional\_Services exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Contract AFTER UPDATE ON Contract for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Contract

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/1 Closing\_Contract on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0006cc38", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Closing\_Contract"

P2C\_VERB\_PHRASE="R/1", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_1", FK\_COLUMNS="ID\_Contract" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Contract.ID\_Contract <> Contract.ID\_Contract

THEN

SELECT count(\*) INTO NUMROWS

FROM Closing\_Contract

WHERE

/\* %JoinFKPK(Closing\_Contract,:%Old," = "," AND") \*/

Closing\_Contract.ID\_Contract = :old.ID\_Contract;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Contract because Closing\_Contract exists.'

);

END IF;

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/22 Payment on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Payment"

P2C\_VERB\_PHRASE="R/22", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_22", FK\_COLUMNS="ID\_Contract" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Contract.ID\_Contract <> Contract.ID\_Contract

THEN

SELECT count(\*) INTO NUMROWS

FROM Payment

WHERE

/\* %JoinFKPK(Payment,:%Old," = "," AND") \*/

Payment.ID\_Contract = :old.ID\_Contract;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Contract because Payment exists.'

);

END IF;

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/28 Additional\_Services on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Additional\_Services"

P2C\_VERB\_PHRASE="R/28", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_28", FK\_COLUMNS="ID\_Contract" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Contract.ID\_Contract <> Contract.ID\_Contract

THEN

SELECT count(\*) INTO NUMROWS

FROM Additional\_Services

WHERE

/\* %JoinFKPK(Additional\_Services,:%Old," = "," AND") \*/

Additional\_Services.ID\_Contract = :old.ID\_Contract;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Contract because Additional\_Services exists.'

);

END IF;

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Customers R/6 Contract on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Customers"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/6", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_6", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Customers

WHERE

/\* %JoinFKPK(:%New,Customers," = "," AND") \*/

:new.ID\_Customer = Customers.ID\_Customer;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Contract because Customers does not exist.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Tariff\_Plan R/7 Contract on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Tariff\_Plan"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/7", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_7", FK\_COLUMNS="ID\_Tariff\_Plan" \*/

SELECT count(\*) INTO NUMROWS

FROM Tariff\_Plan

WHERE

/\* %JoinFKPK(:%New,Tariff\_Plan," = "," AND") \*/

:new.ID\_Tariff\_Plan = Tariff\_Plan.ID\_Tariff\_Plan;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Contract because Tariff\_Plan does not exist.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Employee R/10 Contract on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Employee"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/10", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_10", FK\_COLUMNS="ID\_Employee" \*/

SELECT count(\*) INTO NUMROWS

FROM Employee

WHERE

/\* %JoinFKPK(:%New,Employee," = "," AND") \*/

:new.ID\_Employee = Employee.ID\_Employee;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Contract because Employee does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tI\_Customers BEFORE INSERT ON Customers for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- INSERT trigger on Customers

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Physical R/31 Customers on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0001f780", PARENT\_OWNER="", PARENT\_TABLE="Physical"

CHILD\_OWNER="", CHILD\_TABLE="Customers"

P2C\_VERB\_PHRASE="R/31", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_31", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Physical

WHERE

/\* %JoinFKPK(:%New,Physical," = "," AND") \*/

:new.ID\_Customer = Physical.ID\_Customer;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Customers because Physical does not exist.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Law R/32 Customers on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Law"

CHILD\_OWNER="", CHILD\_TABLE="Customers"

P2C\_VERB\_PHRASE="R/32", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_32", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Law

WHERE

/\* %JoinFKPK(:%New,Law," = "," AND") \*/

:new.ID\_Customer = Law.ID\_Customer;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Customers because Law does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Customers AFTER DELETE ON Customers for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Customers

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Customers R/6 Contract on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000e35f", PARENT\_OWNER="", PARENT\_TABLE="Customers"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/6", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_6", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(Contract,:%Old," = "," AND") \*/

Contract.ID\_Customer = :old.ID\_Customer;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Customers because Contract exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Customers AFTER UPDATE ON Customers for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Customers

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Customers R/6 Contract on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000324fa", PARENT\_OWNER="", PARENT\_TABLE="Customers"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/6", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_6", FK\_COLUMNS="ID\_Customer" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Customers.ID\_Customer <> Customers.ID\_Customer

THEN

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(Contract,:%Old," = "," AND") \*/

Contract.ID\_Customer = :old.ID\_Customer;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Customers because Contract exists.'

);

END IF;

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Physical R/31 Customers on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Physical"

CHILD\_OWNER="", CHILD\_TABLE="Customers"

P2C\_VERB\_PHRASE="R/31", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_31", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Physical

WHERE

/\* %JoinFKPK(:%New,Physical," = "," AND") \*/

:new.ID\_Customer = Physical.ID\_Customer;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Customers because Physical does not exist.'

);

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Law R/32 Customers on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Law"

CHILD\_OWNER="", CHILD\_TABLE="Customers"

P2C\_VERB\_PHRASE="R/32", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_32", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Law

WHERE

/\* %JoinFKPK(:%New,Law," = "," AND") \*/

:new.ID\_Customer = Law.ID\_Customer;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Customers because Law does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tI\_Employee BEFORE INSERT ON Employee for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- INSERT trigger on Employee

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Position R/11 Employee on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000f706", PARENT\_OWNER="", PARENT\_TABLE="Position"

CHILD\_OWNER="", CHILD\_TABLE="Employee"

P2C\_VERB\_PHRASE="R/11", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_11", FK\_COLUMNS="ID\_Position" \*/

SELECT count(\*) INTO NUMROWS

FROM Position

WHERE

/\* %JoinFKPK(:%New,Position," = "," AND") \*/

:new.ID\_Position = Position.ID\_Position;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Employee because Position does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Employee AFTER DELETE ON Employee for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Employee

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Employee R/10 Contract on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000e3ed", PARENT\_OWNER="", PARENT\_TABLE="Employee"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/10", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_10", FK\_COLUMNS="ID\_Employee" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(Contract,:%Old," = "," AND") \*/

Contract.ID\_Employee = :old.ID\_Employee;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Employee because Contract exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Employee AFTER UPDATE ON Employee for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Employee

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Employee R/10 Contract on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0002315a", PARENT\_OWNER="", PARENT\_TABLE="Employee"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/10", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_10", FK\_COLUMNS="ID\_Employee" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Employee.ID\_Employee <> Employee.ID\_Employee

THEN

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(Contract,:%Old," = "," AND") \*/

Contract.ID\_Employee = :old.ID\_Employee;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Employee because Contract exists.'

);

END IF;

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Position R/11 Employee on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Position"

CHILD\_OWNER="", CHILD\_TABLE="Employee"

P2C\_VERB\_PHRASE="R/11", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_11", FK\_COLUMNS="ID\_Position" \*/

SELECT count(\*) INTO NUMROWS

FROM Position

WHERE

/\* %JoinFKPK(:%New,Position," = "," AND") \*/

:new.ID\_Position = Position.ID\_Position;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Employee because Position does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Law AFTER DELETE ON Law for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Law

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Law R/32 Customers on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000e5a1", PARENT\_OWNER="", PARENT\_TABLE="Law"

CHILD\_OWNER="", CHILD\_TABLE="Customers"

P2C\_VERB\_PHRASE="R/32", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_32", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Customers

WHERE

/\* %JoinFKPK(Customers,:%Old," = "," AND") \*/

Customers.ID\_Customer = :old.ID\_Customer;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Law because Customers exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Law AFTER UPDATE ON Law for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Law

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Law R/32 Customers on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000105e6", PARENT\_OWNER="", PARENT\_TABLE="Law"

CHILD\_OWNER="", CHILD\_TABLE="Customers"

P2C\_VERB\_PHRASE="R/32", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_32", FK\_COLUMNS="ID\_Customer" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Law.ID\_Customer <> Law.ID\_Customer

THEN

SELECT count(\*) INTO NUMROWS

FROM Customers

WHERE

/\* %JoinFKPK(Customers,:%Old," = "," AND") \*/

Customers.ID\_Customer = :old.ID\_Customer;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Law because Customers exists.'

);

END IF;

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tI\_Payment BEFORE INSERT ON Payment for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- INSERT trigger on Payment

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/22 Payment on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000f17a", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Payment"

P2C\_VERB\_PHRASE="R/22", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_22", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(:%New,Contract," = "," AND") \*/

:new.ID\_Contract = Contract.ID\_Contract;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Payment because Contract does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Payment AFTER UPDATE ON Payment for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Payment

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Contract R/22 Payment on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000f438", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Payment"

P2C\_VERB\_PHRASE="R/22", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_22", FK\_COLUMNS="ID\_Contract" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(:%New,Contract," = "," AND") \*/

:new.ID\_Contract = Contract.ID\_Contract;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Payment because Contract does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Physical AFTER DELETE ON Physical for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Physical

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Physical R/31 Customers on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000e5f1", PARENT\_OWNER="", PARENT\_TABLE="Physical"

CHILD\_OWNER="", CHILD\_TABLE="Customers"

P2C\_VERB\_PHRASE="R/31", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_31", FK\_COLUMNS="ID\_Customer" \*/

SELECT count(\*) INTO NUMROWS

FROM Customers

WHERE

/\* %JoinFKPK(Customers,:%Old," = "," AND") \*/

Customers.ID\_Customer = :old.ID\_Customer;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Physical because Customers exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Physical AFTER UPDATE ON Physical for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Physical

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Physical R/31 Customers on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00010f6c", PARENT\_OWNER="", PARENT\_TABLE="Physical"

CHILD\_OWNER="", CHILD\_TABLE="Customers"

P2C\_VERB\_PHRASE="R/31", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_31", FK\_COLUMNS="ID\_Customer" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Physical.ID\_Customer <> Physical.ID\_Customer

THEN

SELECT count(\*) INTO NUMROWS

FROM Customers

WHERE

/\* %JoinFKPK(Customers,:%Old," = "," AND") \*/

Customers.ID\_Customer = :old.ID\_Customer;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Physical because Customers exists.'

);

END IF;

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tI\_Position BEFORE INSERT ON Position for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- INSERT trigger on Position

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Salary R/16 Position on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000f856", PARENT\_OWNER="", PARENT\_TABLE="Salary"

CHILD\_OWNER="", CHILD\_TABLE="Position"

P2C\_VERB\_PHRASE="R/16", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_16", FK\_COLUMNS="ID\_Position" \*/

SELECT count(\*) INTO NUMROWS

FROM Salary

WHERE

/\* %JoinFKPK(:%New,Salary," = "," AND") \*/

:new.ID\_Position = Salary.ID\_Position;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20002,

'Cannot insert Position because Salary does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Position AFTER DELETE ON Position for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Position

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Position R/11 Employee on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000e8ba", PARENT\_OWNER="", PARENT\_TABLE="Position"

CHILD\_OWNER="", CHILD\_TABLE="Employee"

P2C\_VERB\_PHRASE="R/11", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_11", FK\_COLUMNS="ID\_Position" \*/

SELECT count(\*) INTO NUMROWS

FROM Employee

WHERE

/\* %JoinFKPK(Employee,:%Old," = "," AND") \*/

Employee.ID\_Position = :old.ID\_Position;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Position because Employee exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Position AFTER UPDATE ON Position for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Position

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Position R/11 Employee on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000217d7", PARENT\_OWNER="", PARENT\_TABLE="Position"

CHILD\_OWNER="", CHILD\_TABLE="Employee"

P2C\_VERB\_PHRASE="R/11", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_11", FK\_COLUMNS="ID\_Position" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Position.ID\_Position <> Position.ID\_Position

THEN

SELECT count(\*) INTO NUMROWS

FROM Employee

WHERE

/\* %JoinFKPK(Employee,:%Old," = "," AND") \*/

Employee.ID\_Position = :old.ID\_Position;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Position because Employee exists.'

);

END IF;

END IF;

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Salary R/16 Position on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Salary"

CHILD\_OWNER="", CHILD\_TABLE="Position"

P2C\_VERB\_PHRASE="R/16", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_16", FK\_COLUMNS="ID\_Position" \*/

SELECT count(\*) INTO NUMROWS

FROM Salary

WHERE

/\* %JoinFKPK(:%New,Salary," = "," AND") \*/

:new.ID\_Position = Salary.ID\_Position;

IF (

/\* %NotnullFK(:%New," IS NOT NULL AND") \*/

NUMROWS = 0

)

THEN

raise\_application\_error(

-20007,

'Cannot update Position because Salary does not exist.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Salary AFTER DELETE ON Salary for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Salary

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Salary R/16 Position on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000e852", PARENT\_OWNER="", PARENT\_TABLE="Salary"

CHILD\_OWNER="", CHILD\_TABLE="Position"

P2C\_VERB\_PHRASE="R/16", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_16", FK\_COLUMNS="ID\_Position" \*/

SELECT count(\*) INTO NUMROWS

FROM Position

WHERE

/\* %JoinFKPK(Position,:%Old," = "," AND") \*/

Position.ID\_Position = :old.ID\_Position;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Salary because Position exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Salary AFTER UPDATE ON Salary for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Salary

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Salary R/16 Position on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000113f5", PARENT\_OWNER="", PARENT\_TABLE="Salary"

CHILD\_OWNER="", CHILD\_TABLE="Position"

P2C\_VERB\_PHRASE="R/16", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_16", FK\_COLUMNS="ID\_Position" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Salary.ID\_Position <> Salary.ID\_Position

THEN

SELECT count(\*) INTO NUMROWS

FROM Position

WHERE

/\* %JoinFKPK(Position,:%Old," = "," AND") \*/

Position.ID\_Position = :old.ID\_Position;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Salary because Position exists.'

);

END IF;

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Services AFTER DELETE ON Services for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Services

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Services R/30 Additional\_Services on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0001019f", PARENT\_OWNER="", PARENT\_TABLE="Services"

CHILD\_OWNER="", CHILD\_TABLE="Additional\_Services"

P2C\_VERB\_PHRASE="R/30", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_30", FK\_COLUMNS="ID\_Services" \*/

SELECT count(\*) INTO NUMROWS

FROM Additional\_Services

WHERE

/\* %JoinFKPK(Additional\_Services,:%Old," = "," AND") \*/

Additional\_Services.ID\_Services = :old.ID\_Services;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Services because Additional\_Services exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Services AFTER UPDATE ON Services for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Services

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Services R/30 Additional\_Services on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00012430", PARENT\_OWNER="", PARENT\_TABLE="Services"

CHILD\_OWNER="", CHILD\_TABLE="Additional\_Services"

P2C\_VERB\_PHRASE="R/30", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_30", FK\_COLUMNS="ID\_Services" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Services.ID\_Services <> Services.ID\_Services

THEN

SELECT count(\*) INTO NUMROWS

FROM Additional\_Services

WHERE

/\* %JoinFKPK(Additional\_Services,:%Old," = "," AND") \*/

Additional\_Services.ID\_Services = :old.ID\_Services;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Services because Additional\_Services exists.'

);

END IF;

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tD\_Tariff\_Plan AFTER DELETE ON Tariff\_Plan for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- DELETE trigger on Tariff\_Plan

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Tariff\_Plan R/7 Contract on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0000e0bf", PARENT\_OWNER="", PARENT\_TABLE="Tariff\_Plan"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/7", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_7", FK\_COLUMNS="ID\_Tariff\_Plan" \*/

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(Contract,:%Old," = "," AND") \*/

Contract.ID\_Tariff\_Plan = :old.ID\_Tariff\_Plan;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20001,

'Cannot delete Tariff\_Plan because Contract exists.'

);

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END;

/

CREATE TRIGGER tU\_Tariff\_Plan AFTER UPDATE ON Tariff\_Plan for each row

-- ERwin Builtin 17 мая 2020 г. 11:58:35

-- UPDATE trigger on Tariff\_Plan

DECLARE NUMROWS INTEGER;

BEGIN

/\* ERwin Builtin 17 мая 2020 г. 11:58:35 \*/

/\* Tariff\_Plan R/7 Contract on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00011064", PARENT\_OWNER="", PARENT\_TABLE="Tariff\_Plan"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="R/7", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_7", FK\_COLUMNS="ID\_Tariff\_Plan" \*/

IF

/\* %JoinPKPK(:%Old,:%New," <> "," OR ") \*/

Tariff\_Plan.ID\_Tariff\_Plan <> Tariff\_Plan.ID\_Tariff\_Plan

THEN

SELECT count(\*) INTO NUMROWS

FROM Contract

WHERE

/\* %JoinFKPK(Contract,:%Old," = "," AND") \*/

Contract.ID\_Tariff\_Plan = :old.ID\_Tariff\_Plan;

IF (NUMROWS > 0)

THEN

raise\_application\_error(

-20005,

'Cannot update Tariff\_Plan because Contract exists.'

);

END IF;

END IF;

-- ERwin Builtin 17 мая 2020 г. 11:58:35

END