МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 01 02 03 Информационные системы

и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«База данных «Магазин бытовой техники»

с использованием технологии APEX»

Выполнил студент Потапейко П. П.

(Ф.И.О.)

Руководитель проекта ассистент Копыток Д.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2022

**Оглавление**

[Введение 3](#_Toc102953197)

[1. Постановка задачи 4](#_Toc102953198)

[2. Проектирование базы данных 5](#_Toc102953199)

[3. Разработка объектов базы данных 6](#_Toc102953200)

[3.1 Таблицы 6](#_Toc102953201)

[3.2 Пользователи 8](#_Toc102953202)

[3.3 Хранимые процедуры 8](#_Toc102953203)

[3.4 Триггеры 9](#_Toc102953204)

[4. Описание процедур импорта и экспорта 10](#_Toc102953205)

[5. Тестирование производительности 11](#_Toc102953206)

[6. Описание технологии и ее применения в базе данных 13](#_Toc102953207)

[7. Руководство пользователя 16](#_Toc102953208)

[Заключение 22](#_Toc102953209)

[Cписок источников 23](#_Toc102953210)

[Приложение А 24](#_Toc102953211)

[Приложение Б 27](#_Toc102953212)

[Приложение В 29](#_Toc102953213)

## **Введение**

В современном мире доступ к информации является важным приоритетом любой компании и организации. Роль посредника между информацией и конечным пользователем давно занимают базы данных. Базы данных обеспечивают хранение и обработку информации в структурированном виде, а также доступ к этой информации.

Со временем любая информация усложняется, дополняется новыми формами представления данных, создаются новые технологии для обработки и манипулирования, а также доступа к данным

Темой курсового проекта является разработка приложения программы базы данных «Магазин бытовой техники».

Приложение «Магазин бытовой техники» имеет доступ к таблицам в базе данных и предназначено для взаимодействия с данными. Под взаимодействием понимается получение данных, их представление в определенном формате для просмотра пользователем и изменение данных.

Приложение базы данных «Магазин бытовой техники» призвано упростить работу сотрудников магазина бытовой техники, а также предоставляет возможность администратору своевременно вносить необходимые изменения в базу данных.

1. **Постановка задачи**

Задача проекта: разработать архитектуру приложения, создать интерфейс с использованием технологии APEX, взаимодействие с которым будет понятно любому пользователю. Построить базу данных и интерфейс и выполнить тестирование готового продукта.

Функционально должны быть выполнены следующие задачи:

* разработка приложения для взаимодействия с базой данных с использованием технологии APEX;
* возможность оформить заказ на товар;
* возможность осуществлять поиск товара.

Должны быть выполнены следующие требования:

* доступ к данным должен осуществляться только через соответствующие процедуры;
* необходимо протестировать производительность базы данных на таблице, содержащей не менее 100 000 строк, и внести изменения в структуру в случае необходимости. Необходимо проанализировать планы запросов к таблице;
* применить технологию базы данных согласно выбранной теме: подробно описать применяемые системные пакеты, утилиты или технологии; показать применение указанной технологии в базе данных.

Форма реализации продукта – веб-приложение.

Веб-приложение должно быть реализовано с использованием технологии APEX.

1. **Проектирование базы данных**

На первом этапе выполнения курсового проекта нужно создать логически взаимосвязанные таблицы. Чтобы спроектировать взаимосвязанную структуру базы данных, необходимо определить данные, которые будут храниться в таблицах, затем создать связи с помощью первичных и внешних ключей.

Диаграмма базы данных, спроектированной в ходе разработки показана на рисунке 2.1.

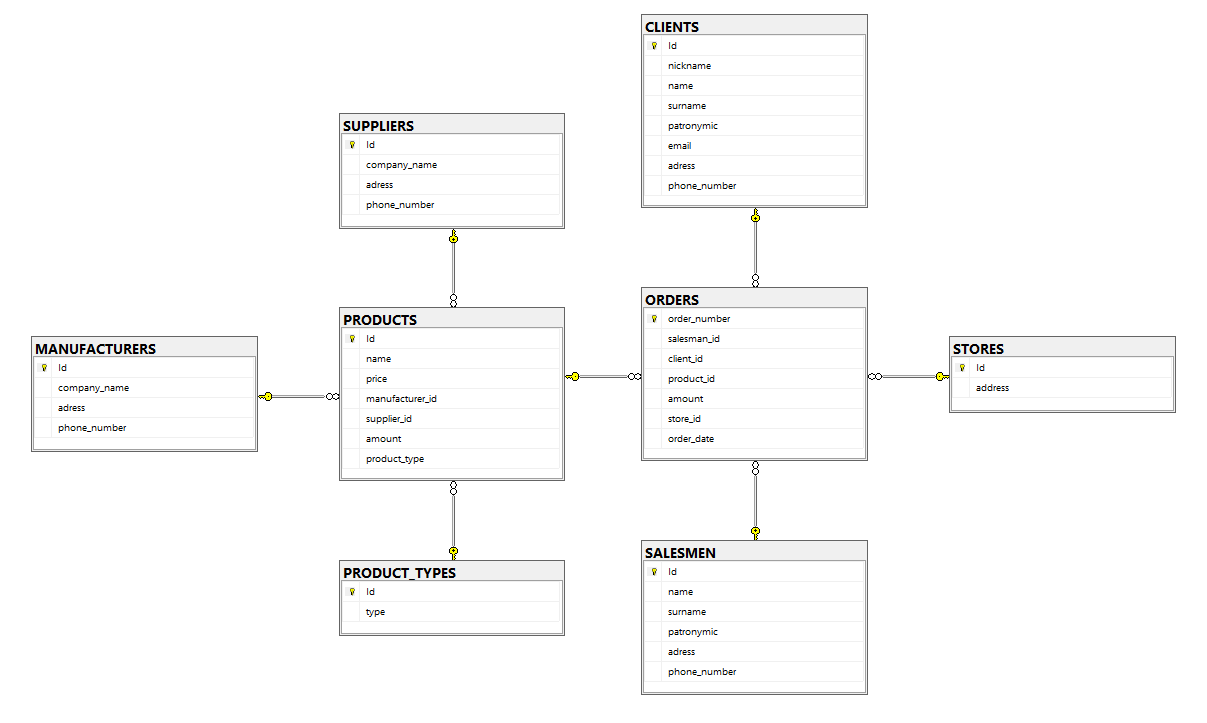


Рисунок 2.1 – Диаграмма базы данных

База данных приложения состоит из 8 таблиц, взаимосвязанных между собой внешними ключами:

* PRODUCTS;
* PRODUCT\_TYPE;
* SUPPLIERS;
* MANUFACTURERS;
* ORDERS;
* CLIENTS;
* STORES;
* SALESMEN.

1. **Разработка объектов базы данных**

# **Таблицы**

Для реализации базы данных «Магазин бытовой техники» было разработано 8 таблиц: PRODUCTS, PRODUCT\_TYPES, SUPPLIERS, MANUFACTURERS, ORDERS, CLIENTS, STORES, SALESMEN.

Таблица PRODUCTS представляет собой список товаров, состоит из столбцов (таблица 3.1):

Таблица 3.1 – Столбцы таблицы PRODUCTS

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор продукта, первичный ключ | number |
| name | наименование товара | nvarchar2(200) |
| price | цена товара | number(10, 0) |
| manufacturer id | идентификатор производителя, внешний ключ для связи с таблицей MANUFACTURERS | number |
| supplier id | идентификатор поставщика, внешний ключ для связи с таблицей SUPPLIERS | number |
| amount | количество | number |
| product type | тип продукта | number |

Таблица PRODUCT\_TYPES представляет собой список типов продуктов, состоит из столбцов (таблица 3.2):

Таблица 3.2 – Столбцы таблицы PRODUCT\_TYPES

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор типа продукта, первичный ключ | number |
| type | наименование типа продукта | nvarchar2(200) |

Таблица SUPPLIERS представляет собой список поставщиков, состоит из столбцов (таблица 3.3):

Таблица 3.3 – Столбцы таблицы SUPPLIERS

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор поставщика, первичный ключ | number |
| company name | название компании поставщика | nvarchar2(200) |
| adress | адрес | nvarchar2(200) |
| phone number | номер телефона | nvarchar2(50) |

Таблица MANUFACTURERS представляет собой список производителей, состоит из столбцов (таблица 3.4):

Таблица 3.4 – Столбцы таблицы MANUFACTURERS

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор производителя, первичный ключ | number |
| company name | название компании производителя | nvarchar2(200) |
| adress | адрес | nvarchar2(200) |
| phone number | номер телефона | nvarchar2(50) |

Таблица CLIENTS представляет собой список клиентов, состоит из столбцов (таблица 3.5):

Таблица 3.5 – Столбцы таблицы CLIENTS

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор клиента, первичный ключ | number |
| nickname | имя пользователя | nvarchar2(200) |
| name | имя клиента | nvarchar2(50) |
| surname | фамилия клиента | nvarchar2(50) |
| patronymic | отчество клиента | nvarchar2(50) |
| email | адрес электронной почты | nvarchar2(200) |
| address | адрес | nvarchar2(100) |
| phone number | телефон | nvarchar2(100) |

Таблица STORES представляет собой список магазинов, состоит из столбцов (таблица 3.6):

Таблица 3.6 – Столбцы таблицы STORES

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор магазина, первичный ключ | number |
| address | адрес | nvarchar2(200) |

Таблица SALESMEN представляет собой список продавцов, состоит из столбцов (таблица 3.7):

Таблица 3.7 – Столбцы таблицы SALESMEN

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор продавца, первичный ключ | number |
| name | имя продавца | nvarchar2(50) |
| surname | фамилия продавца | nvarchar2(50) |
| patronymic | отчество продавца | nvarchar2(50) |
| address | адрес | nvarchar2(100) |
| phone number | телефон | nvarchar2(50) |

Таблица ORDERS представляет собой список заказов клиентов, состоит из столбцов (таблица 3.8):

Таблица 3.8 – Столбцы таблицы ORDERS

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| order number | номер заказа, первичный ключ | number |
| salesman id | идентификатор продавца, внешний ключ для связи с таблицей SALESMEN | number |
| client id | идентификатор клиента, внешний ключ для связи с таблицей CLIENTS | number |
| product id | идентификатор товара, внешний ключ для связи с таблицей PRODUCTS | number |
| amount | количество | number |
| store id | идентификатор магазина, внешний ключ для связи с таблицей STORES | number |
| order date | дата осуществления заказа | date |

# **Пользователи**

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к базе данных и пользуется услугами информационной системы для получения информации.

При проектировании базы данных было использовано 2 пользователя. Первый пользователь – сотрудник, который является системным администратором, имеет доступ для чтения и изменения таблиц базы данных.

Второй пользователь – клиент – имеет доступ к таблице с товарами и может оформить заказ товара.

# **Хранимые процедуры**

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. При создании хранимой процедуры можно определить необязательный список параметров. Таким образом, процедура будет принимать соответствующие аргументы при каждом ее вызове. Все хранимые процедуры, созданные в данном курсовом проекте, содержат обработку исключений, для того, чтобы ограничить пользователя от возможных ошибок.

При разработке курсового проекта было создано 26 процедур для следующих целей:

* добавление/удаление/изменение товара;
* добавление/удаление/изменение типа товара;
* добавление/удаление/изменение поставщика;
* добавление/удаление/изменение производителя;
* добавление/удаление/изменение магазина;
* добавление/удаление/изменение продавца;
* добавление/удаление/изменение клиента;
* добавление/удаление/изменение заказа;
* экспорт и импорт таблиц в формат xml.

# **Триггеры**

Триггер – хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а выполнение которой обусловлено действием по модификации данных в таблице. Триггеры различаются по типу привязанного объекта (на таблице, на представлении), по событиям запуска (insert, update, delete), по области действия (уровень всей команды, уровень записи, составные триггеры), также по времени срабатывания (перед выполнением операции, после выполнения операции).

Для базы данных «Магазин бытовой техники» было создано по одному before-триггеру для каждой таблицы. Триггер срабатывает перед операцией вставки и проверяет, если происходит попытка вставки NULL-значения в столбец первичного ключа, то инициализируется принудительная вставка следующего значения соответствующей для таблицы последовательности.

1. **Описание процедур импорта и экспорта**

В курсовом проекте был произведен импорт данных из XML файлов и экспорт данных в формат XML.

XML – формат данных, предназначенный для обмена данными в форме, которая может быть легко использована и распространена.

Часто возникает необходимость импортировать и экспортировать XML-файлы в Oracle, в данной курсовой работе используются пакеты DBMS\_XMLPARSER и DBMS\_XSLPROCESSOR для импорта xml, и DBMS\_XMLDOM для экспорта данных в xml формат.

Для работы с файлами так же использовался пакет DBMS\_LOB.

Данные пакеты обладают широким функционалом и гибкой настройкой xml документа. В курсовом проекте функции экспорта и импорта используются для таблицы SUPPLIERS.

Пример создания процедуры экспорта представлен на рисунке 4.1.



Рисунок 4.1 – Процедура ExportSupplierstoXml

Для сохранения и управления XML-данными в реляционной таблице применяется специальный тип данных XMLType.

Тип данных XMLType поставляется с набором специальных XML-методов, которые можно использовать для работы с объектами XMLType.

Эти методы можно применять для выполнения как типичных операций в базе данных, например, проверки на предмет существования узла или извлечения узла, так и нескольких специальных операций, позволяющих получать доступ к XML-данными и манипулировать ими в виде части обычного SQL-оператора.

1. **Тестирование производительности**

Для проверки производительности базы данных необходимо заполнить ее большим количеством данных и узнать время выполнения одного запроса.

Для выполнения данной задачи был создан анонимный блок, в котором вызывается готовая процедура вставки данных в таблицу. Разработанный анонимный блок позволяет добавлять большое количество строк за одно выполнение (рисунок 5.1).

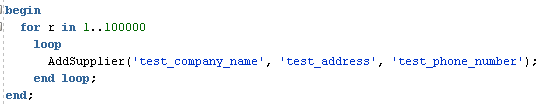


Рисунок 5.1 – Анонимный блок для заполнения таблицы большим количеством данных

Таким образом было добавлено 100000 тестовых строк в таблицу.

Затем во втором анонимном блоке, который содержит в себе дополнительно переменную для отслеживания времени выполнения, был сделан запрос, возвращающий 20000 строк. Данный анонимный блок представлен на рисунке 5.2.



Рисунок 5.2 – Блок отбора данных

После выполнения данного блока таблица будет содержать большое количество данных, и можно проанализировать время выполнения запроса. Также можно просмотреть план запроса используя стандартные средства Oracle: кнопку на главной панели, предварительно выделив данный запрос. Покажем результат, в котором будет заметна разница во времени выполнения запроса до создания индекса на рисунке 5.3.

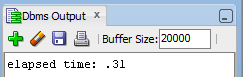


Рисунок 5.3 – Результат отбора данных

Теперь нужно создать индекс и протестировать время выполнения аналогичного запроса. Результат представлен на рисунке 5.4.



Рисунок 5.4 – Результат выполнения запроса после создания индекса

Время запроса уменьшилось, но даже без индекса база данных успешно прошла тест на производительность.

1. **Описание технологии и ее применения в базе данных**

В процессе разработки проекта была использована технология Oracle APEX (Oracle Application Express).

APEX – проприетарная среда быстрой разработки прикладного программного обеспечения на основе СУБД Oracle Database, целиком реализованная как веб-приложение.

APEX позволяет разрабатывать более быстрые и надежные профессиональные приложения, используя для этого лишь только веб-браузер и минимальный опыт программирования. Данная технология сочетает в себе высокую производительность, безопасность, интегрируемость и масштабируемость корпоративных БД, разработанных на основе веб-технологий, с простотой использования, доступностью и гибкостью персональных БД.

Разработка и запуск программ на APEX осуществляется посредством стандартного веб-браузера и не требует установки и обслуживания какого-либо клиентского ПО. Домашняя страница APEX показана на рисунке 6.1.

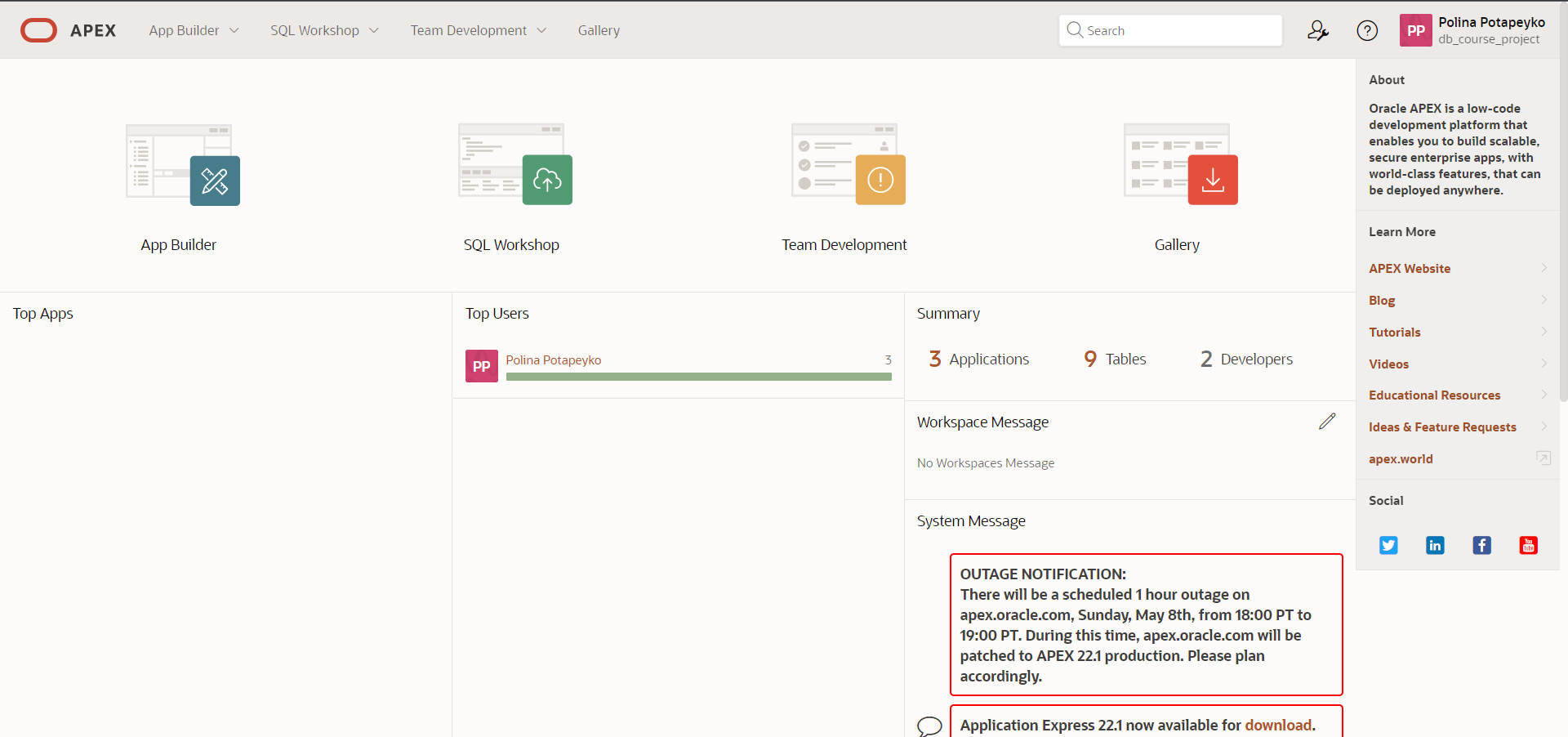


Рисунок 6.1 – Домашняя страница APEX

В числе основных возможностей Oracle APEX следующие:

* визуализация данных – платформа позволяет быстро искать и фильтровать данные с фасетным поиском, встраивать в приложение возможности создания настраиваемых пользовательских отчётов и использовать при разработки обширные инструменты построения интерактивных графиков;
* готовые к использованию приложения – в поставку включены полностью функциональные и готовые к использованию компоненты, предназначенные для удовлетворения наиболее популярных требований и покрытия рабочих процессов;
* расширяемые приложения – готовые приложения, построенные на Oracle Application Express, могут быть расширены и дополнительно настроены в соответствие с индивидуальными потребностями бизнеса;
* повышенная безопасность – платформа даёт возможность выбрать способ защиты приложений из списка SSO, LDAP или авторизация через социальные сети, также APEX-приложения могут включать отслеживание безопасности и аудит.

Основные инструменты APEX: Application Builder, SQL Workshop, Team Development.

Application Builder – инструмент, позволяющий разрабатывать динамические веб-приложения, управляемые БД. Страница Application Builder представлена на рисунке 6.2.

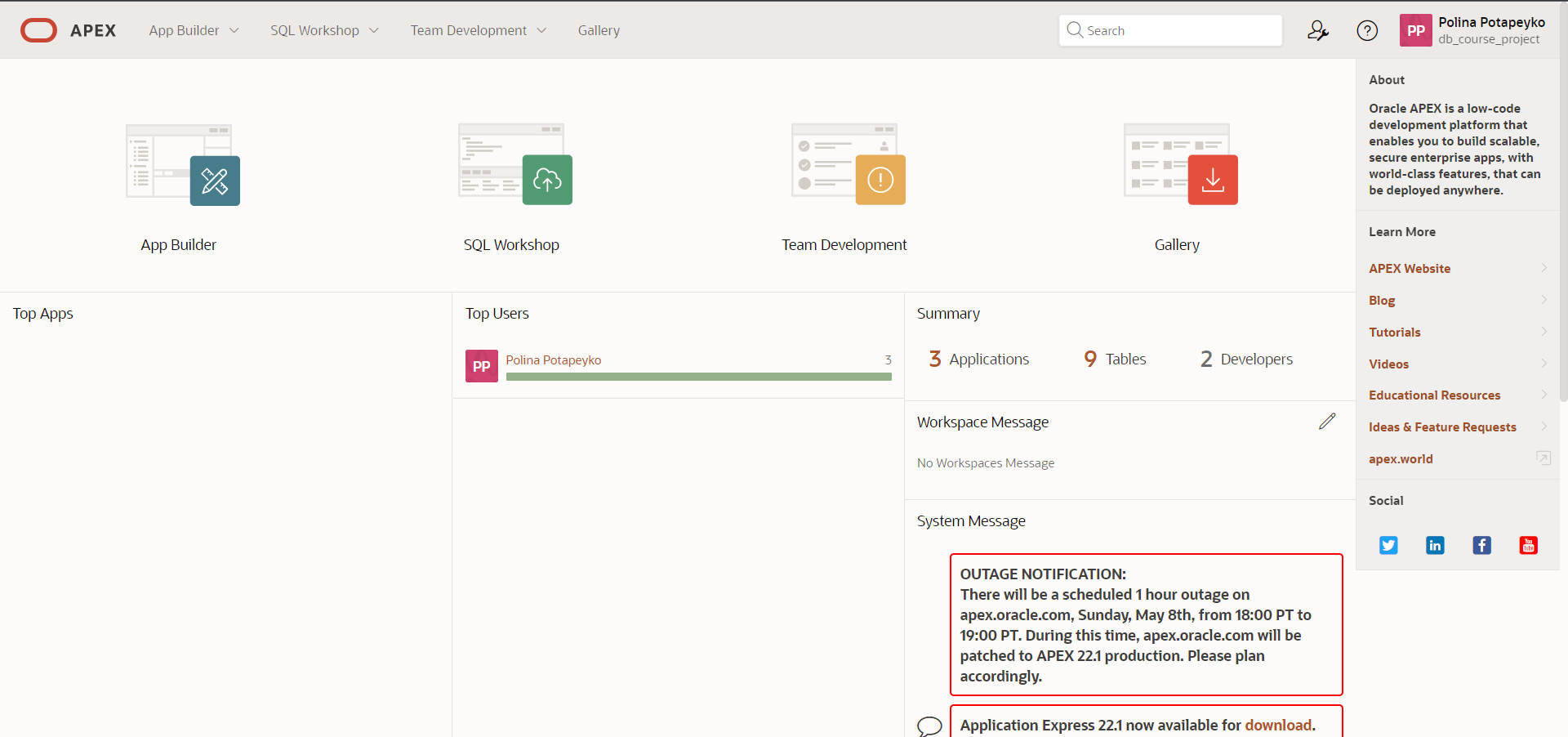


Рисунок 6.2 – Страница Application Builder

SQL Workshop – инструмент для просмотра содержимого БД, визуального построения запросов, в том числе и сложных SQL запросов. Страница SQL Workshop представлена на рисунке 6.3.

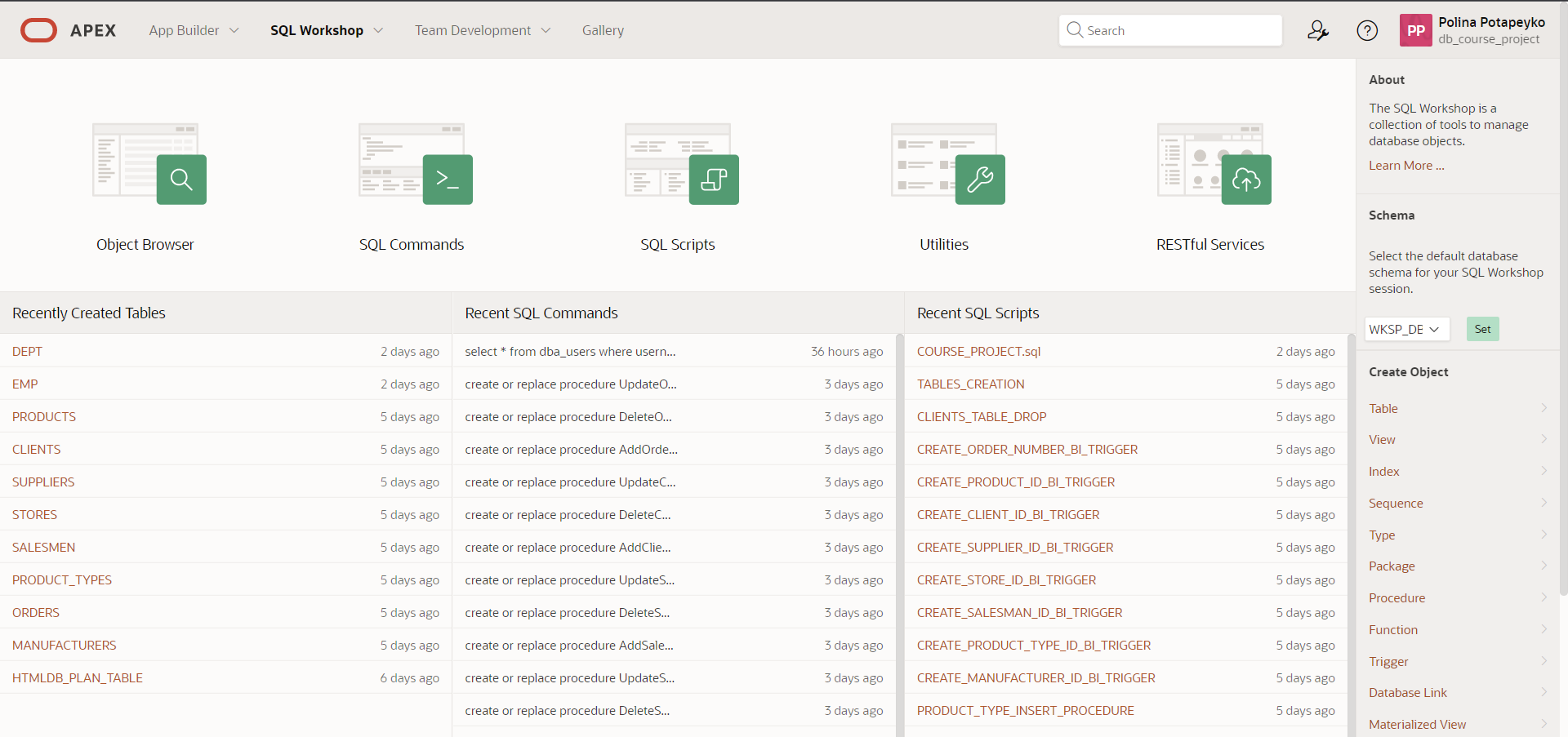


Рисунок 6.3 – Страница SQL Workshop

Team Development – инструмент, выполняющий экспорт и импорт разных форматов данных. Страница Team Development представлена на рисунке 6.4.

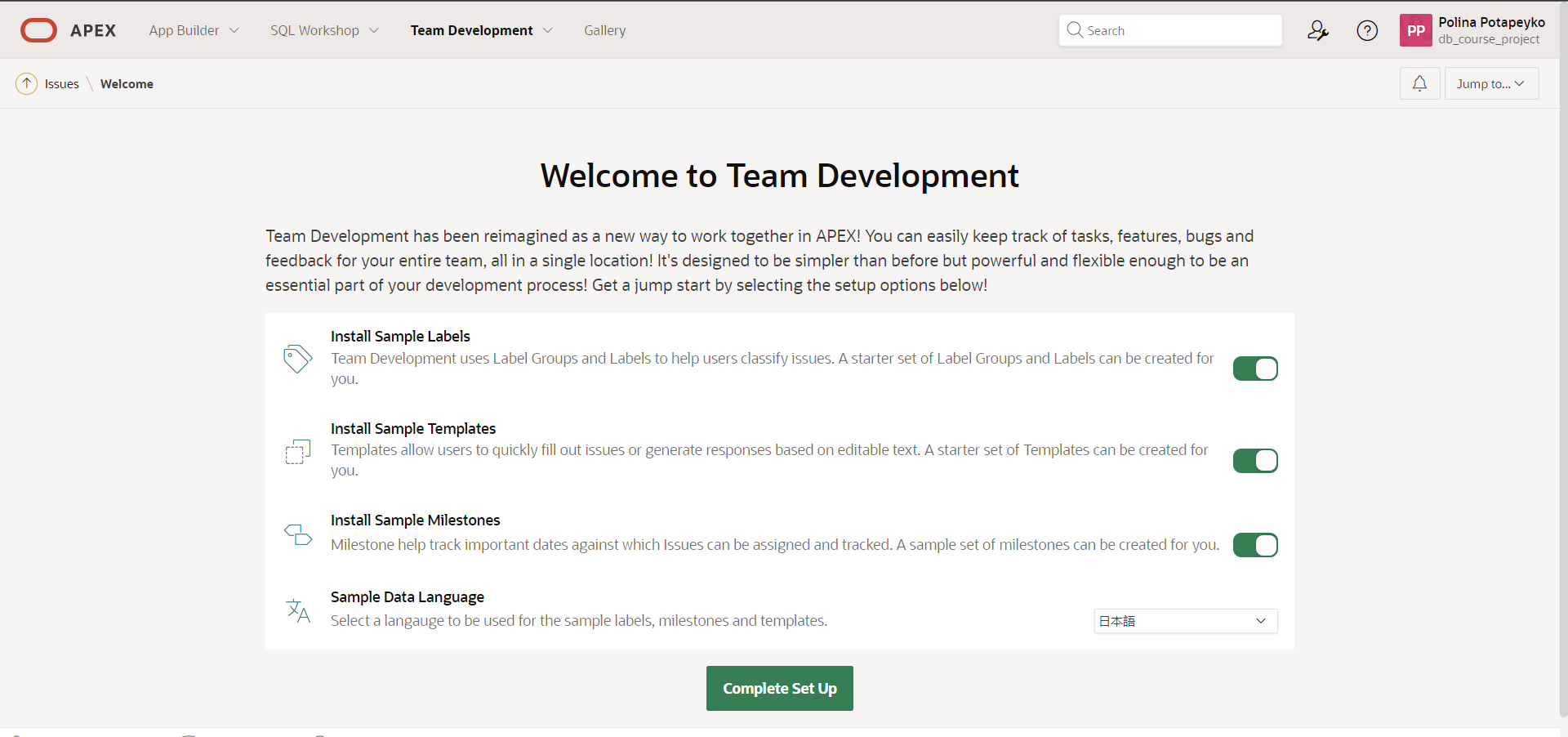


Рисунок 6.4 – Страница Team Development

APEX полезен, когда необходимо создать БД, произвольной архитектуры и назначения, для получения доступа к большим массивам данных и безопасного их обновления, а также при работе с большим числом декларативных элементов управления формами, такими как различные текстовые редакторы, чек-боксы, формы ввода данных.

APEX также является эффективным инструментом для разработки аналитических программ, обрабатывающих информацию, хранящуюся в БД. Отчеты связаны между собой ссылками, что позволяет пользователю легко ориентироваться в содержимом БД и переходить от одного отчета к другому так же легко, как и между страницами сайта. Помимо этого, простота, наглядность и информативность достигаются также благодаря расширенной функциональности APEX при выводе результатов SQL запросов в графическом виде.

В APEX применяется декларативное программирование, при котором код не генерируется, не компилируется, а пользователь взаимодействует с мастерами и списками свойств. Очень важным является знание языка SQL, так, как именно этот язык используется для определения отчетов и диаграмм. Применение декларативного кода устраняет разногласия между разработчиками и делает программы, созданные на APEX, легко управляемыми и сопровождаемыми.

Первым преимуществом Oracle APEX является разработка с минимальным программированием. Управляемый подход с функциональным, интуитивно понятным графическим интерфейсом на основе браузера ускоряет создание приложений. Мощные компоненты пользовательского интерфейса позволяют добавлять в приложения обширные функциональные возможности с минимальным программированием. На рисунке 6.5 показаны шаблоны самых популярных видов пользовательских страниц для визуализации данных.

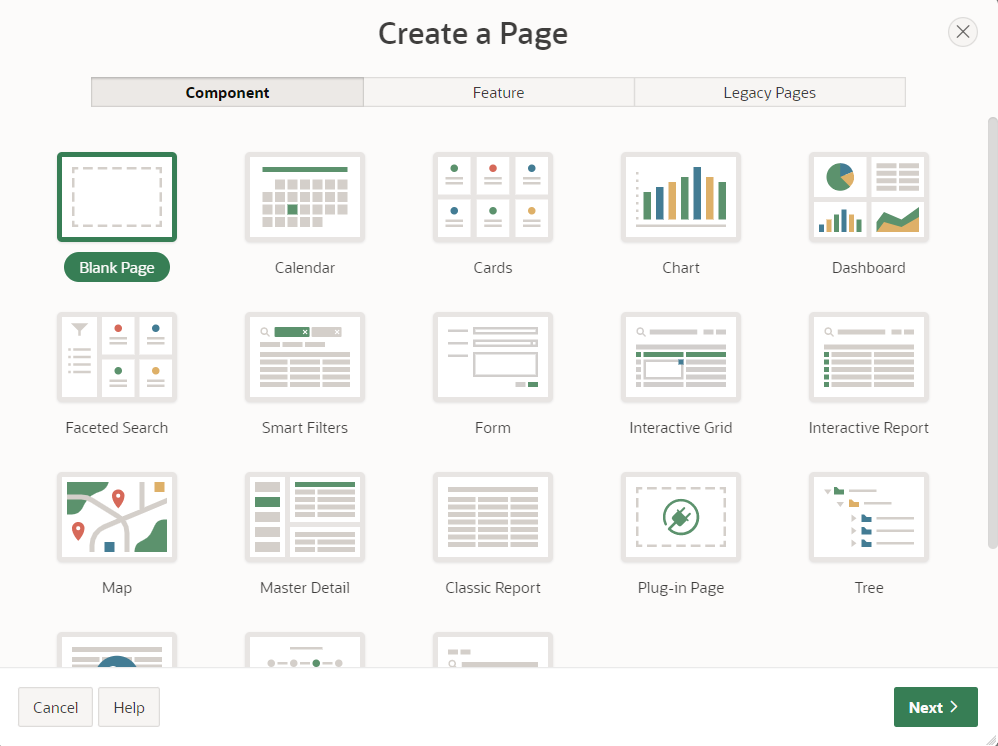


Рисунок 6.5 – Мастер шаблонов страниц

Следующим преимуществом является сверхбыстрое реагирование. Приложения, созданные с помощью APEX, отличаются высокой степенью гибкости. Такие компоненты, как формы, диаграммы и отчеты, четко работают на экранах с разным разрешением, сохраняя одинаковые функции без написания дополнительного кода. На рисунке 6.6 представлен мастер создания страницы.

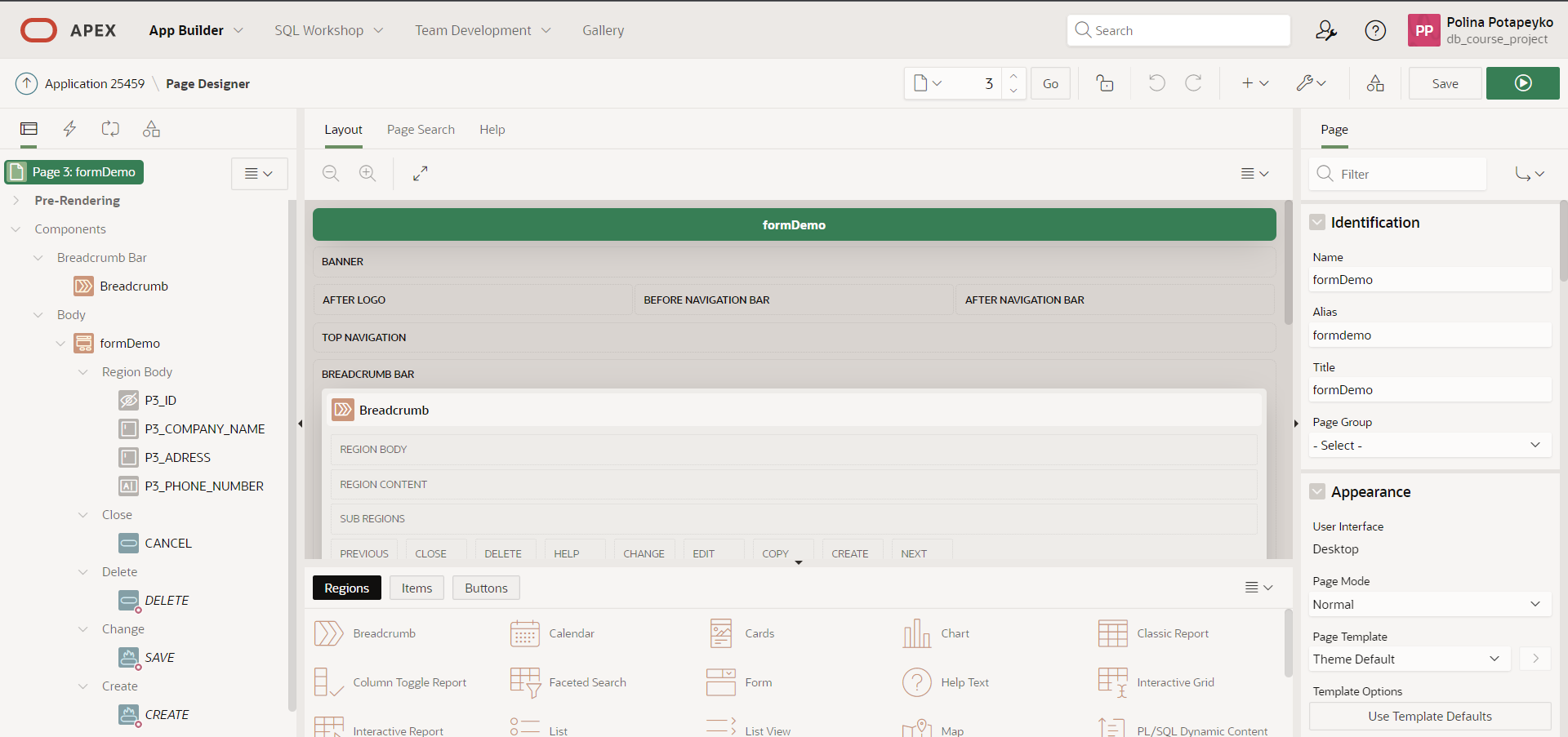


Рисунок 6.6 – Мастер создания страницы

Значимой особенностью Oracle APEX является гибкая разработка. Разработчики могут легко реализовывать изменения и совершенствовать свои процессы, часто в режиме реального времени, без необходимости в сложном многоэтапном развертывании.

Разработчики на платформе APEX также выделяют уверенное создание приложений. Количество дефектов безопасности в приложениях сводится к минимуму — от межсайтового скриптинга до внедрения SQL — с помощью встроенной в платформу системы безопасности.

Средства разработки и отслеживания кода также характерны данной технологии. Подробные диагностические данные помогут разработчикам выявлять дефекты или проблемы с производительностью в приложениях в начале цикла разработки.

Oracle APEX позволяет создавать современные приложения с конвергентной базой данных. Это означает, что можно создавать современные приложения на основе данных с помощью конвергентной базы данных, поддерживающей все современные типы данных, включая реляционные, JSON, пространственные и многие другие. Данные интегрируются из локальных и удаленных источников с использованием интерфейсов REST.

Автоматизированное управление заменяет выполняемые вручную операции, часто связанные с ошибками, автоматизацией на основе машинного обучения, для инициализации, масштабирования, защиты, резервного копирования и восстановления всего стека приложений.

Oracle APEX обеспечивает стабильную, высокую производительность благодаря уникальной архитектуре приложений в базе данных, автоматизированной настройке баз данных и возможностям индексирования, имеющимся в Autonomous Database, а также за счет специализированного оборудования Exadata.

Также данная технология повышает продуктивность разработчиков, масштабируемость и производительность приложений благодаря высокооптимизированной архитектуре. Автоматическое управление состоянием сеанса, разрешение логики приложений и управление данными непосредственно в базе данных с практически нулевым временем задержки доступа к данным.

Автоматическое масштабирование вычислительных ресурсов в соответствии с требованиями к пропускной способности без простоев по мере роста объемов данных и числа пользователей.

APEX обеспечивает высокую доступность, используя сочетание облачной инфраструктуры Oracle Gen 2, Oracle RAC и ежедневных автоматических резервных копий.

Простое управление с помощью консоли Oracle Cloud Infrastructure, общедоступных API REST и пакетов разработки ПО (SDK) позволяет пользователям выполнять наиболее распространенные задачи, такие как настройка управления и анализ использования/выставления счетов.

Фасетный поиск позволяет пользователям быстро искать и фильтровать данные. Предоставьте пользователям возможность просматривать данные новыми способами и извлекать из них ценную информацию несколькими кликами мыши.

Мощные возможности построения диаграмм и визуализации полностью поддерживают HTML5 и могут быть использованы в любом современном браузере. Oracle JET предоставляет разработчикам модульный набор инструментов для построения диаграмм на основе открытого кода.

Компонент интерактивных отчетов позволяет пользователям легко настраивать отображаемые ими данные таким образом, чтобы они удовлетворяли их потребностям, причем все это без программирования.

Новый мощный способ представления данных с помощью карт. Компонент карт обеспечивает простую настройку внешнего вида, включая значки, бейджи, носители, расширенные выражения HTML и возможность задавать несколько действий на одну карту. Пример создания карт представлен на рисунке 6.7.

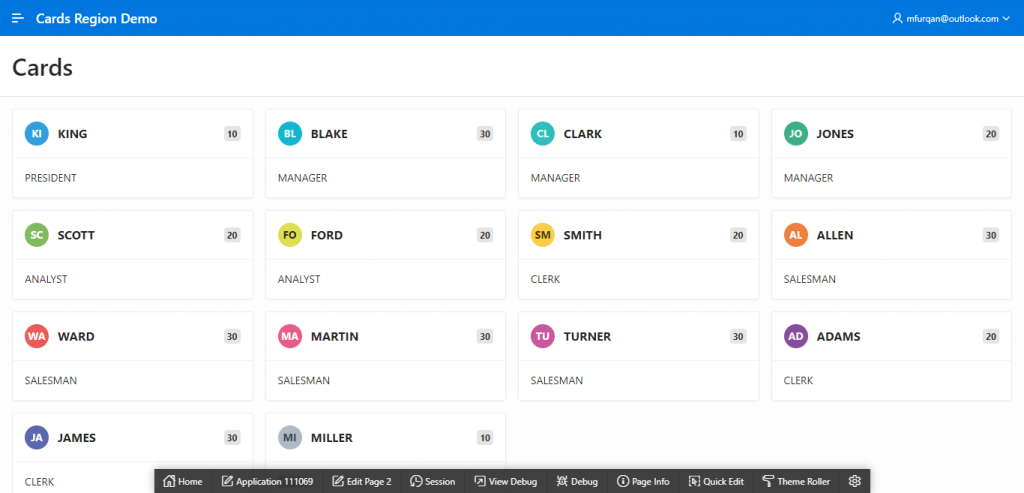


Рисунок 6.7 – Элемент карт

Преобразование электронных таблиц веб-приложение за считаные минуты. Несколькими кликами мыши можно преобразовать электронную таблицу в полнофункциональное приложение с настраиваемыми информационными панелями, диаграммами, отчетами и формами. Простой доступ и ведение данных из веб-приложения.

Обеспечение аутентификации и авторизации. Защита приложений осуществляется с помощью механизма единого входа, OAuth2, OpenID Connect и других средств. Приложения обеспечиваются безопасностью, которой нет в большинстве электронных таблиц.

Более 20 полнофункциональных и готовых к использованию производительных приложений для важных и распространенных бизнес-сценариев.

Есть возможность расширения и настройки готовых приложений в соответствии с потребностями бизнеса. Эти приложения легко адаптируются к уникальным требованиям.

Использование веб-служб. Можно получать доступ к данным из различных веб-сервисов, включая конечные точки REST приложения Oracle SaaS, и манипулировать результатами с помощью SQL.

Предоставление API-интерфейсов REST. Быстрое создание веб-сервисов, которые обращаются к объектам СУБД Oracle Database. Страница сервисов REST представлена на рисунке 6.8.

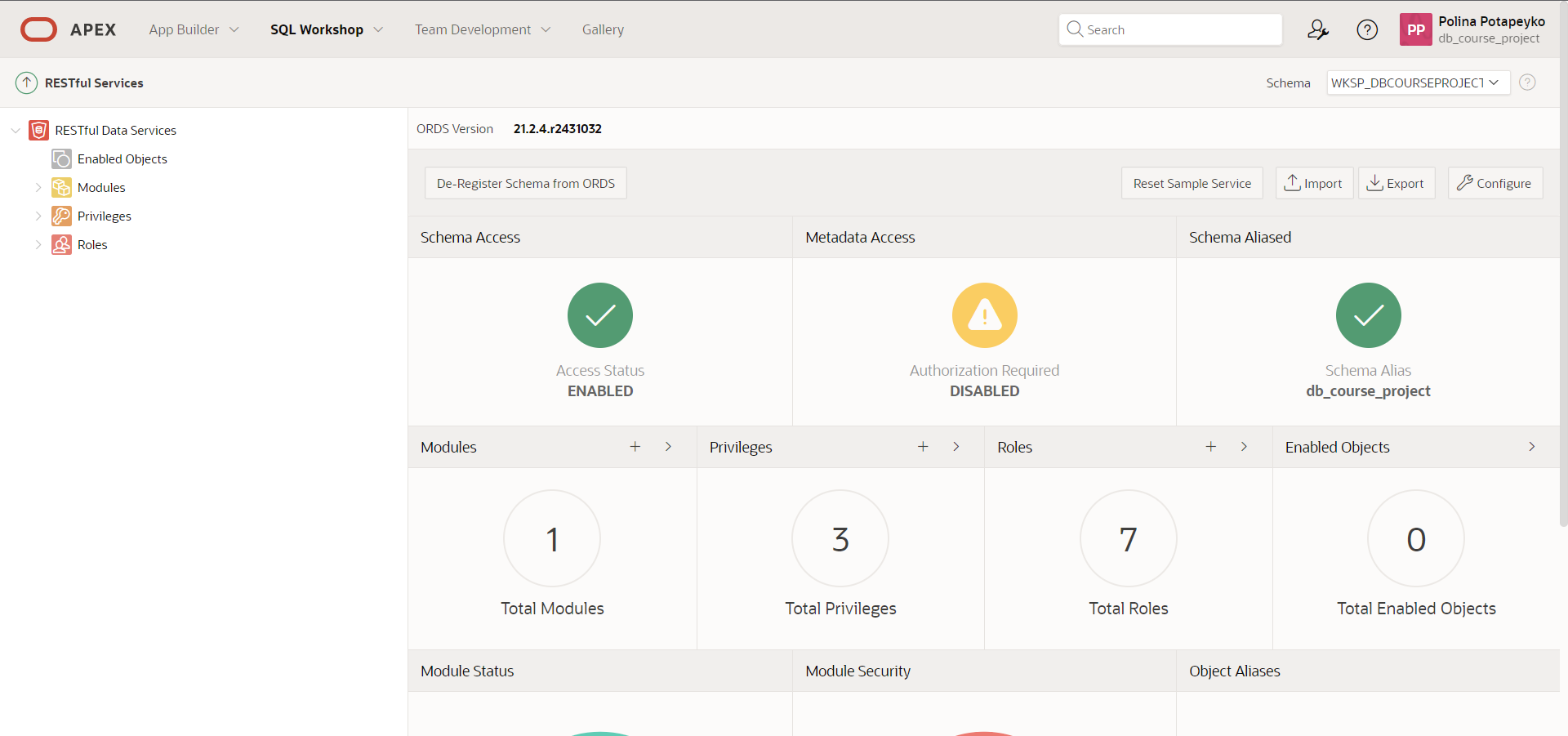


Рисунок 6.8 – Страница RESTful Services

Перевод пользовательского интерфейса работающего приложения на несколько языков без дублирования логики или повторной упаковки приложений.

Простое форматирование и отображение содержимого приложения на основе текущего местоположения конечного пользователя.

В данном курсовом проекте на базе APEX было создано приложение базы данных «Магазин бытовой техники».

Был создан ряд пользовательских интерфейсов, разделенных на клиентские (список товаров только для чтения и список заказов, представляющий собой интерфейс типа «Интерактивный отчет») и административные (интерактивные отчеты для каждой таблицы базы данных).

Пример страницы таблицы SUPPLIERS представлен на рисунке 6.9.

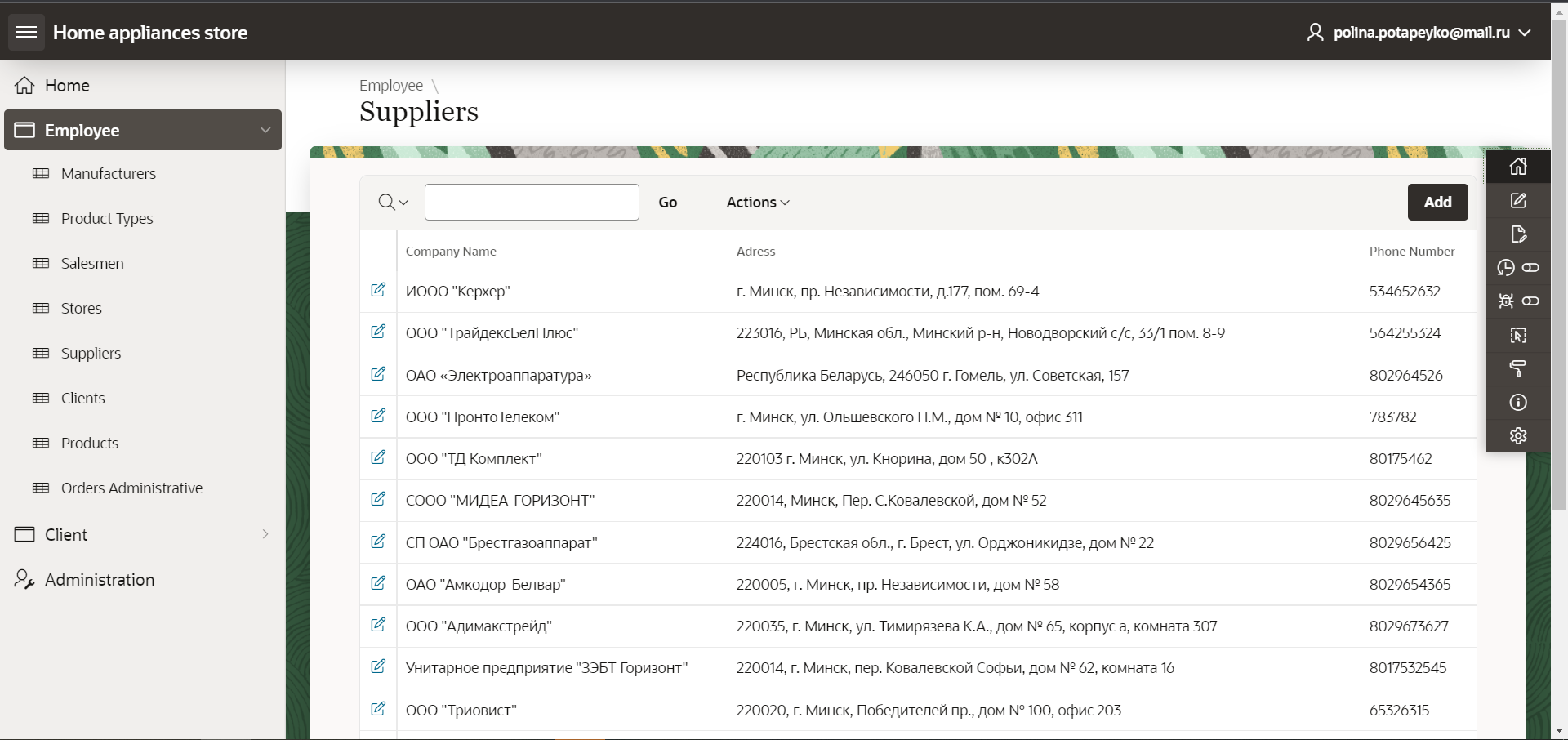


Рисунок 6.9 – Страница таблицы SUPPLIERS

Аналогично пользовательскому интерфейсу таблицы SUPPLIERS были созданы пользовательские интерфейсы остальных таблиц со стороны администратора.

1. **Руководство пользователя**

Так как приложение на основе Oracle Apex представляет собой полностью веб-приложение, то, чтобы его запустить, нужно первоначально перейти на сайт <https://apex.oracle.com/pls/apex/>, затем авторизоваться, для чего нужно ввести в поле WORKSPACE строку DB\_COURSE\_PROJECT, в поле Username ввести строку POLINA.POTAPEYKO@MAIL.RU, и в поле Password ввести строку courseProjectPassword. На рисунке 7.1 представлен результат проделанных операций.

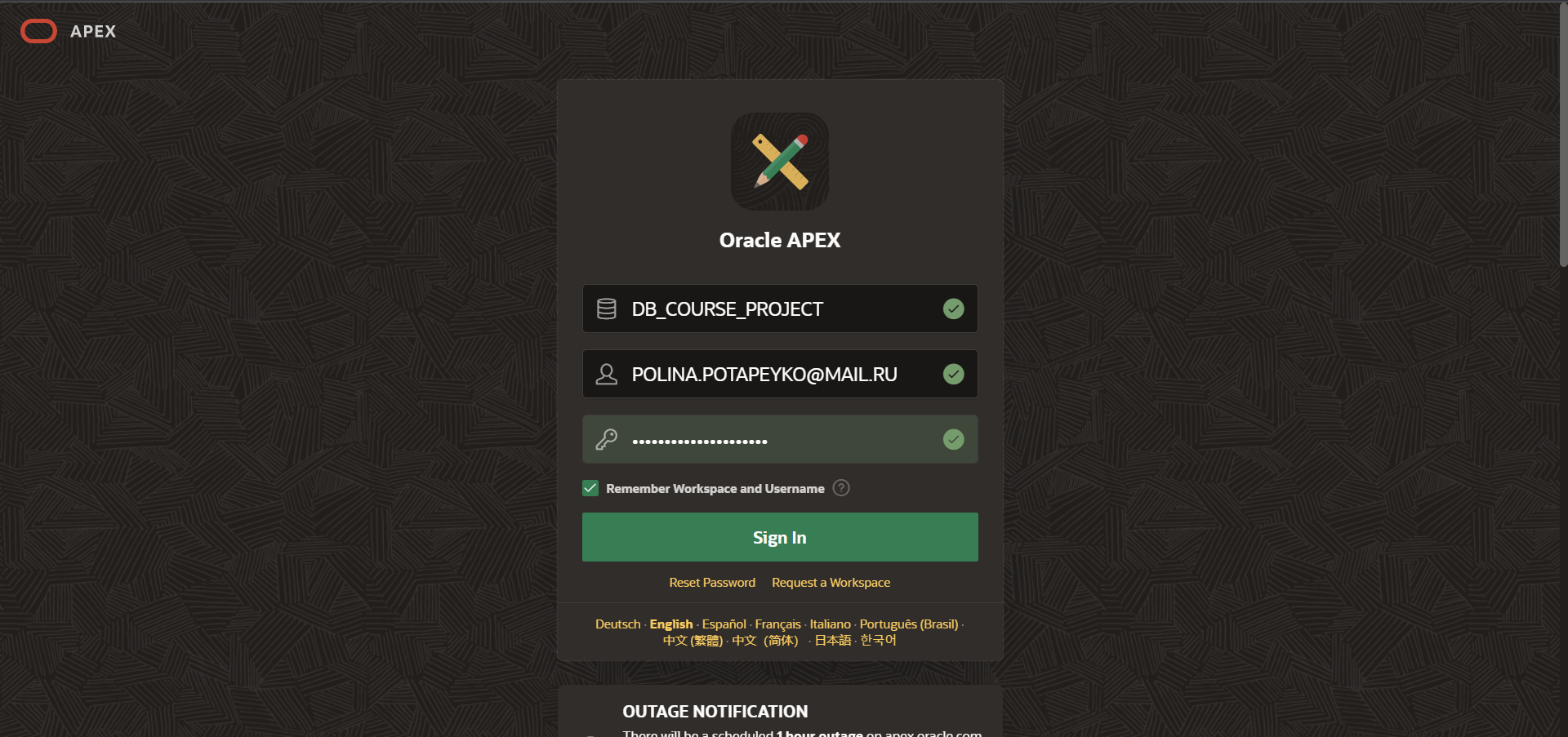


Рисунок 7.1 – Окно авторизации на сайт

После нажатия на кнопку Sign In будет произведен переход на домашнюю страницу рабочего пространства apex.oracle.com. Следующим действием нужно нажать на раздел App Builder, чтобы открылась страница конструктора проектов (рисунок 7.2).

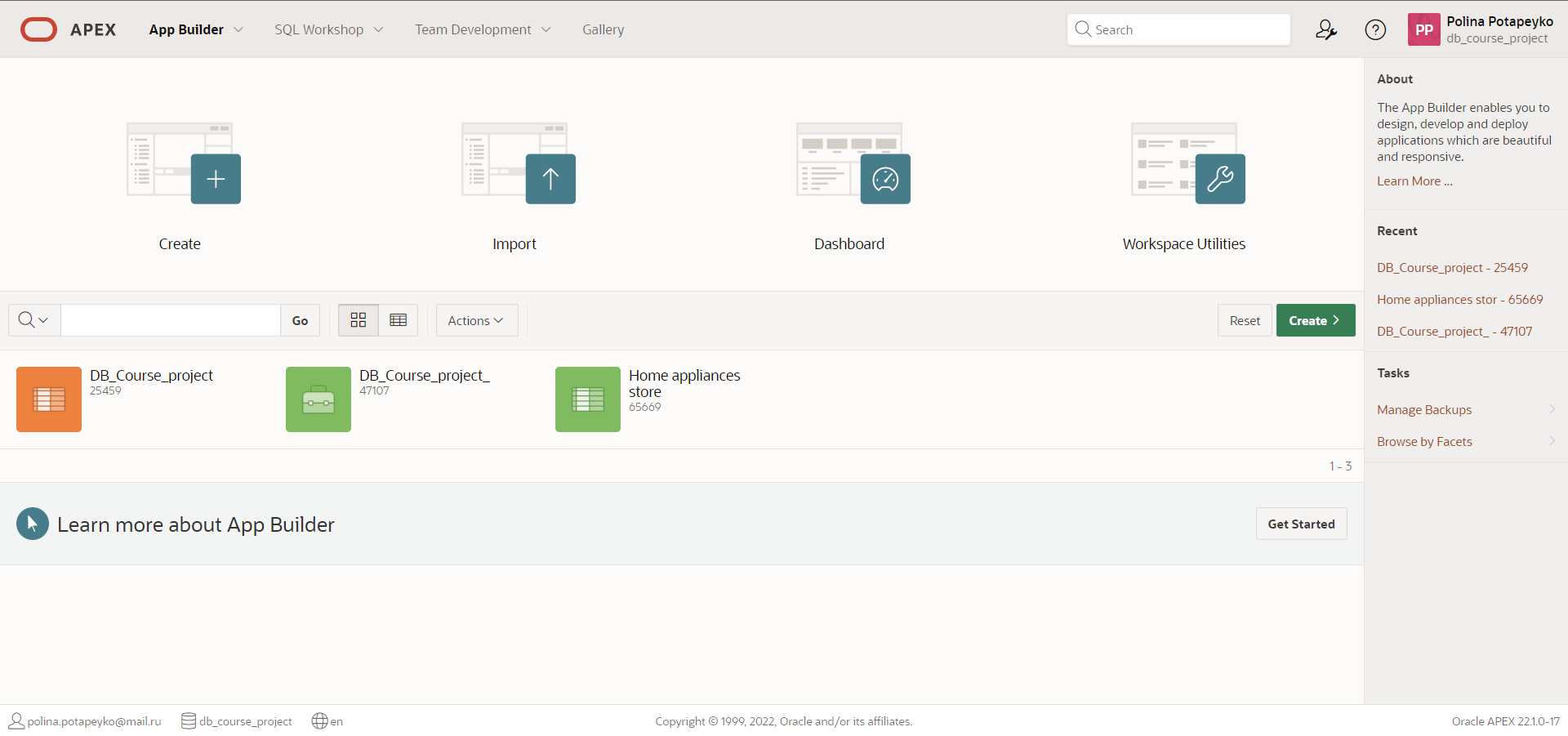


Рисунок 7.2 – Страница конструктора проектов

Затем нужно нажать на иконку проекта «Home appliances store», откроется домашняя страница проекта (рисунок 7.3).

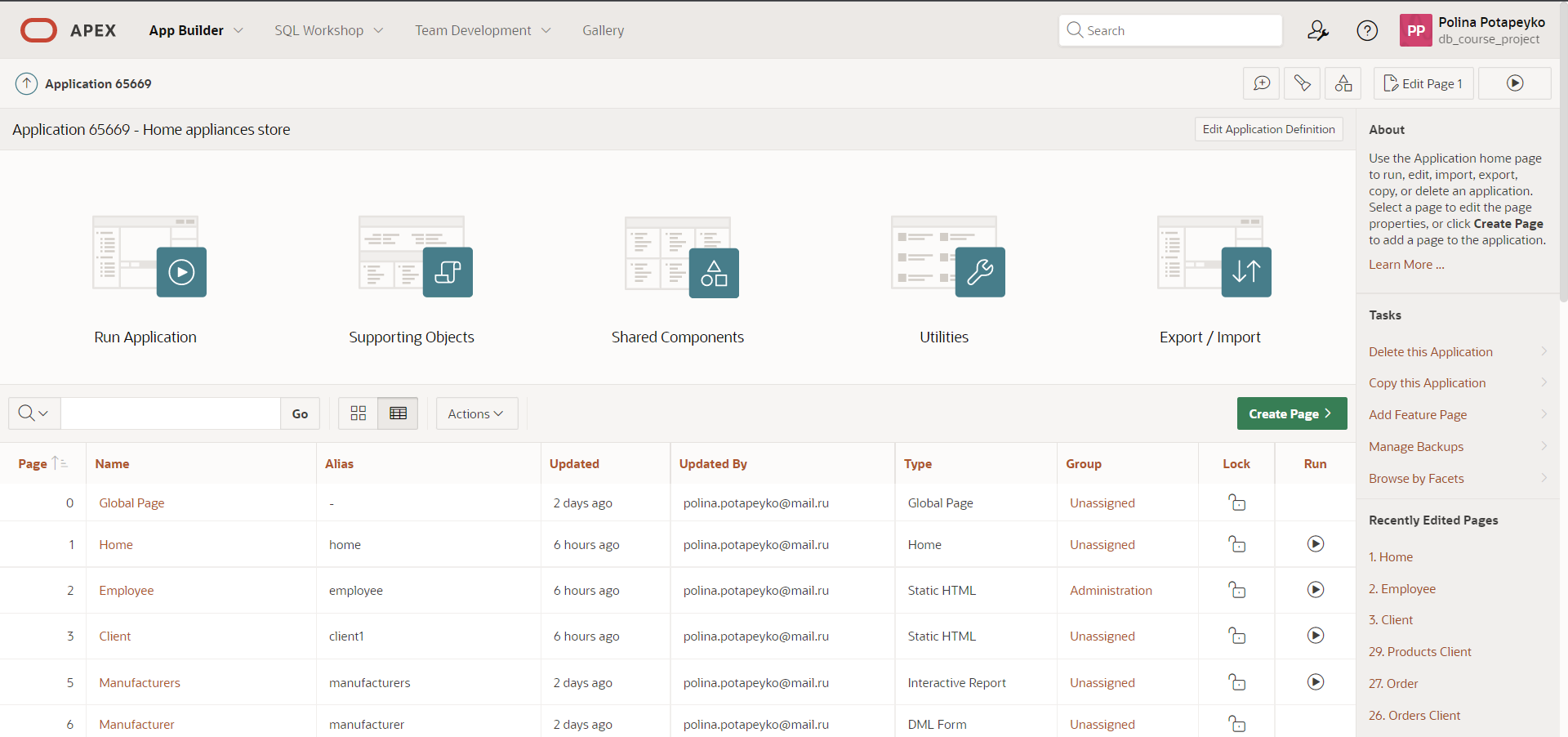


Рисунок 7.3 – Домашняя страница приложения «Home appliances store»

После этого нужно нажать кнопку Run Application, которая и запустит приложение, вызвав его в отдельной веб-странице.

Первоначально появится страница логина, в которой нужно заполнить поля Username и Password. Было создано два пользователя с различными ролями. На рисунке 7.4. представлена страница логина.

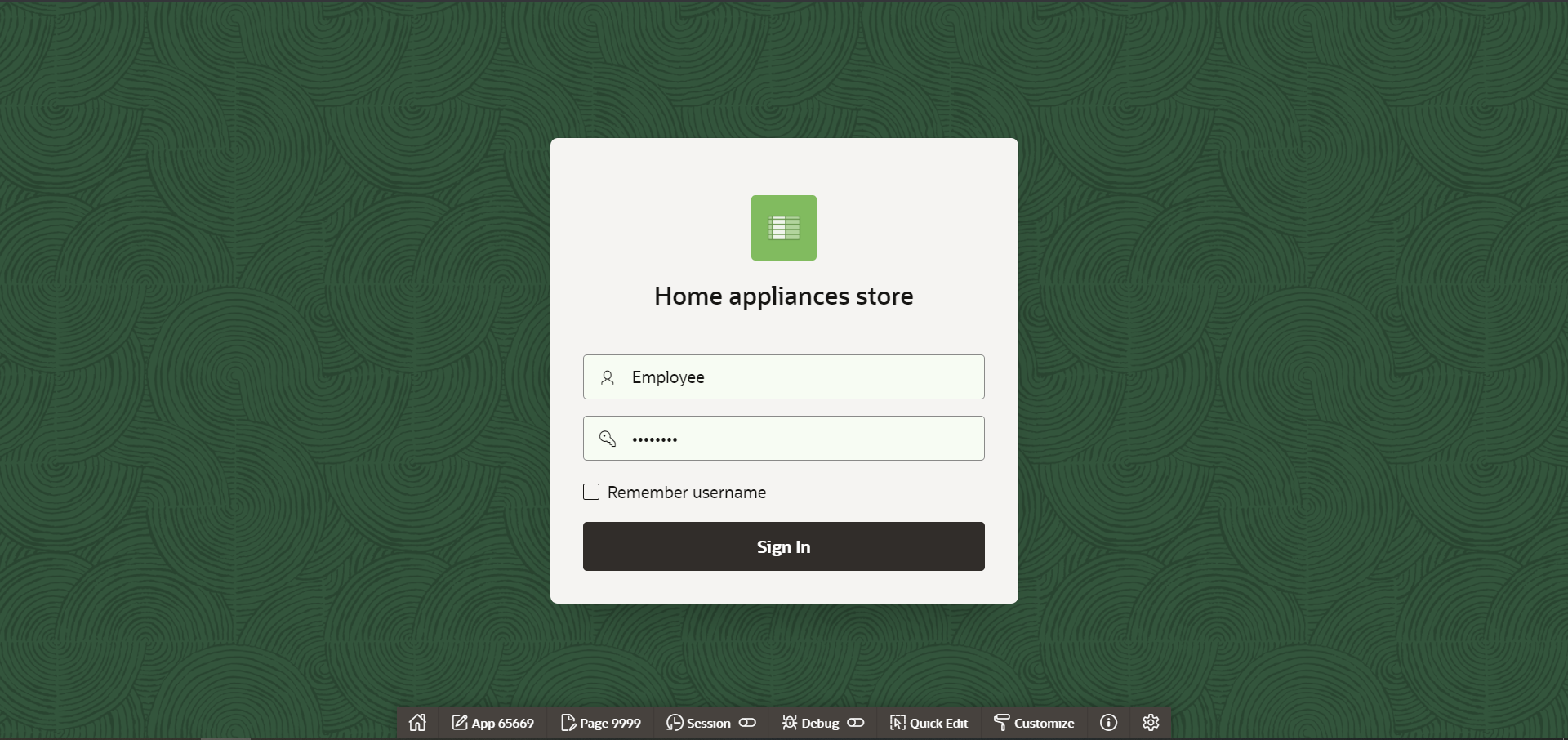


Рисунок 7.4 – Страница логина

Пользователь employee является администратором приложения и имеет доступ ко всем таблицам базы данных с возможностью их изменения. Пароль для входа pa$$w0rd.

Второй пользователь client является клиентом приложения и имеет доступ к двум таблицам: таблице Orders для полного изменения и таблице Products только для чтения с возможностью поиска товара. Пароль пользователя client также pa$$w0rd.

После успешной авторизации мы попадаем на главную страницу приложения (рисунок 7.5).

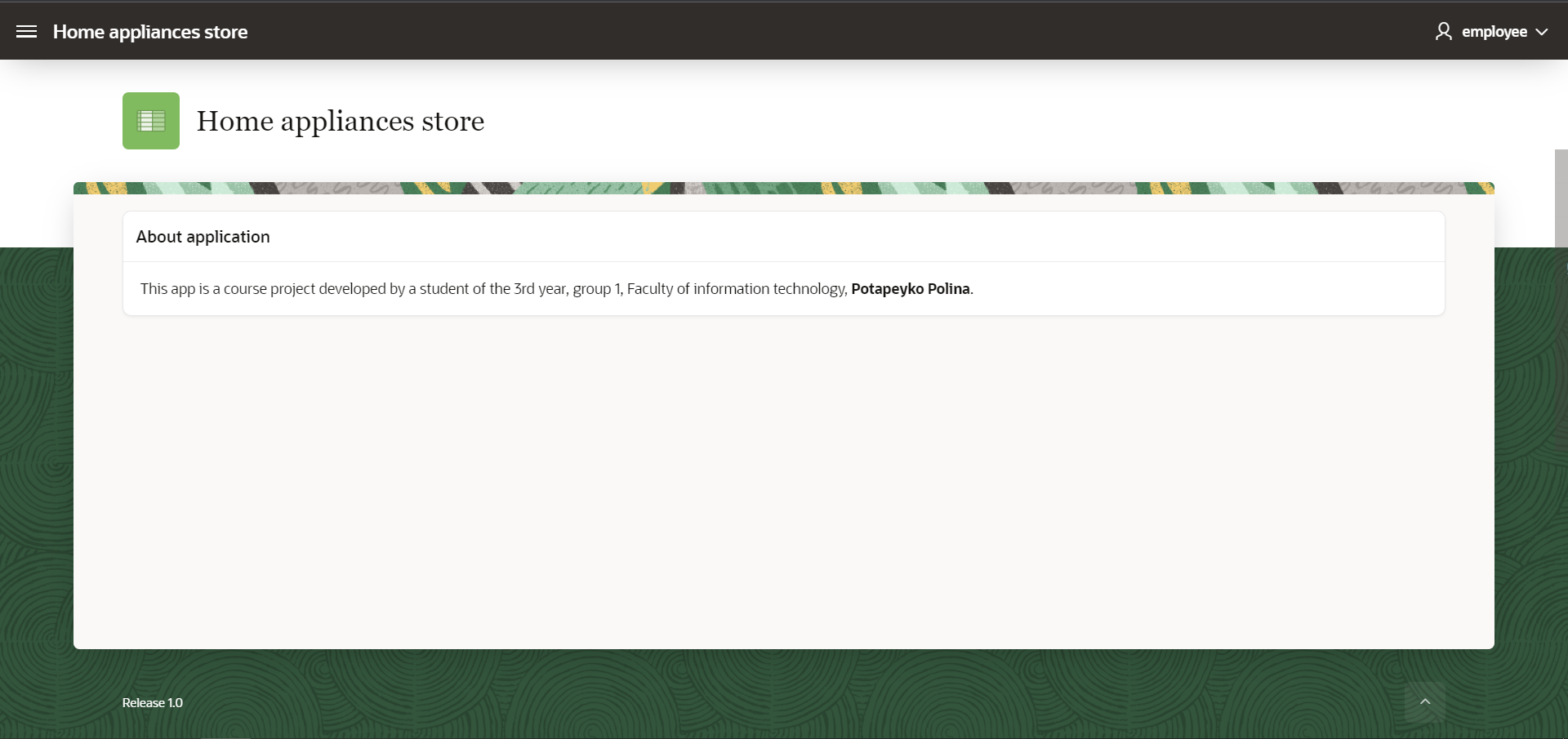


Рисунок 7.5 – Главная страница приложения «Home appliances store»

Чтобы попасть на другие страницы приложения, нужно в левом верхнем углу нажать на кнопку с изображением трех полос, откроется меню (рисунок 6.5).

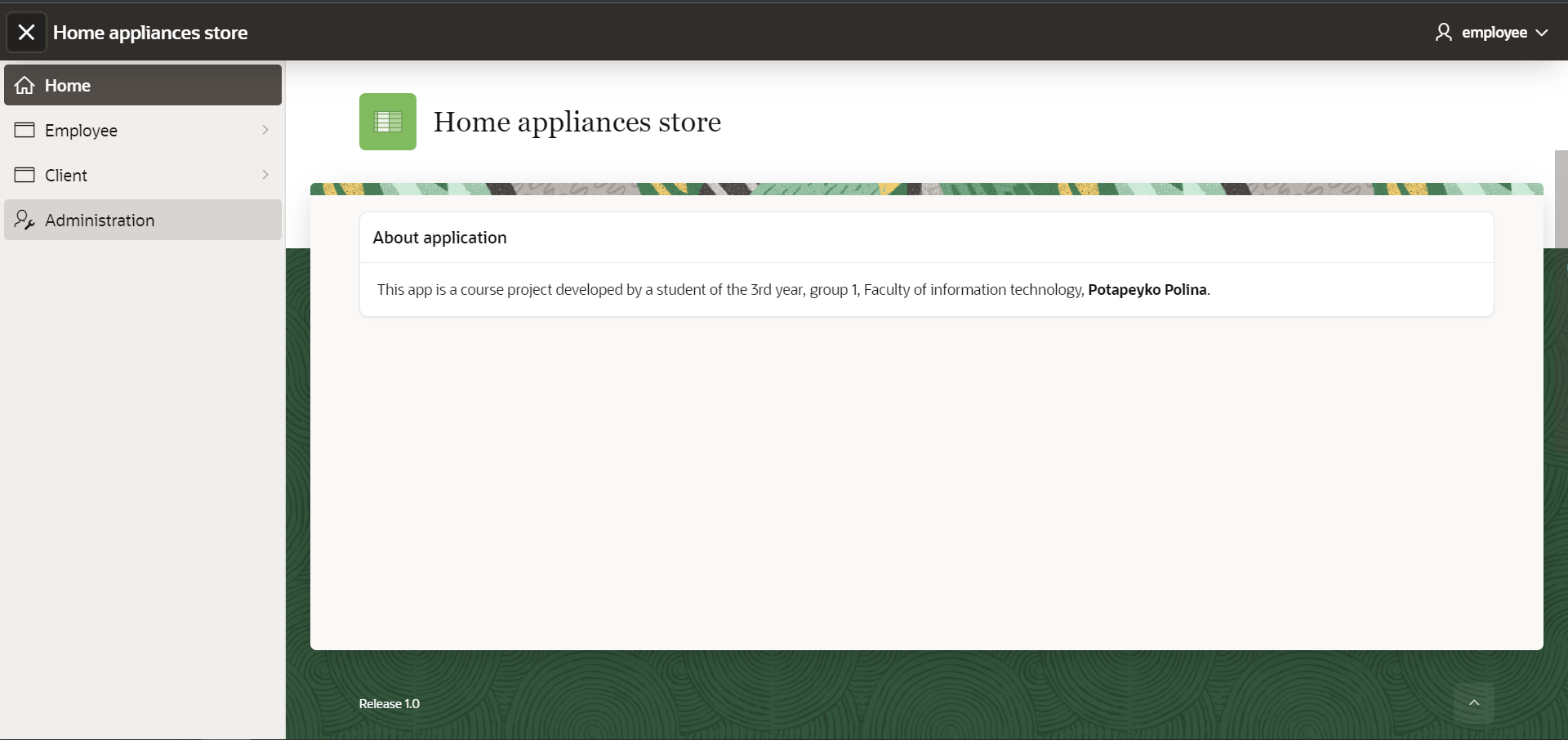


Рисунок 7.6 – Открытое меню приложения

В зависимости от роли пользователя осуществляется доступ к двум секциям: секции таблиц сотрудника или клиента. Клиент не сможет попасть в секцию Employee с таблицами сотрудника, однако сотрудник, являясь администратором, имеет доступ как к таблицам сотрудника, так и к таблицам клиента в секции Client. При клике по пункту меню Employee откроется страница Employee с описанием страницы (рисунок 7.7).

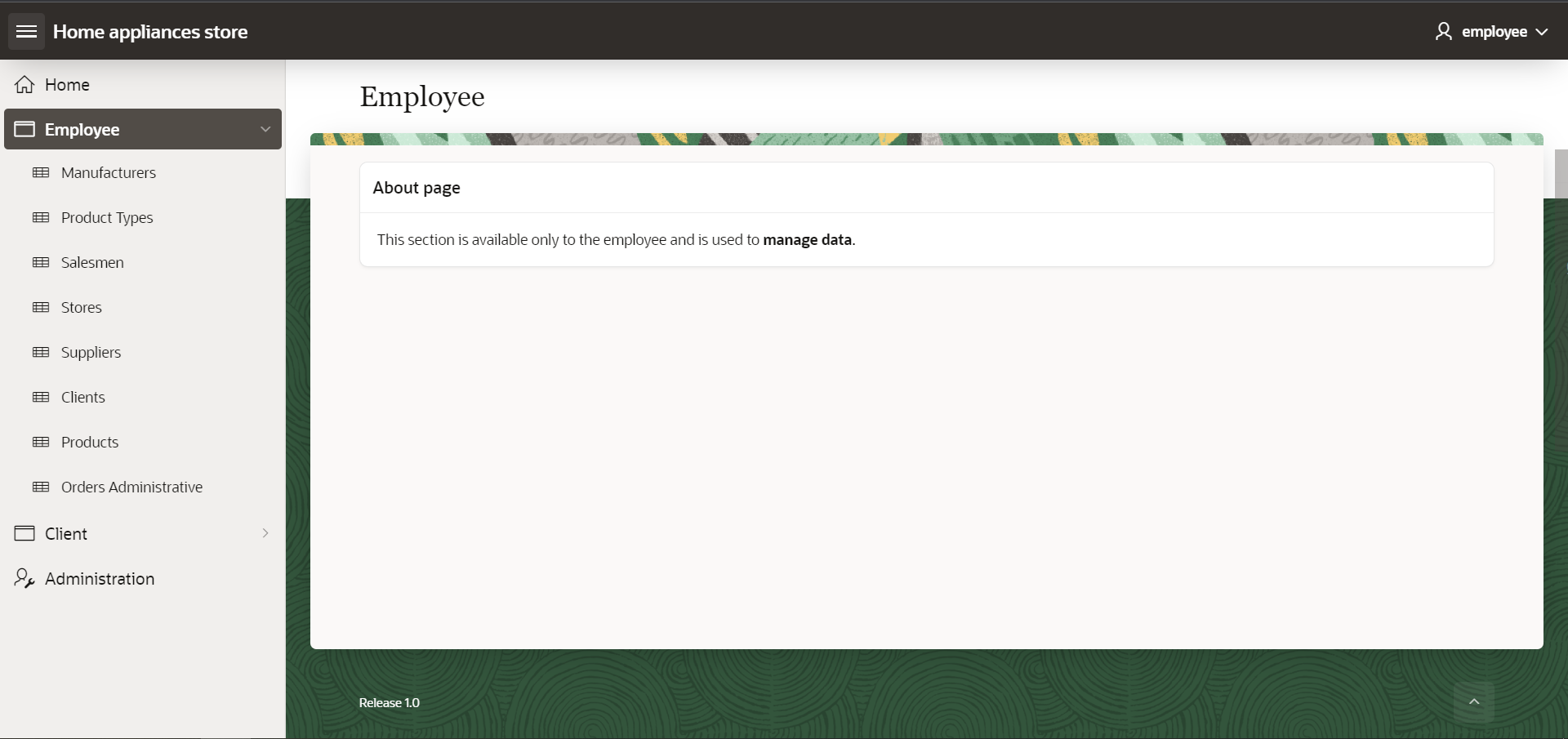


Рисунок 7.7 – Страница Employee

После этого можно выбирать, на страницу какой таблицы перейти. Перейдем, к примеру, на страницу таблицы Products.

На странице представлен список товаров с возможностью поиска товара, добавления нового товара, а также удаления и изменения существующего товара (рисунок 7.8).

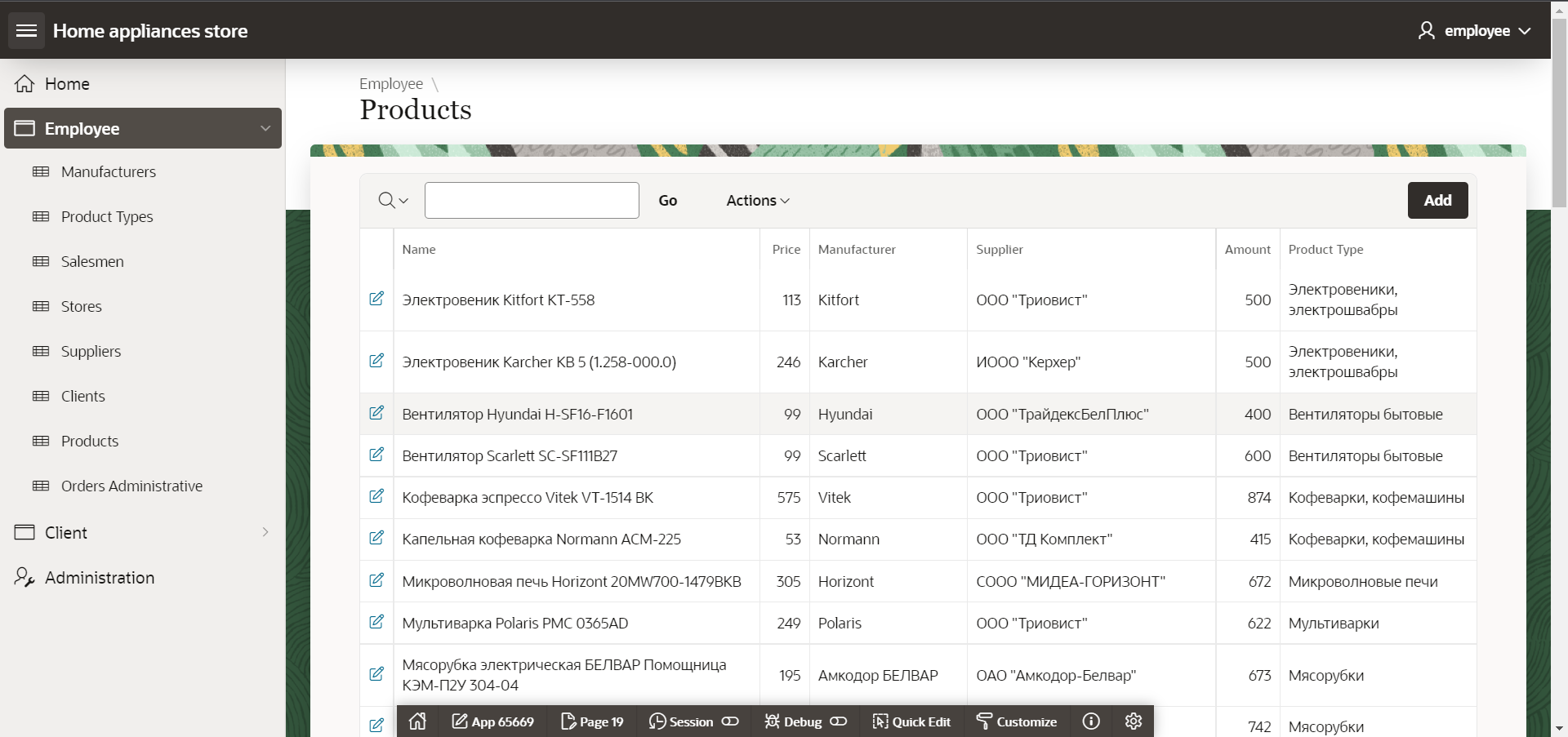


Рисунок 7.8 – Страница Products

По нажатию на кнопку с иконкой карандаша откроется форма выбранного для изменения товара (рисунок 7.9).

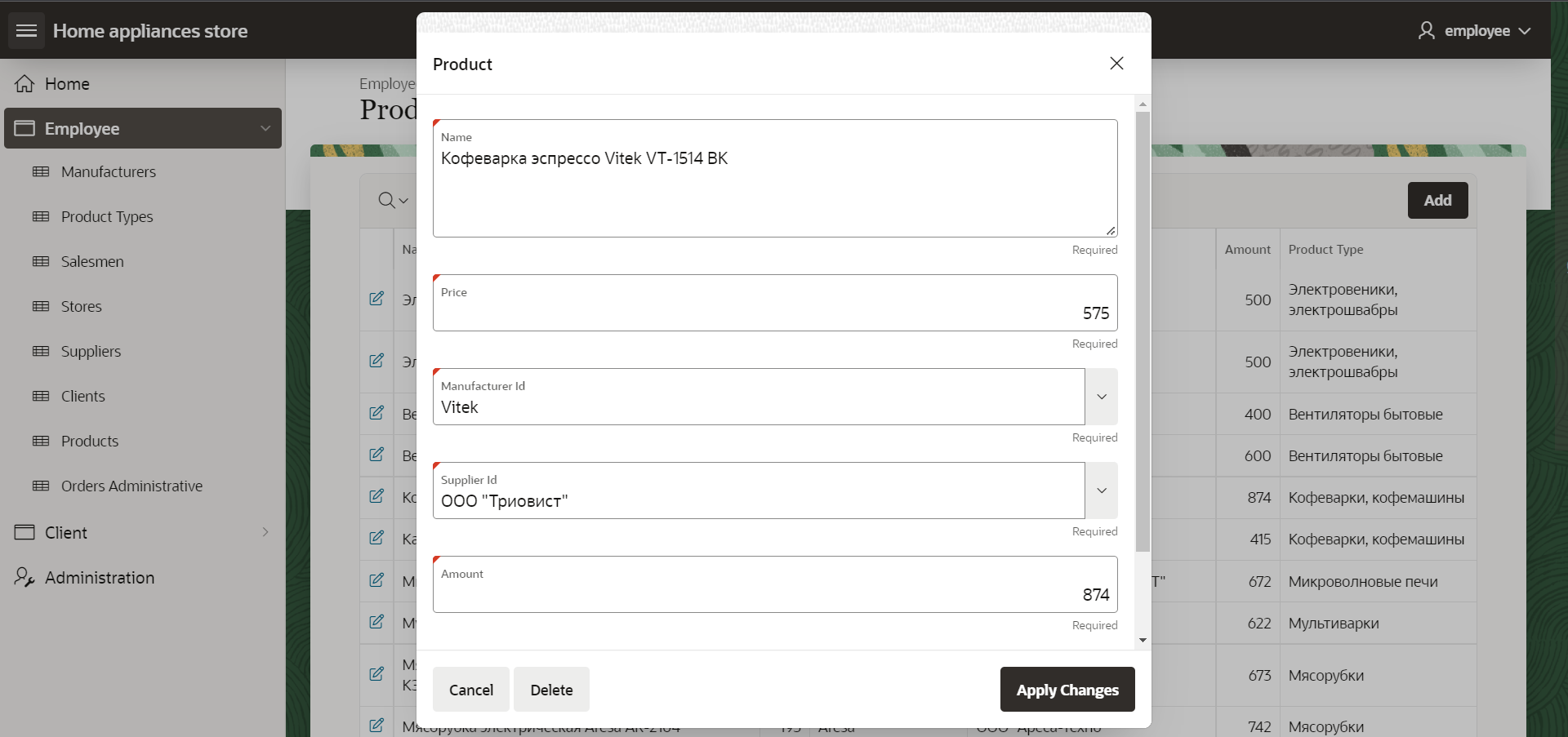


Рисунок 7.9 – Форма изменения товара

Аналогичным образом построены все остальные административные страницы, кроме страницы Orders Administrative. Она предназначена только для просмотра списка заказов, оформленных клиентом (рисунок 7.10).

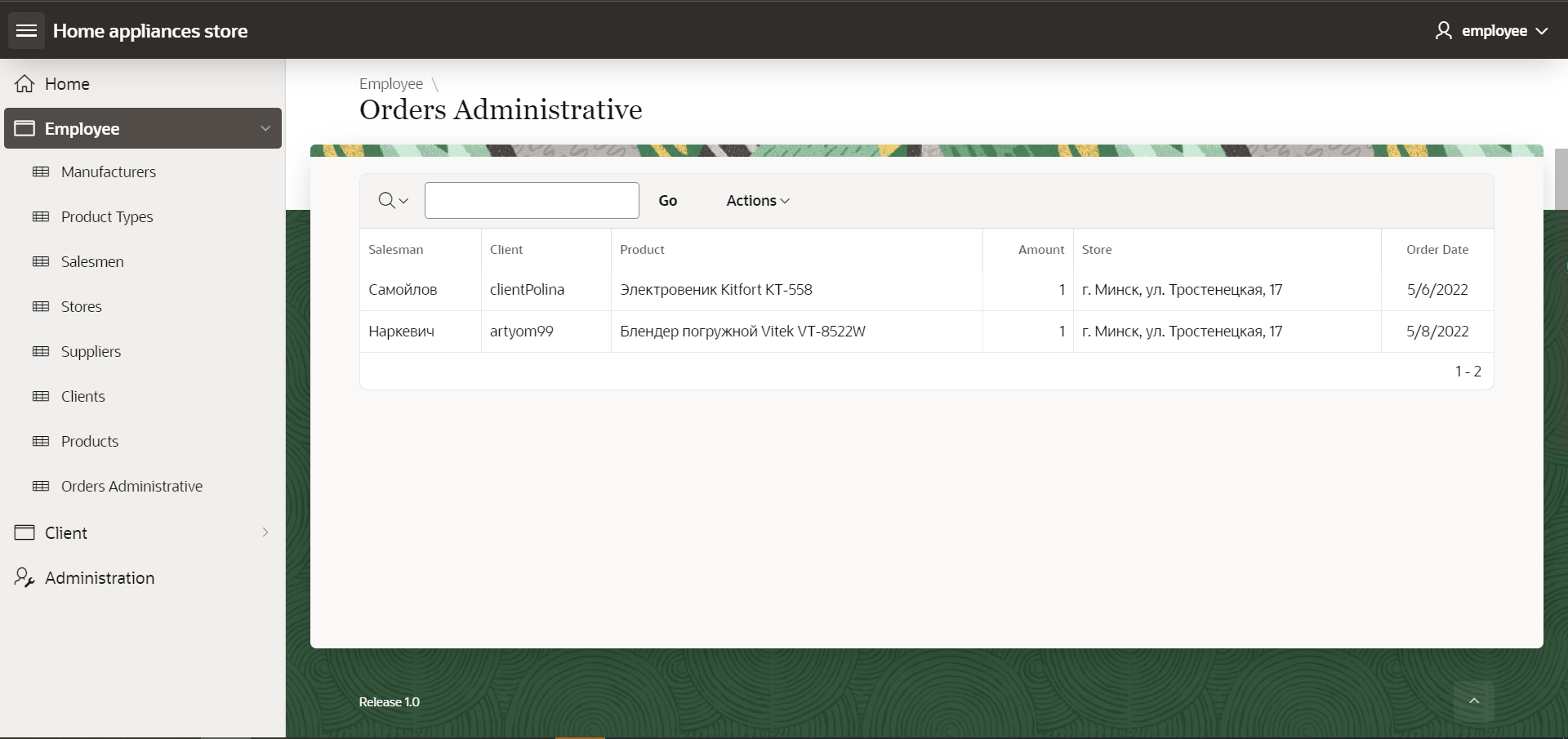


Рисунок 7.10 – Страница Orders Administrative

Переключившись на пользователя client попытаемся зайти на страницу Employee, получаем сообщение об ошибке доступа (рисунок 7.11).



Рисунок 7.11 – Ошибка доступа

Клиенту, как говорилось выше, доступно две таблицы, перейдем на страницу Client, а с нее на страницу Products Client (рисунок 7.12). на данной странице реализована возможность просмотра списка товаров, а также поиска конкретного товара.

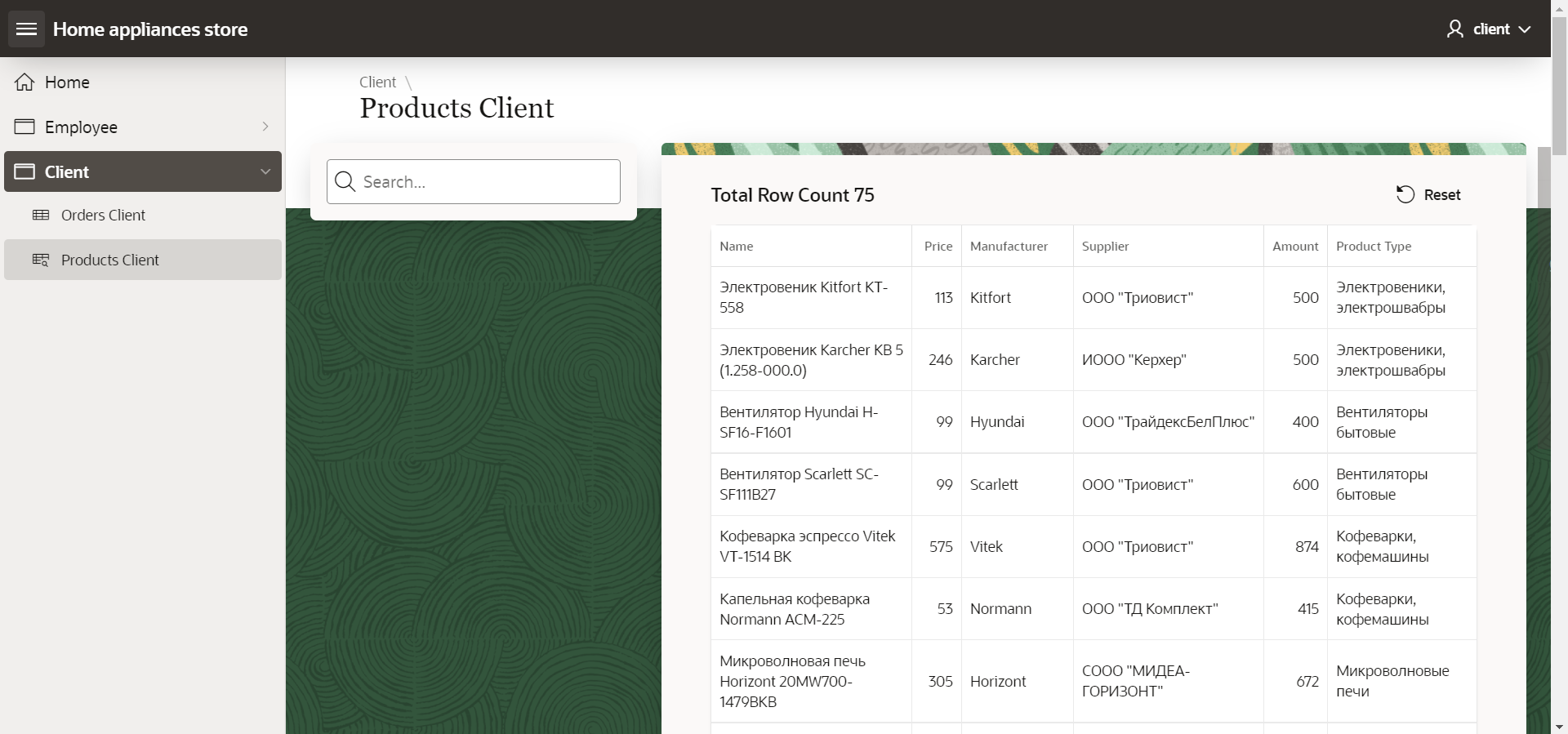


Рисунок 7.12 – Страница Products Client

На странице Orders Client есть возможность создания, изменения и удаления существующего заказа (рисунок 7.13).

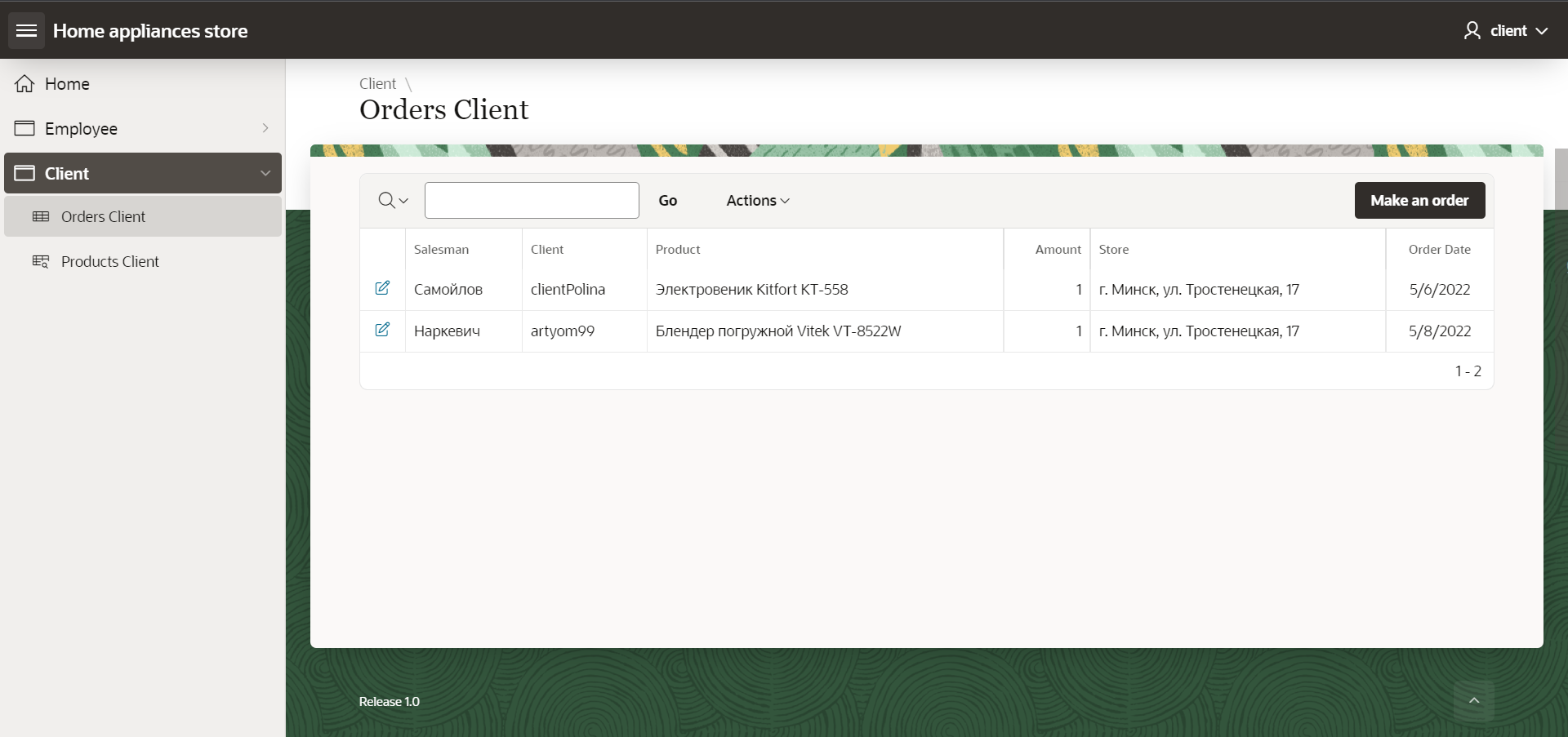


Рисунок 7.13 – Страница Orders Client

Исходя из проделанной работы можно сказать, что разработанное приложение корректно выполняет свои функции, такие как создание заказа, поиск товара, изменение данных. При этом приложение разграничивает работу клиента и администратора.

# **Заключение**

В ходе выполнения поставленной нами задачи была достигнута цель по созданию базы данных для программного средства «Магазин бытовой техники», которая в совокупности с приложением составляет полноценное веб-приложение для управления магазином. В данной работе были использованы возможности проприетарной среды разработки Oracle APEX для разработки самой базы данных и интерфейсов взаимодействия с ней. При разработке использовались объекты: таблицы, хранимые процедуры, индексы, последовательности, триггеры.

При разработке были выполнены следующие пункты:

* авторизация пользователей;
* составление интерфейса для взаимодействия с БД;
* возможность выбора и оформления заказа на товар;
* изменение данных пользователя через учётную запись администратора;
* изменение данных сущностей базы данных.

Приложение прошло тестирование при использовании в БД большого количества данных. Также были реализованы процедуры для импорта и экспорта данных в формат XML.

При разработке активно использовалась технология APEX.

В соответствии с полученным результатом работы приложения можно сделать вывод, что разработанное приложение работает верно, а требования технического задания выполнены в полном объеме.

# **Cписок источников**

1. Руководство Oracle APEX для начинающих (APEX 5.0) [Электронный ресурс] – <https://betacode.net/10345/oracle-apex-tutorial-for-beginners#a361175> – Дата доступа: 10.04.2022.
2. Документация Oracle [Электронный ресурс] / Foundation, Inc. https://docs.oracle.com/cd/E18283\_01/appdev.112/e12512/adm\_users.htm – Дата доступа: 10.04.2022.
3. Блог Oracle APEX [Электронный ресурс] – <https://blogs.oracle.com/apex/post/load-data-into-existing-tables-with-apex-191> – Дата доступа: 13.04.2022.
4. Официальный сайт Oracle APEX [Электронный ресурс] – <https://apex.oracle.com/en/learn/getting-started/> – Дата доступа: 08.04.2021.

# **Приложение А**

//Создание таблиц и последовательностей

-- 1. PRODUCT\_ID sequence and PRODUCTS table

drop sequence PRODUCT\_ID\_SEQ;

CREATE sequence PRODUCT\_ID\_SEQ;

CREATE TABLE PRODUCTS (

Id INT DEFAULT PRODUCT\_ID\_SEQ.NEXTVAL NOT NULL,

name NVARCHAR2(200) NOT NULL,

price DECIMAL(10) NOT NULL,

manufacturer\_id INT NOT NULL,

supplier\_id INT NOT NULL,

amount int NOT NULL,

product\_type int NOT NULL,

constraint PRODUCT\_PK PRIMARY KEY (Id));

-- 2. MANUFACTURER\_ID sequence and MANUFACTURERS table

drop sequence MANUFACTURER\_ID\_SEQ;

CREATE sequence MANUFACTURER\_ID\_SEQ;

CREATE TABLE MANUFACTURERS (

Id INT DEFAULT MANUFACTURER\_ID\_SEQ.NEXTVAL NOT NULL,

сompany\_name NVARCHAR2(200) NOT NULL,

adress NVARCHAR2(200) NOT NULL,

phone\_number NVARCHAR2(50) NOT NULL,

constraint MANUFACTURER\_PK PRIMARY KEY (Id));

-- 3. SUPPLIER\_ID sequence and SUPPLIERS table

drop sequence SUPPLIER\_ID\_SEQ;

CREATE sequence SUPPLIER\_ID\_SEQ;

CREATE TABLE SUPPLIERS (

Id INT DEFAULT SUPPLIER\_ID\_SEQ.NEXTVAL NOT NULL,

сompany\_name NVARCHAR2(200) NOT NULL,

adress NVARCHAR2(200) NOT NULL,

phone\_number NVARCHAR2(50) NOT NULL,

constraint SUPPLIER\_PK PRIMARY KEY (Id));

-- 4. CLIENT\_ID sequence and CLIENTS table

drop sequence CLIENT\_ID\_SEQ;

CREATE sequence CLIENT\_ID\_SEQ;

CREATE TABLE CLIENTS (

Id INT DEFAULT CLIENT\_ID\_SEQ.NEXTVAL NOT NULL,

nickname NVARCHAR2(200) NOT NULL,

name NVARCHAR2(50) NOT NULL,

surname NVARCHAR2(50) NOT NULL,

patronymic NVARCHAR2(50) NOT NULL,

email NVARCHAR2(200),

adress NVARCHAR2(100) NOT NULL,

phone\_number NVARCHAR2(100) UNIQUE NOT NULL,

constraint CLIENT\_PK PRIMARY KEY (id));

-- 5. SALESMAN\_ID sequence and SALESMEN table

drop sequence SALESMAN\_ID\_SEQ;

CREATE sequence SALESMAN\_ID\_SEQ;

CREATE TABLE SALESMEN (

Id INT DEFAULT SALESMAN\_ID\_SEQ.NEXTVAL NOT NULL,

name NVARCHAR2(50) NOT NULL,

surname NVARCHAR2(50) NOT NULL,

patronymic NVARCHAR2(50) NOT NULL,

adress NVARCHAR2(100) NOT NULL,

phone\_number NVARCHAR2(50) UNIQUE NOT NULL,

constraint SALESMEN\_PK PRIMARY KEY (id));

-- 6. PRODUCT\_TYPE\_ID sequence and PRODUCT\_TYPES table

drop sequence PRODUCT\_TYPE\_ID\_SEQ;

CREATE sequence PRODUCT\_TYPE\_ID\_SEQ;

CREATE TABLE PRODUCT\_TYPES (

Id INT DEFAULT PRODUCT\_TYPE\_ID\_SEQ.NEXTVAL NOT NULL,

type NVARCHAR2(200) NOT NULL,

constraint PRODUCT\_TYPES\_PK PRIMARY KEY (id));

-- 7. STORE\_ID sequence and STORES table

drop sequence STORE\_ID\_SEQ;

CREATE sequence STORE\_ID\_SEQ;

CREATE TABLE STORES (

Id INT DEFAULT STORE\_ID\_SEQ.NEXTVAL NOT NULL,

address NVARCHAR2(200) NOT NULL,

constraint STORES\_PK PRIMARY KEY (id));

-- 8. ORDER\_NUMBER sequence and ORDERS table

drop sequence ORDER\_NUMBER\_SEQ;

CREATE sequence ORDER\_NUMBER\_SEQ;

CREATE TABLE ORDERS (

order\_number INT DEFAULT ORDER\_NUMBER\_SEQ.NEXTVAL NOT NULL,

salesman\_id int NOT NULL,

client\_id int NOT NULL,

product\_id int NOT NULL,

amount int NOT NULL,

store\_id int NOT NULL,

order\_date date NOT NULL,

constraint ORDERS\_PK PRIMARY KEY (order\_number));

ALTER TABLE PRODUCTS ADD CONSTRAINT PRODUCTS\_FK1 FOREIGN KEY (manufacturer\_id)

REFERENCES MANUFACTURERS(Id);

ALTER TABLE PRODUCTS ADD CONSTRAINT PRODUCTS\_FK2 FOREIGN KEY (supplier\_id)

REFERENCES SUPPLIERS(Id);

ALTER TABLE PRODUCTS ADD CONSTRAINT PRODUCTS\_FK3 FOREIGN KEY (product\_type)

REFERENCES PRODUCT\_TYPES(Id);

ALTER TABLE ORDERS ADD CONSTRAINT ORDERS\_FK1 FOREIGN KEY (salesman\_id)

REFERENCES SALESMEN(Id);

ALTER TABLE ORDERS ADD CONSTRAINT ORDERS\_FK2 FOREIGN KEY (client\_id)

REFERENCES CLIENTS(Id);

ALTER TABLE ORDERS ADD CONSTRAINT ORDERS\_FK3 FOREIGN KEY (store\_id)

REFERENCES STORES(Id);

ALTER TABLE ORDERS ADD CONSTRAINT ORDERS\_FK4 FOREIGN KEY (product\_id)

REFERENCES PRODUCTS(Id);

# **Приложение Б**

--XML import and export

--Export Suppliers to Xml

create or replace directory XMLDATAREPOSITORY AS 'C:/app';

create or replace procedure ExportSupplierstoXml

is

DOC DBMS\_XMLDOM.DOMDocument;

XDATA XMLTYPE;

CURSOR XMLCUR IS

SELECT XMLELEMENT("SUPPLIERS",

XMLAttributes('http://www.w3.org/2001/XMLSchema' AS "xmlns:xsi",

'http://www.oracle.com/Users.xsd' AS "xsi:nonamespaceSchemaLocation"),

XMLAGG(XMLELEMENT("SUPPLIERS",

xmlelement("id", SUPPLIERS.id),

xmlelement("сompany\_name", SUPPLIERS.сompany\_name),

xmlelement("adress", SUPPLIERS.adress),

xmlelement("phone\_number", SUPPLIERS.phone\_number)

))) from SUPPLIERS;

begin

open xmlcur;

loop

fetch xmlcur into xdata;

exit when xmlcur%notfound;

end loop;

close xmlcur;

DOC := DBMS\_XMLDOM.NewDOMDocument(XDATA);

DBMS\_XMLDOM.WRITETOFILE(DOC, 'XMLDATAREPOSITORY/Suppliers.xml');

END;

--26. Import Xml data to Suppliers

create or replace procedure ImportSuppliersDataFromXml

IS

L\_CLOB CLOB;

L\_BFILE BFILE := BFILENAME('XMLDATAREPOSITORY', 'Suppliers.xml');

L\_DEST\_OFFSET INTEGER := 1;

L\_SRC\_OFFSET INTEGER := 1;

L\_BFILE\_CSID NUMBER := 0;

L\_LANG\_CONTEXT INTEGER := 0;

L\_WARNING INTEGER := 0;

P DBMS\_XMLPARSER.PARSER;

v\_doc dbms\_xmldom.domdocument;

v\_root\_element dbms\_xmldom.domelement;

V\_CHILD\_NODES DBMS\_XMLDOM.DOMNODELIST;

V\_CURRENT\_NODE DBMS\_XMLDOM.DOMNODE;

et SUPPLIERS%rowtype;

begin

DBMS\_LOB.CREATETEMPORARY (L\_CLOB, TRUE);

DBMS\_LOB.FILEOPEN(L\_BFILE, DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADCLOBFROMFILE (DEST\_LOB => L\_CLOB, SRC\_BFILE => L\_BFILE, AMOUNT => DBMS\_LOB.LOBMAXSIZE,

DEST\_OFFSET => L\_DEST\_OFFSET, SRC\_OFFSET => L\_SRC\_OFFSET, BFILE\_CSID => L\_BFILE\_CSID,

LANG\_CONTEXT => L\_LANG\_CONTEXT, WARNING => L\_WARNING);

DBMS\_LOB.FILECLOSE(L\_BFILE);

COMMIT;

P := Dbms\_Xmlparser.Newparser;

DBMS\_XMLPARSER.PARSECLOB(P,L\_CLOB);

V\_DOC := DBMS\_XMLPARSER.GETDOCUMENT(P);

V\_ROOT\_ELEMENT := DBMS\_XMLDOM.Getdocumentelement(v\_Doc);

V\_CHILD\_NODES := DBMS\_XMLDOM.GETCHILDRENBYTAGNAME(V\_ROOT\_ELEMENT,'\*');

FOR i IN 0 .. DBMS\_XMLDOM.GETLENGTH(V\_CHILD\_NODES) - 1

LOOP

V\_CURRENT\_NODE := DBMS\_XMLDOM.ITEM(V\_CHILD\_NODES,i);

DBMS\_XSLPROCESSOR.VALUEOF(V\_CURRENT\_NODE,'id/text()',et.id);

Dbms\_Xslprocessor.Valueof(V\_Current\_Node,'сompany\_name/text()',et.сompany\_name);

Dbms\_Xslprocessor.Valueof(V\_Current\_Node,'adress/text()',et.adress);

dbms\_xslprocessor.valueof(v\_current\_node,'phone\_number/text()',et.phone\_number);

insert into SUPPLIERS(сompany\_name, adress, phone\_number)

values(et.сompany\_name, et.adress, et.phone\_number);

end loop;

DBMS\_LOB.FREETEMPORARY(L\_CLOB);

DBMS\_XMLPARSER.FREEPARSER(P);

DBMS\_XMLDOM.FREEDOCUMENT(V\_DOC);

commit;

END;

# **Приложение В**

--Процедуры

--1. AddProduct

create or replace procedure AddProduct

(p\_name products.name%type,

p\_price products.price%type,

p\_manufacturer\_id products.manufacturer\_id%type,

p\_supplier\_id products.supplier\_id%type,

p\_amount products.amount%type,

p\_product\_type products.product\_type%type)

is

begin

insert into PRODUCTS(name, price, manufacturer\_id, supplier\_id, amount, product\_type)

VALUES(p\_name, p\_price, p\_manufacturer\_id, p\_supplier\_id, p\_amount, p\_product\_type);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--2. DeleteProduct

create or replace procedure DeleteProduct(p\_id products.id%type)

is

begin

delete from products where products.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--3. UpdateProduct

create or replace procedure UpdateProduct

(p\_id products.id%type,

p\_name products.name%type,

p\_price products.price%type,

p\_manufacturer\_id products.manufacturer\_id%type,

p\_supplier\_id products.supplier\_id%type,

p\_amount products.amount%type,

p\_product\_type products.product\_type%type)

is

begin

update products set id = p\_id,

name = p\_name,

price = p\_price,

manufacturer\_id = p\_manufacturer\_id,

supplier\_id = p\_supplier\_id,

amount = p\_amount,

product\_type = p\_product\_type

where products.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--4. AddProductType

create or replace procedure AddProductType(p\_type product\_types.type%type)

is

begin

insert into PRODUCT\_TYPES(type)

VALUES(p\_type);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--5. DeleteProductType

create or replace procedure DeleteProductType(p\_id product\_types.id%type)

is

begin

delete from product\_types where product\_types.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--6. UpdateProductType

create or replace procedure UpdateProductType

(p\_id product\_types.id%type,

p\_type product\_types.type%type)

is

begin

update product\_types set id = p\_id,

type = p\_type

where product\_types.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--7. AddSupplier

create or replace procedure AddSupplier

(p\_company\_name suppliers.сompany\_name%type,

p\_adress suppliers.adress%type,

p\_phone\_number suppliers.phone\_number%type)

is

begin

insert into SUPPLIERS(сompany\_name, adress, phone\_number)

VALUES(p\_company\_name, p\_adress, p\_phone\_number);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--8. DeleteSupplier

create or replace procedure DeleteSupplier(p\_id suppliers.id%type)

is

begin

delete from suppliers where suppliers.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--9. UpdateSupplier

create or replace procedure UpdateSupplier

(p\_id suppliers.id%type,

p\_company\_name suppliers.сompany\_name%type,

p\_adress suppliers.adress%type,

p\_phone\_number suppliers.phone\_number%type)

is

begin

update suppliers set id = p\_id,

сompany\_name = p\_company\_name,

adress = p\_adress,

phone\_number = p\_phone\_number

where suppliers.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--10. AddManufacturer

create or replace procedure AddManufacturer

(p\_company\_name manufacturers.сompany\_name%type,

p\_adress manufacturers.adress%type,

p\_phone\_number manufacturers.phone\_number%type)

is

begin

insert into MANUFACTURERS(сompany\_name, adress, phone\_number)

VALUES(p\_company\_name, p\_adress, p\_phone\_number);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--11. DeleteManufacturer

create or replace procedure DeleteManufacturer(p\_id manufacturers.id%type)

is

begin

delete from manufacturers where manufacturers.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--12. UpdateManufacturer

create or replace procedure UpdateManufacturer

(p\_id manufacturers.id%type,

p\_company\_name manufacturers.сompany\_name%type,

p\_adress manufacturers.adress%type,

p\_phone\_number manufacturers.phone\_number%type)

is

begin

update manufacturers set id = p\_id,

сompany\_name = p\_company\_name,

adress = p\_adress,

phone\_number = p\_phone\_number

where manufacturers.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--13. AddStore

create or replace procedure AddStore(p\_address stores.address%type)

is

begin

insert into STORES(address)

VALUES(p\_address);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--14. DeleteStore

create or replace procedure DeleteStore(p\_id stores.id%type)

is

begin

delete from stores where stores.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--15. UpdateStore

create or replace procedure UpdateStore

(p\_id stores.id%type,

p\_address stores.address%type)

is

begin

update stores set id = p\_id,

address = p\_address

where stores.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--16. AddSalesman

create or replace procedure AddSalesman

(p\_name salesmen.name%type,

p\_surname salesmen.surname%type,

p\_patronymic salesmen.patronymic%type,

p\_adress salesmen.adress%type,

p\_phone\_number salesmen.phone\_number%type)

is

begin

insert into SALESMEN(name,

surname,

patronymic,

adress,

phone\_number)

VALUES(p\_name, p\_surname, p\_patronymic, p\_adress, p\_phone\_number);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--17. DeleteSalesman

create or replace procedure DeleteSalesman(p\_id salesmen.id%type)

is

begin

delete from salesmen where salesmen.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--18. UpdateSalesman

create or replace procedure UpdateSalesman

(p\_id salesmen.id%type,

p\_name salesmen.name%type,

p\_surname salesmen.surname%type,

p\_patronymic salesmen.patronymic%type,

p\_adress salesmen.adress%type,

p\_phone\_number salesmen.phone\_number%type)

is

begin

update salesmen set id = p\_id,

name = p\_name,

surname = p\_surname,

patronymic = p\_patronymic,

adress = p\_adress,

phone\_number = p\_phone\_number

where salesmen.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--19. AddClient

create or replace procedure AddClient

(p\_nickname clients.nickname%type,

p\_name clients.name%type,

p\_surname clients.surname%type,

p\_patronymic clients.patronymic%type,

p\_email clients.email%type,

p\_adress clients.adress%type,

p\_phone\_number clients.phone\_number%type)

is

begin

insert into CLIENTS(nickname,

name,

surname,

patronymic,

email,

adress,

phone\_number)

VALUES(p\_nickname, p\_name, p\_surname, p\_patronymic, p\_email, p\_adress, p\_phone\_number);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--20. DeleteClient

create or replace procedure DeleteClient(p\_id clients.id%type)

is

begin

delete from clients where clients.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--21. UpdateClient

create or replace procedure UpdateClient

(p\_id clients.id%type,

p\_nickname clients.nickname%type,

p\_name clients.name%type,

p\_surname clients.surname%type,

p\_patronymic clients.patronymic%type,

p\_email clients.email%type,

p\_adress clients.adress%type,

p\_phone\_number clients.phone\_number%type)

is

begin

update clients set id = p\_id,

nickname = p\_nickname,

name = p\_name,

surname = p\_surname,

patronymic = p\_patronymic,

email = p\_email,

adress = p\_adress,

phone\_number = p\_phone\_number

where clients.id = p\_id;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--22. AddOrder

create or replace procedure AddOrder

(p\_salesman\_id orders.salesman\_id%type,

p\_client\_id orders.client\_id%type,

p\_product\_id orders.product\_id%type,

p\_amount orders.amount%type,

p\_store\_id orders.store\_id%type,

p\_order\_date orders.order\_date%type)

is

begin

insert into ORDERS(salesman\_id,

client\_id,

product\_id,

amount,

store\_id,

order\_date)

VALUES(p\_salesman\_id,

p\_client\_id,

p\_product\_id,

p\_amount,

p\_store\_id,

p\_order\_date);

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--23. DeleteOrder

create or replace procedure DeleteOrder(p\_order\_number orders.order\_number%type)

is

begin

delete from orders where orders.order\_number = p\_order\_number;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;

--24. UpdateOrder

create or replace procedure UpdateOrder

(p\_order\_number orders.order\_number%type,

p\_salesman\_id orders.salesman\_id%type,

p\_client\_id orders.client\_id%type,

p\_product\_id orders.product\_id%type,

p\_amount orders.amount%type,

p\_store\_id orders.store\_id%type,

p\_order\_date orders.order\_date%type)

is

begin

update orders set order\_number = p\_order\_number,

salesman\_id = p\_salesman\_id,

client\_id = p\_client\_id,

product\_id = p\_product\_id,

amount = p\_amount,

store\_id = p\_store\_id,

order\_date = p\_order\_date

where orders.order\_number = p\_order\_number;

exception

when others

then DBMS\_OUTPUT.put\_line(sqlerrm);

end;