МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных

Специализация

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Выполнил студент Костюкова Анна Олеговна

(Ф.И.О.)

Руководитель проекта

(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)

Консультант:

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

Нормоконтролер:

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

Курсовой проект защищен с оценкой

Минск 2020

# **Содержание**

[**Содержание** 2](#_Toc57091181)

[**ВВЕДЕНИЕ** 4](#_Toc57091182)

[1. Постановка задачи 5](#_Toc57091183)

[**1.1 Обзор прототипов** 5](#_Toc57091184)

[2. Разработка модели базы данных 9](#_Toc57091185)

[3. Разработка необходимых объектов 10](#_Toc57091186)

[3.1. Таблицы 10](#_Toc57091187)

[3.2. Процедуры 12](#_Toc57091188)

[4. Описание процедур импорта и экспорта 14](#_Toc57091189)

[5. Описание технологии 15](#_Toc57091190)

[4.1 Технология «Шифрование» 15](#_Toc57091191)

[4.2 Применение технологии «Шифрования» 16](#_Toc57091192)

[4.3 Технология «Маскирование» 18](#_Toc57091193)

[4.4 Применение технологии «Маскирование» 19](#_Toc57091194)

[**Список используемых источников** 21](#_Toc57091195)

Реферат

Пояснительная записка страниц, рисунок, литературных источников.

Реализация технологии шифрование и маскирование для базы данных приложения «Сеть для командной работы».

Целью проекта является разработка базы для программного средства «Сеть для командной работы».

Во первом разделе была обоснована задача курсового проекта.

Во втором разделе приводится разработка базы данных.

В третьем разделе приводиться описание разработки необходимых объектов базы данных.

В четвёртом разделе описываются процедуры экспорта и импорта.

В пятом разделе приводится тестирование производительности.

В шестом разделе приводится описание технологии.

В седьмом разделе приводится руководство пользователя.

В заключении приведены результаты проделанной работы.

# **ВВЕДЕНИЕ**

Центральной частью любого бизнес-приложения являются данные. Данные – это представленная в формализованном виде конкретная информация об объектах предметной области, их свойствах и взаимосвязях, отражающая события и ситуации в этой области. Именно на различных манипуляциях с данными строится большая часть пользовательских сценариев современных приложений. Эти сценарии описываются акронимом «CRUD» [1]. Он подразумевает под собой основные возможные пользовательские действия с данными – их создание (create), чтение (read), обновление (update) и удаление (delete).

На сегодняшний день, самой распространённой является реляционная модель структурирования данных. Реляционная база данных – это база данных, основанная на реляционной модели данных [2]. Она подразумевает под собой описание каждой сущности набором атрибутов, которые хранятся в соответствующих, для этих сущностей, таблицах.

Целью данной курсовой работы является разработка реляционной базы данных для программного обеспечения «Сеть для командной работы». База данных должна содержать необходимые данные о пользователе, командах, задачах, а также хранить сообщения, которыми обмениваются пользователи при выполнении поставленных целей.

В качестве СУБД для базы данных была выбрана Oracle Database 12c, в связи с ее производительностью и надежностью.

1. Постановка задачи

Основной задачей курсового проекта является разработка базы данных для программного обеспечения «Сеть для командной работы». Главная идея этой сети заключается в том, что пользователи объединяются в команды для решения поставленных задах, могут обсуждать в ходе их выполнения, а также могут легко отследить, в каком темпе идёт работа.

Задачу можно сформулировать следующим образом: создать необходимые сущности, которые будут отражать хранимые данные; охарактеризовать эти сущности с помощью соответствующих атрибутов, с корректно подобранными ограничениями целостности; связать таблицы между собой, тем самым получив полноценную реляционную структуру.

Необходимо реализовать шифрования и маскирование и протестировать производительность базы данных на таблицах, содержащих не менее 100 000 строк.

Функционально должны быть выполнены следующие задачи:

* Возможность авторизации по ролям;
* Возможность создавать, удалять и редактировать команду;
* Возможность добавлять и удалять пользователя в команду;
* Возможность прикреплять и удалять в задаче фотографии, видеоматериалы;
* Возможность осуществлять поиск пользователей по имени;
* Возможность раздавать и удалять задания для пользователей в команде;
* Возможность удалять и редактировать комментарии.

## **1.1 Обзор прототипов**

Trello – это сервис для командной работы, который позволяет планировать и публиковать текущие задачи, систематизировать их и следить за исполнением. Основан на японской системе досок канбан – удобного инструмента для организации работы.

Для начала работы перейти по адресу https://trello.com и авторизоваться (рисунок 1.1). Это можно сделать через существующий аккаунт или другим предложенным способом.

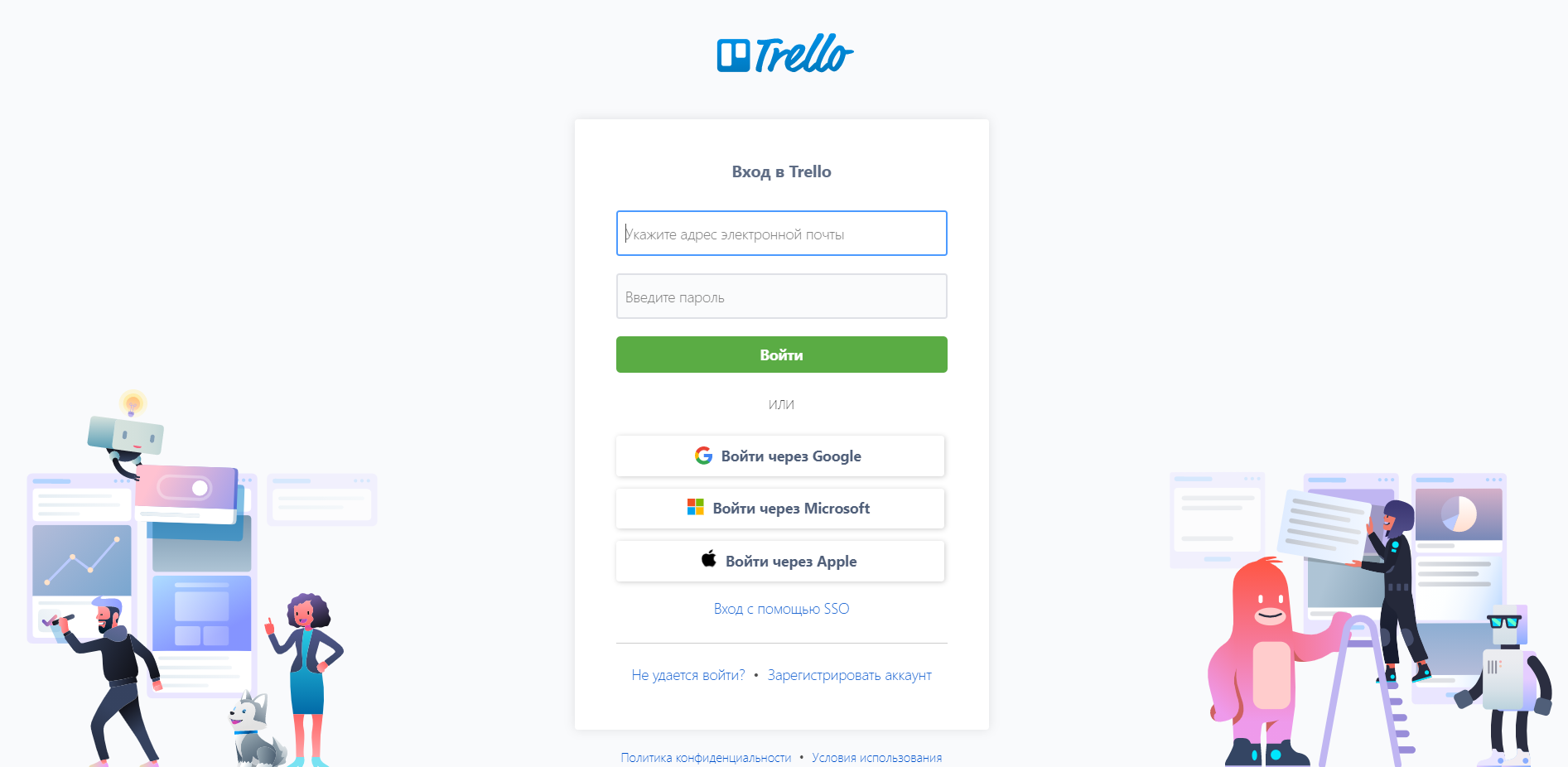


Рисунок 1.1 – Авторизация Trello

После авторизации открывается окно с персональными досками (рисунок 1.2).

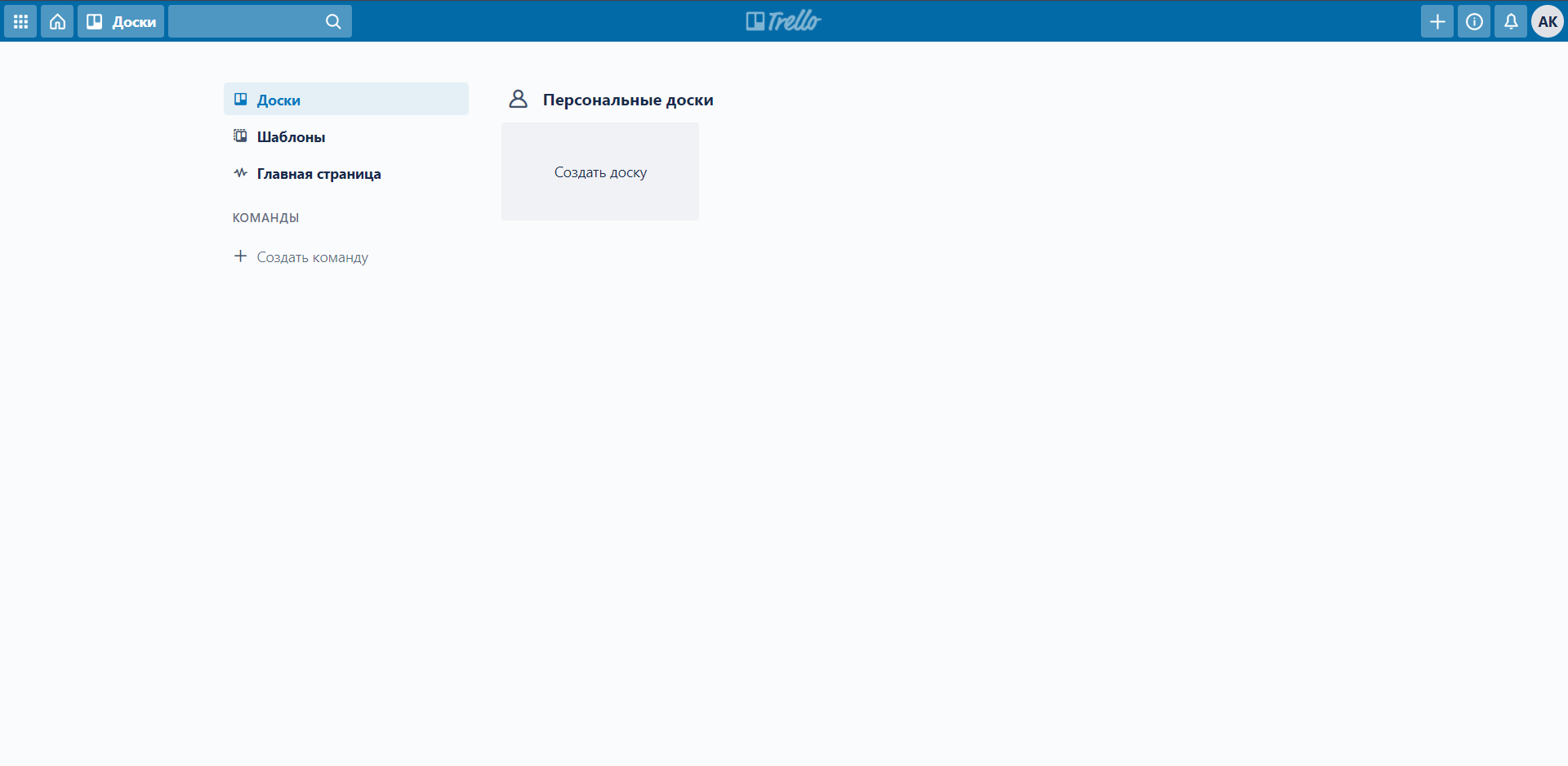


Рисунок 1.2 – Персональные доски

Весь сервис состоит из досок. Как правило, одна доска посвящена одному проекту (если в сервисе работает команда) или работе одного отдела, если речь идет о компании.

Здесь можно создать доску, для этого необходимо всего лишь ввести название доски, выбрать фон, а также выбрать тип доступа: приватная, доступная только для добавленных участников, или публичная, доступная для просмотра всем в Интернете (рисунок 1.3).

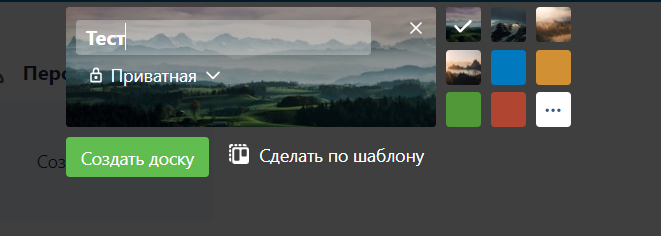


Рисунок 1.3 – Создание доски

После создание доски доступны несколько столбцов – это те самые задачи, которые выполняются в процессе работы (рисунок 1.4).

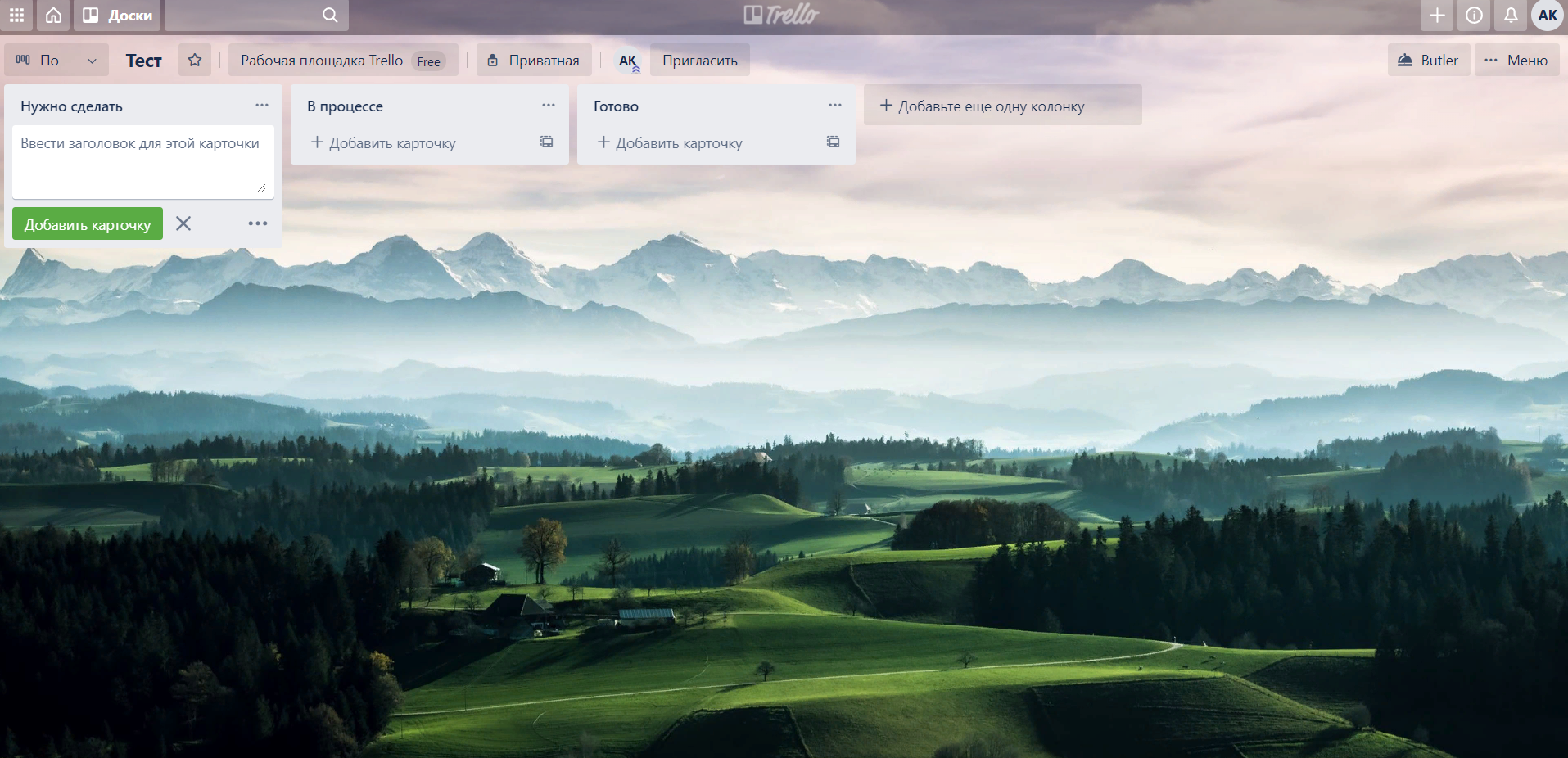


Рисунок 1.4 – Задачи

Каждая команда сама решает, какие списки задач и в каком порядке будут располагаться. Как правило, задачи разбиваются на:

* Запланированные;
* Текущие;
* Выполненные.

Карточки можно двигать как внутри одного списка, так и свободно перемещать между списками или досками. Списки тоже можно перемещать. Для любой задачи можно назначить людей, ответственных за ее выполнение. Trello предлагает множество полезных возможностей для оформления, настройки и управления своими функциональными элементами.

Что можно сделать с карточкой в Trello:

* Переименовать, заполнить описанием и редактировать текст с помощью простейших тегов Markdown;
* Присвоить метки, участников, срок выполнения, добавить файл или чек-лист;
* Добавить комментарии, смайлы, вложения, другие задачи, оповестить выбранных участников;
* Изменить положение блока в списке, перемещать его по спискам и другим доскам;
* Скопировать, следить за изменениями, заархивировать;
* Распечатать, экспортировать в JSON, поделиться ссылкой на карточку или ее почтовым адресом (письма будут появляться в виде комментариев);
* Удалить навсегда.

Списки тоже можно копировать, перемещать и архивировать. Меню с досками в Trello можно сделать фиксированным, а сами доски добавлять в «Избранные» и сортировать.

1. Разработка модели базы данных

Одним из ключевых моментов при проектировании и создании базы данных является грамотный анализ предметной области приложения. Как следствие, составление такой модели данных, которая будет правильно отражать то, как с этими с данными в общем, и этой моделью, в частности, подразумевается взаимодействовать.

Результатом корректно проведённого анализа, проектирования, и разработки, является модель, способная предоставить функционально все необходимые возможности для пользователя. Для приложения «Сеть для командной работы», можно выделить набор таких операций, как:

* возможность регистрации, авторизации, редактирования и удаления пользователей;
* возможность добавления, редактирования и удаления задач;
* возможность добавления, редактирования и удаления команд;
* возможность прикреплять фотографии и видеоматериалы к задаче;
* возможность назначать задачи на конкретных пользователей;
* возможность добавлять пользователей в команду;
* возможность искать пользователя по имени.

Основой инфраструктуры базы данных является грамотно спроектированная модель, которая отображает связь пользовательских таблиц. Правильное и корректное взаимодействие их друг с другом как раз и заключается в схеме базы данных со связями, верно отображающими их положение.

1. Разработка необходимых объектов

При разработке курсового проекта понадобились следующие объекты: таблицы и хранимые процедуры.

* 1. Таблицы

Центральным объектом любой базы данных является связанная совокупность таблиц. Таблицы ограничивают и упорядочивают хранимую информацию, а также обеспечивают связанность за счет внешних ключей. [2]

Для реализации базы данных для программного обеспечения «Сеть для командной работы» было разработано 10 таблиц.

Таблица UserRoles содержит возможные роли пользователей и включает 2 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении роли;
* Столбец role хранит роль пользователя, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой.

Таблица Users содержит информацию пользователей и включает 4 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении пользователя;
* Столбец username хранит имя пользователя, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой;
* Столбец password хранит пароль в зашифрованном виде, тип данных varchar2, максимальный размер 100 символов, ненулевой;
* Столбец roleId является внешним ключом, идентификатор роли, тип данных number, ненулевой.

Таблица Categories содержит категории команд и включает 2 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении категории;
* Столбец category хранит категорию команды, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой.

Таблица Teams содержит команды и включает 5 столбцов:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении команды;
* Столбец teamName хранит название команды, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой;
* Столбец teamDescription хранит описание команды, тип данных varchar2, максимальный размер 1000 символов, необязателен для заполнения;
* Столбец managerId является внешним ключом, идентификатор пользователя, который создал команду, тип данных number, ненулевой;
* Столбец categoryId является внешним ключом, идентификатор категории команды, тип данных number, ненулевой.

Таблица TaskStates содержит состояния задачи и включает 2 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении состояния задачи;
* Столбец stateName хранит состояние задачи, тип данных varchar2, максимальный размер 100 символов, уникальный, ненулевой.

Таблица Tasks содержит команды и включает 7 столбцов:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении задачи;
* Столбец taskName хранит название задачи, тип данных varchar2, максимальный размер 100 символов, ненулевой;
* Столбец taskDescription хранит описание задачи, тип данных varchar2, максимальный размер 1000 символов, необязателен для заполнения;
* Столбец dateOfCreation хранит дату создания задачи, тип данных date, ненулевой;
* Стоблец dateOfDeadline хранит дату завершения задачи, тип данных date, необязателен для заполнения;
* Столбец managerId хранит идентификатор пользователя, который создал задачу, тип данных number, ненулевой;
* Столбец teamId является внешним ключом, идентификатор команды, в которую добавлена задача, тип данных number, ненулевой;
* Столбец taskStateId является внешним ключом, идентификатор состояния задачи, тип данных number, ненулевой.

Таблица Comments хранит комментарии к задаче и включает 5 столбцов:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении комментария;
* Столбец creatorId является внешним ключом, идентификатор пользователя, который добавил комментарий, тип данных number, ненулевой;
* Столбец dateOfCreation хранит дату добавления комментария, тип данных date, ненулевой;
* Столбец commentContent хранит содержимое комментария, тип данных varchar2, максимальный размер 4000 символов, ненулевой;
* Столбец taskId является внешним ключом, идентификатор задачи, к которой добавлен комментарий, тип данных number, ненулевой.

Таблица TaskFiles содержит прикрепленные фотографии или видеоматериалы к задаче и включает 3 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении документа;
* Столбец fileUrl хранит ссылку на документ, прикрепленный к задаче, тип данных varchar2, максимальный размер 4000 символов, ненулевой;
* Столбец taskId является внешним ключом, идентификатор задачи, к которой прикреплен документ, тип данных number, ненулевой.

Таблица UsersTasks является реализацией связующей таблицы между пользователями и их задачами (связь многие ко многим) и включает 3 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении пользователя в задачу;
* Столбец userId является внешним ключом, хранит идентификатор пользователя, который записан в задачу, тип данных number, ненулевой;
* Столбец taskId является внешним ключам, хранит идентификатор задачи, тип данных number, ненулевой.

Таблица UsersTeams является реализацией связующей таблицы между пользователями и их командами (связь многие ко многим) и включает 3 столбца:

* Столбец id является первичным ключом, тип данных number, инкрементируется при добавлении пользователя в команду;
* Столбец userId является внешним ключом, хранит идентификатор пользователя, который записан в команду, тип данных number, ненулевой;
* Столбец teamId является внешним ключам, хранит идентификатор команды, тип данных number, ненулевой.

Листинг создания таблиц представлен в приложении А.

* 1. Процедуры

Процедура представляет собой модуль, выполняющий одно или несколько действий. Поскольку вызов процедуры в PL/SQL является отдельным исполняемым оператором, блок кода PL/SQL может состоять только из вызова процедуры. Процедуры относятся к числу ключевых компонентов модульного кода, обеспечивающих оптимизацию и повторное использование программной логики [3].

Для реализации базы данных для программного обеспечения «Сеть для командной работы» было разработано 7+5+6+2+2+2+1 процедур, которые реализуют функции CRUD [1].

Процедуры, связанные с пользователем, находятся в файле UserService.sql:

* CheckUser – проверка на наличие пользователя в базе данных;
* AuthorizeUser – авторизация пользователя;
* encryptPassword – шифрование пароля пользователя;
* RegistrationUser – регистрация пользователя;
* ChangeUserName – изменение имени пользователя;
* ChangePassword – изменение пароля пользователя;
* DeleteUser – удаление пользователя;
* SearchUserByName – поиск пользователя по его имени.

Процедуры, связанные с командой, находятся в файле TaskService.sql:

* CreateTeam – создание команды и автоматическое добавление пользователя, который создает, в команду;
* DeleteTeam – удаление одной команды;
* DeleteAllTeams – удаление всех команд пользователя, который их создавал;
* UpdateTeam – изменение информации о команде;
* AddUserInTeam – добавить других пользователей в команду;
* DeleteUserFromTeam – удалить пользователя из команды.

Процедуры, связанные с задачей, находятся в файле TeamService.sql:

* CreateTask – создание задача и установка состояния «В процессе» по умолчанию;
* CompletedTask – изменение состояния задачи на «Выполнено»;
* CancelledTask – изменение состояния задачи на «Отменено»;
* InProgressTask – изменение состояния задачи на «В процессе»;
* DeleteTask –удаление задачи;
* DeleteTasksInTeam – удаление всех задач в команде;
* AddUserInTask – добавление пользователя в задачу;
* DeleteUserFromTask – удаление пользователя из задачи;
* UpdateTask – изменение информации о задаче;
* AddFilesToTask – добавление фотографии или видеоматериала к задаче;
* DeleteFilesFromTask – удаление фотографии или видеоматериала в задаче.

Процедуры, связанные с комментарием, находятся в файле CommentService.sql:

* AddComment – создание комментария;
* DeleteComment – удаления комментария;
* DeleteAllComments – удаление всех комментариев пользователя;
* DeleteCommentsInTask – удаление всех комментариев к задаче;
* UpdateComment – изменение комментария.

Листинги создания процедур представлены в приложениях Б, В, Г, Д соответственно.

1. Описание процедур импорта и экспорта
2. Описание технологии

4.1 Технология «Шифрование»

В простейшем определении шифрование представляет собой «маскировку» данных, или их преобразование, при котором данные не могут использоваться посторонними.

Oracle поддерживает шифрование сетевых данных посредством своей опции Advanced Security. Для обеспечения шифрования данных Oracle предлагает два пакета PL/SQL, один из которых, DBMS\_OBFUSCATION\_TOOLKIT, является более старым. Этот пакет позволяет шифровать данные по алгоритму DES. Для обеспечения наивысшего уровня безопасности этот пакет инструментов поддерживает тройное DES-шифрование. Кроме того, он поддерживает использование безопасного криптографического хеша MD5.

Для шифрования и дешифрации данных можно также применять более новый пакет шифрования PL/SQL по имени DBMS\_CRYPTO. По сравнению с DBMS\_OBFUSCATION\_TOOLKIT пакет DBMS\_CRYPTO предлагает более широкое множество шифровальных и криптографических алгоритмов для обеспечения повышенной безопасности и проще в использовании. Этот пакет предназначен для замены более старого пакета DBMS\_OBFUSCATION\_TOOLKIT. Но какой бы пакет не использовался, придется управлять ключами шифрования данных, что является нетривиальной задачей. Часто нужно создавать представления, которые облегчают дешифрацию и шифрование данных, что увеличивает количество задач управления. Кроме того, согласно рекомендациям Oracle, шифрованные данные нельзя индексировать, что в некоторых случаях снижает ценность обоих этих пакетов шифрования.

Существует также и третий, более простой подход: шифрование с помощью функции прозрачного шифрования данных. Далее будет показано, как зашифровать табличные данные Oracle с использованием Oracle Wallet (Бумажник Oracle) для хранения ключей шифрования. При этом можно также индексировать столбцы шифрованной таблицы, тем самым избавляясь от одного из наибольших недостатков применения пакетов шифрования Oracle.

Одной из важнейших проблем при построении инфраструктуры шифрования является построение эффективной системы управления ключами. Если злоумышленник полу­чит доступ к ключам шифрования, то зашифрованные данные окажутся под угрозой независимо от сложности алгоритма. С другой стороны, некоторые пользователи должны иметь доступ к ключам для своей работы, и он должен быть достаточно простым для нормальной работы приложения. Проблема заключается в определении оптимального соотношения между простотой доступа и безопасностью ключей.

Существует немало распространенных, пользующихся коммерческой поддержкой алгоритмов шифрования. Все алгоритмы, поддерживаемые Oracle для приложений PL/SQL, относятся к категории алгоритмов с закрытым ключом (иногда называемых симметричными алгоритмами).

В Oracle чаще всего используются следующие алгоритмы:

* DES (Data Encryption Standard). Традиционно алгоритм DES занимал ведущие позиции в области шифрования. Он был разработан более 20 лет назад для Национального бюро стандартов (позднее переименованного в Национальный институт стандартов и технологий), и с тех пор был принят в качестве стандарта ISO. Об алгоритме DES и его истории можно рассказать очень много, но моей задачей является не описание алгоритма, а краткое описание его применения. Алгоритму DES необходим 64-разрядный ключ, но 8 бит ключа не используются. Чтобы подобрать ключ, злоумышленнику придется перебрать до 72 057 594 037 927 936 комбинаций. Возможностей DES было достаточно в течение нескольких десятилетий, но сейчас он постепенно уходит в прошлое. Современные мощные компьютеры способны перебрать даже огромное число комбинаций, необходимое для взлома ключа DES.
* DES3. В этой схеме, базирующейся на исходном алгоритме DES, данные шифруются дважды или трижды (в зависимости от режима вызова). DES3 использует 128- или 192-разрядный ключ; его длина определяется количеством проходов. Надежность алгоритма DES3 тоже была приемлемой в течение некоторого времени, но сейчас и этот алгоритм постепенно устаревает и не обеспечивает защиты от целенаправленных атак.
* AES. В ноябре 2001 года был одобрен новый стандарт AES (Advanced Encryption Standard), вступивший в силу в мае 2002 года. Полный текст стандарта можно найти по этому адресу.

Пакет DBMS\_CRYPTO появился в Oracle10g. В более ранних версиях пакет DBMS\_OBFUSCATION\_TOOLKIT предоставлял похожую (но не идентичную) функциональность. Старый пакет все еще остается доступным, но сейчас он считается устаревшим, и вместо него рекомендуется использовать новый пакет.

Для выполнения шифрования кроме входных данных необходимы еще четыре компонента:

* ключ шифрования;
* алгоритм шифрования;
* метод заполнения;
* метод сцепления.

Ключ шифрования предоставляете вы, а остальные компоненты предоставляет Oracle. Выбор осуществляется при помощи соответствующих констант пакета DBMS\_CRYPTO.

4.2 Применение технологии «Шифрования»

Для реализации технологии применяется функция DBMS\_CRYPTO.ENCRYPT. Функция получает четыре аргумента:

* src – исходные данные, подлежащие шифрованию должны иметь тип данных raw.
* key – ключ шифрования, тип данных raw. Длина ключа должна соответствовать вы­бранному алгоритму. Например, для алгоритма DES она должна быть не менее 64 бит.
* typ – определение трех компонентов алгоритм, механизм заполнения и метод сце­пления в виде суммы соответствующих констант.
* iv – необязательный вектор инициализации, еще один компонент схемы шиф­рования, затрудняющий анализ «закономерностей» в зашифрованном тексте.

В курсовой работе будут использоваться:

* алгоритм – AES c 128-разрядным ключом;
* метод сцепления – CBC;
* механизм заполнения – PKCS#5.

На рисунке 6.1 продемонстрирована процедура, которая применяют функцию шифрования для маскирования паролей от пользовательских аккаунтов.

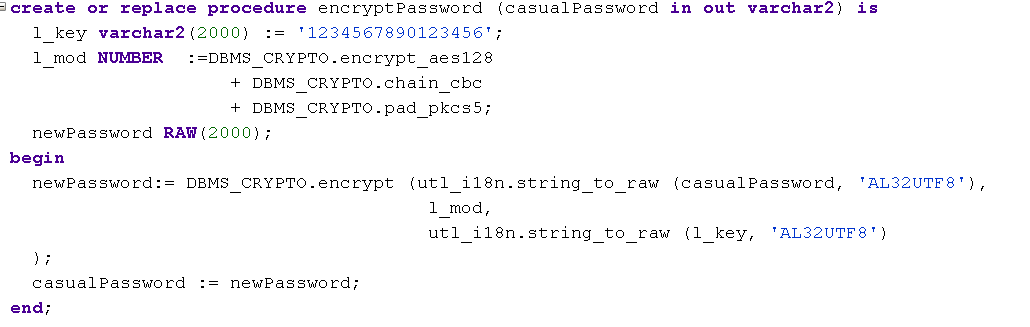


Рисунок 6.1 – Процедура шифрования пароля

Данная процедура принимает пароль в качестве единственного параметра и возвращает зашифрованный (рисунок 6.2-6.3).

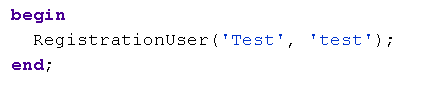


Рисунок 6.2 – Вызов процедуры шифрования



Рисунок 6.3 – Зашифрованный пароль

4.3 Технология «Маскирование»

Маскирование данных – это один из методов защиты конфиденциальной информации от несанкционированного доступа, при котором используются различные методы модификации данных, требующих защиты. Метод чаще всего используется при работе с базами данных (в том числе с помощью веб-приложений). Маскирование позволяет скрыть часть данных либо при создании копии базы данных, либо в ответе на запрос какой-либо информации, сохраняя общую информационную структуру и оставляя доступной ту часть данных, которая необходима для работы.

Применение маскировки данных наиболее распространено в процессе [разработки приложений](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F). При этом общепринятой практикой является использование производственных данных на всех этапах разработки: при создании приложений и [расширений](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%9F%D0%9E)), на этапах [тестирования](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) и [отладки](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B).

Основной проблемой с точки зрения руководства предприятий[[3]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D0%BA%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85#cite_note-3) и организаций является то, что разработчики приложений не всегда подвергаются проверке корпоративными службами безопасности прежде, чем получают доступ к производственным данным. Подобная практика может стать причиной возникновения серьёзных уязвимостей в системе безопасности, поскольку данные могут быть скопированы неавторизованными пользователями, а меры безопасности на различных этапах производства легко обойти.

Существует несколько видов маскирования данных:

* статическое маскирование – это необратимый процесс замены критичных данных на реалистичные, основанный на заданных правилах, при котором данные преобразуются в одном направлении, а первоначальные данные не могут быть получены, извлечены или восстановлены, в отличие от шифрования и токенизации, которые позволяют обратить процесс преобразования данных, тем самым увеличивая риск утечки критичной информации.;
* динамическое маскирование – предусматривает подмену критичных данных в режиме реального времени при обращении к производственной базе данных. Реальные данные не покидают базу данных; они заменяются на этапе запроса, например, на полностью реалистичные, без промежуточной записи.

Данные, подвергнутые маскировке, должны соответствовать следующим критериям:

* Замаскированные данные должны быть понятными с точки зрения логики приложения. Например, рассмотрим ситуацию, когда необходимо замаскировать элементы почтовых адресов, а названия городов заменить другими названиями. Если приложение имеет возможность проверки почтового индекса или поиска по почтовому индексу, то маскировка не должна мешать корректному выполнению этих функций. Это же касается алгоритмов проверки номеров платёжных карт, страховых свидетельств и т. д.
* Маскировка должна полностью исключать возможность восстановления реальных производственных данных из замаскированных. Например, может быть общеизвестным, что в организации работают 10 руководителей высшего звена, зарплата которых составляет свыше $300,000. Если в замаскированную базу данных отдела кадров включено 10 значений из указанного числового диапазона (свыше $300,000), то злоумышленник может восстановить оставшиеся сведения методом реверс-инжиниринга. Поэтому маскировка данных должна проводиться таким образом, чтобы гарантировать защиту записей, содержащих персональные сведения, а не просто отдельных элементов в разрозненных полях и таблицах.

Методы маскировки данных:

– Замена является одним из самых эффективных способов маскировки, позволяющим сохранить исходный внешний вид данных. Например, если исходная таблица БД содержит записи с информацией о клиентах, то реальные имена и фамилии можно заменить именами и фамилиями, взятыми из специально созданного (подготовленного) файла. Так, на первом этапе маскировки все имена клиентов могут заменяться произвольными мужскими именами, а на втором этапе можно произвести вставку женских имён в ячейки, соответствующие клиентам-женщинам (с помощью фильтрации списка клиента по ячейке с указанием пола). Применение подобного подхода к маскировке позволяет обеспечить должную анонимность записей и сохранить половое соотношение клиентов в замаскированной таблице. Важно, что база данных при этом выглядит реалистично, а факт маскировки информации не является очевидным.

– Перемешивание – очень распространённый способ маскировки данных. Он схож с методом замены, но при перемешивании данные для замены берутся из той же колонки таблицы, что и исходные данные. Однако, маскировка с использованием лишь одного этого метода имеет серьёзные недостатки. Злоумышленник, имеющий доступ к части реальных сведений, может восстановить остальные данные путём анализа методом «что, если?». Кроме того, перемешивание можно обратить вспять, дешифровав его алгоритм.

– Метод дисперсии (разброса) применяется при работе c полями БД, содержащими финансовую информацию и даты. Этот метод заключается в отклонении замаскированного числового значения от исходного на определённую величину. Например, при маскировке ячеек, содержащих данные о зарплате сотрудников, отклонение от исходного значения может составлять 10 %, поэтому замаскированная информация выглядит вполне реалистично и логично.

– Шифрование – это наиболее сложный способ маскировки данных. Алгоритм шифрования обычно предполагает наличие «ключа», необходимого для дешифровки и просмотра исходных данных. Шифрование также может сопровождаться преобразованием исходных данных в бинарный вид, что способно вызвать проблемы в работе приложений. Для выявления и устранения конфликтов внутри приложений необходимо проводить тестирование с передачей исходной информации тестировщикам, а это, в свою очередь, предполагает проверку задействованных в тестировании IT-специалистов службой безопасности.

4.4 Применение технологии «Маскирование»

Для реализации технологии в курсовой работе используется вызов процедуры, представленная на рисунке 6.4.

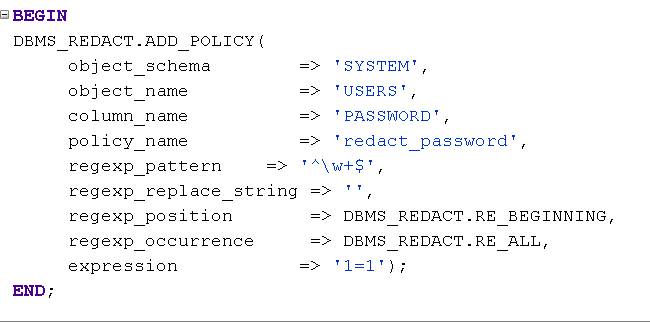


Рисунок 6.4 – Создание политики выполнения

После создания политики выполнения, у пользователей, отличных от system и sys, пароль заменяется на пустую строку (рисунок 6.5).

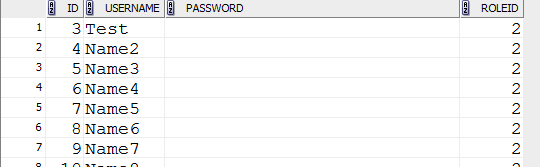


Рисунок 6.5 – Замена пароля

Таким образом, можно скрывать данные от пользователей, которые не имеют нужные права на просмотр.

# **Список используемых источников**

1. Свободная энциклопедия Википедия [Электронный ресурс] /CRUD – Режим доступа: https://ru.wikipedia.org/wiki/CRUD ­– Дата доступа: 23.11.2020.
2. Базы Данных. Учебное пособие [Электронный ресурс] /Базы данных – Режим доступа: <https://siblec.ru/informatika-i-vychislitelnaya-tekhnika/bazy-dannykh> ­– Дата доступа: 23.11.2020.
3. Patcher It Community [Электронный ресурс] / Процедуры PL/SQL: программирование на примерах – Режим доступа: https://oracle-patches.com/db/sql/3611-процедуры-pl-sql-программирование-на-примерах ­– Дата доступа: 23.11.2020.
4. Patcher It Community [Электронный ресурс] / Шифрование и дешифрование в PL/SQL для БД Oracle на примере – Режим доступа: <https://oracle-patches.com/db/sql/3800-shifrovanie-i-deshifrovanie-v-pl-sql-dlya-bd-oracle-na-primere> – Дата доступа: 23.11.2020.