|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***по дисциплине «Базы данных»***

***НА ТЕМУ:***

***\_\_ База данных Партия любителей пива\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-42Б |  | 17.04.2021 | И.С. Марчук |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Руководитель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |

*2021 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_ИУ6\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Пролетарский

« 9 » февраля 2021 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине Базы данных

Студент группы ИУ6-42б

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Марчук Иван Сергеевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсовой работы \_\_\_\_\_\_База данных «Партия любителей пива» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_учебная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения работы: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

***Задание:*** разработать базу данных «Партия любителей пива», содержащую не менее 7 таблиц. В основной таблице должно содержаться не менее 100 тыс. записей, в других таблицах должно содержаться не менее 10 записей. В БД должно быть разработано не менее 5 сложных запросов и не менее 2 сложных отчетов.

***Оформление курсовой работы:***

Расчетно-пояснительная записка на 25-30 листах формата А4.

Дата выдачи задания «8» февраля 2021 г.

|  |  |  |  |
| --- | --- | --- | --- |
| **Руководитель** |  |  | Д.А. Миков |
|  |  | (Подпись, дата) | (И.О. Фамилия) |
| **Студент** |  |  | И.С. Марчук |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

Содержание

Введение.................................................................................................................3

Глава 1. Техническое задание ..............................................................................4

1.1 Описание предметной области ............................................................4

Глава 2. Проектирование.......................................................................................5

2.1 Выбор средств реализации БД и интерфейса......................................5

2.2 Описание сущностей БД ......................................................................6

2.3 Заполненные таблицы .........................................................................14

2.4 Описание функций БД.........................................................................19

2.5 Исходные коды веб приложения APEX.............................................20

Глава 3. Результат ................................................................................................66

3.1 Интерфейс пользователя .....................................................................66

Заключение ...........................................................................................................73

Список источников ..............................................................................................74

Приложение ..........................................................................................................75

**Введение**

Сейчас существует большое количество политических партий, включающих в себя сотни и тысячи людей. Ежедневно необходимо рассылать партийные новости, вести учет всех работников, осуществлять заказы необходимых для партии вещей и контролировать выплаты партийных взносов. Вести учет всего этого переписываясь по электронной почте или используя бумажные носители очень неудобно, а иногда и невозможно. Необходима система, позволяющая структурировать большое количество записей о пользователях в едином пространстве для поддержания актуальности сведений и имеющая возможность массовой рассылки уведомлений для членов партии. Все эти возможности должны быть доступны в любое время и с любого компьютера, подключенного к Интернету.

**Глава 1. Техническое задание**

**Цель:** разработать систему, для партии любителей пива позволяющую вести учет партийных работников и их партийных билетов, публиковать общедоступные новости, делать партийные заказы и контролировать выплаты членских взносов.

**Задачи:**

- разработать БД для хранения информации о членах партии и их парт. билетах, о различных марках пива и партийных заказах, о глобальных новостях и о членских взносах;

- разработать функции и запросы в БД, осуществляющие получение и запись необходимой информации через простой интерфейс;

- разработать графический интерфейс пользователя для взаимодействия с реализованной БД с использованием веб-браузера.

**1.1 Описание предметной области**

Система должна учитывать особенности устройства и взаимодействия между собой частей партии: списки пользователей, уникальный партийный билет для каждого пользователя выдаваемый после одобрения руководством и имеющий также возможность менять свою должность. Необходима система должностей варьирующая доступ к некоторым функциям и регулирующая членский взнос. Также обязательна система рассылки глобальных новостей. Так как партия специализируется на марках пива, то необходим каталог, содержащий в себе наиболее полный список всех сортов с полным описанием и возможность создания партийных заказов пива (в дальнейшем эту систему можно будет расширить до любой специализации).

**Глава 2. Проектирование**

**2.1 Выбор средств реализации БД и интерфейса**

В качестве СУБД в данной работе используется ORACLE Database 11g. Это открытая, кроссплатформенная СУБД, поддерживающая PL/SQL, индексы и имеющая продвинутый конструктор приложений APEX поддерживающий огромное количество форматов от веб-сайтов до мобильных приложений. В качестве пользовательского интерфейса было выбрано именно приложение, созданное в APEX. Разработка собственного пользовательского интерфейса обусловлена особенностями разработки интерфейсов для данной предметной области. Схема разработанной БД представлена на рисунке 1:

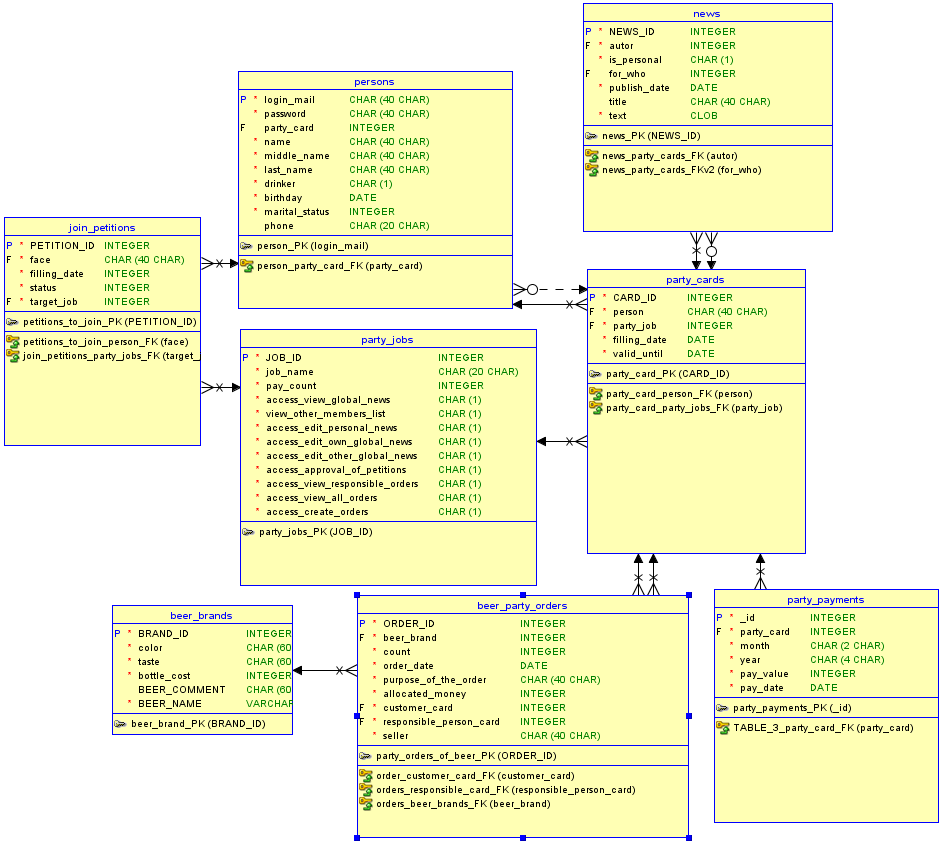


Рисунок 1 - Схема БД

**2.2 Описание сущностей БД**

Beer brands – таблица марок пива

CREATE TABLE beer\_brands (

brand\_id INTEGER NOT NULL PRIMARY KEY,

color CHAR(60 CHAR) NOT NULL,

taste CHAR(60 CHAR) NOT NULL,

bottle\_cost INTEGER NOT NULL,

beer\_comment CHAR(60 CHAR),

beer\_name VARCHAR2(60) NOT NULL

);

- brand\_id – идентификатор бренда пива;

- color – цвет пива;

- taste – вкус пива;

- bottle\_cost – цена за бутылку;

- beer\_comment – комментарий;

- beer\_name – название марки.

beer\_party\_orders – таблица партийных заказов пива

CREATE TABLE beer\_party\_orders (

order\_id INTEGER NOT NULL PRIMARY KEY,

beer\_brand INTEGER NOT NULL,

count INTEGER NOT NULL,

order\_date DATE NOT NULL,

purpose\_of\_the\_order CHAR(40 CHAR) NOT NULL,

allocated\_money INTEGER NOT NULL,

customer\_card INTEGER NOT NULL,

responsible\_person\_card INTEGER NOT NULL,

seller CHAR(40 CHAR) NOT NULL,

FOREIGN KEY(customer\_card) REFERENCES party\_cards (card\_id) CASCADE,

FOREIGN KEY (beer\_brand) REFERENCES beer\_brands (brand\_id) CASCADE,

FOREIGN KEY(responsible\_person\_card) REFERENCES party\_cards(card\_id) CASCADE

);

- order\_id – идентификатор заказа;

- beer\_brand – идентификатор марки пива;

- count – количество бутылок;

- order\_date – дата заказа;

- purpose\_of\_the\_order – цель заказа;

- allocated\_money – выделенные деньги;

- customer\_card – идентификатор заказчика;

- responsible\_person\_card – идентификатор ответственного лица;

- seller – продавец.

Join petitions – заявки на изменение партийной должности и продление партийного билета

CREATE TABLE join\_petitions (

petition\_id INTEGER NOT NULL PRIMARY KEY,

face CHAR(40 CHAR) NOT NULL,

filling\_date INTEGER NOT NULL,

status INTEGER NOT NULL,

target\_job INTEGER NOT NULL,

FOREIGN KEY (target\_job) REFERENCES party\_jobs (job\_id) CASCADE,

FOREIGN KEY (face) REFERENCES persons (login\_mail) CASCADE

);

- petition\_id – уникальный номер записи (первичный ключ);

- face – логин лица, подавшего заявление;

- filling\_date – дата подачи;

- status – статус заявления;

- target\_job – целевая должность.

News – партийные новости

CREATE TABLE news (

news\_id INTEGER NOT NULL PRIMARY KEY,

autor INTEGER NOT NULL,

is\_personal CHAR(1) NOT NULL,

for\_who INTEGER,

publish\_date DATE NOT NULL,

title CHAR(40 CHAR),

text CLOB NOT NULL,

FOREIGN KEY (autor) REFERENCES party\_cards (card\_id) CASCADE, FOREIGN KEY (for\_who) REFERENCES party\_cards (card\_id) SET NULL

);

- news\_id – уникальный номер записи (первичный ключ);

- autor – автор записи;

- is\_personal – новость - личная или для массовой рассылки;

- for\_who – в случае личной новости указывается также и получатель;

- publish\_date – дата публикации новости;

- title – заголовок новости;

- text – текст новости.

party\_cards – партийные билеты участников

CREATE TABLE party\_cards (

card\_id INTEGER NOT NULL PRIMARY KEY,

person CHAR(40 CHAR) NOT NULL,

party\_job INTEGER NOT NULL,

filling\_date DATE NOT NULL,

valid\_until DATE NOT NULL,

FOREIGN KEY (party\_job) REFERENCES party\_jobs (job\_id) CASCADE,

FOREIGN KEY (person) REFERENCES persons (login\_mail) CASCADE

);

- card\_id – идентификатор билета;

- person – идентификатор лица использующего билет;

- party\_job – должность;

- filling\_date – дата получения;

- valid\_until – срок годности.

party\_jobs – партийные должности

CREATE TABLE party\_jobs (

job\_id INTEGER NOT NULL PRIMARY KEY,

job\_name CHAR(20 CHAR) NOT NULL,

pay\_count INTEGER NOT NULL,

access\_view\_global\_news CHAR(1) NOT NULL,

view\_other\_members\_list CHAR(1) NOT NULL,

access\_edit\_personal\_news CHAR(1) NOT NULL,

access\_edit\_own\_global\_news CHAR(1) NOT NULL,

access\_edit\_other\_global\_news CHAR(1) NOT NULL,

access\_approval\_of\_petitions CHAR(1) NOT NULL,

access\_view\_responsible\_orders CHAR(1) NOT NULL,

access\_view\_all\_orders CHAR(1) NOT NULL,

access\_create\_orders CHAR(1) NOT NULL

);

- job\_id – идентификатор записи;

- job\_name – название должности;

- pay\_count – членский взнос;

- access\_view\_global\_news – доступ на просмотр глобальных новостей;

- view\_other\_members\_list – доступ на просмотр списка пользователей;

- access\_edit\_personal\_news – доступ на отправку сообщений;

- access\_edit\_own\_global\_news – создание глобальных новостей;

- access\_edit\_other\_global\_news – доступ на редактирование чужих глобальных новостей (модерирование);

- access\_approval\_of\_petitions – доступ на изменение чужих парт билетов;

- access\_view\_responsible\_orders – доступ на просмотр заказов за которые лицо является ответственным;

- access\_view\_all\_orders – доступ на просмотр всех заказов;

- access\_create\_orders – доступ на создание партийных заказов.

party\_payments – членские взносы

CREATE TABLE party\_payments (

"\_id" INTEGER NOT NULL PRIMARY KEY,

party\_card INTEGER NOT NULL,

month CHAR(2 CHAR) NOT NULL,

year CHAR(4 CHAR) NOT NULL,

pay\_value INTEGER NOT NULL,

pay\_date DATE NOT NULL,

FOREIGN KEY (party\_card) REFERENCES party\_cards (card\_id) CASCADE

);

- \_id – идентификатор взноса;

- party\_card – идентификатор билета человека, внесшего взнос;

- month – оплачиваемый месяц;

- year – оплачиваемый год;

- pay\_value – сумма взноса;

- pay\_date – дата оплаты.

persons – члены партии

CREATE TABLE persons (

login\_mail CHAR(40 CHAR) NOT NULL PRIMARY KEY,

password CHAR(40 CHAR) NOT NULL,

party\_card INTEGER,

name CHAR(40 CHAR) NOT NULL,

middle\_name CHAR(40 CHAR) NOT NULL,

last\_name CHAR(40 CHAR) NOT NULL,

drinker CHAR(1) NOT NULL,

birthday DATE NOT NULL,

marital\_status INTEGER NOT NULL,

phone CHAR(20 CHAR),

FOREIGN KEY (party\_card) REFERENCES party\_cards (card\_id) SET NULL

);

- login\_mail – электронная почта-логин-идентификатор;

- password – пароль;

- party\_card – идентификатор партийного билета;

- name – имя;

- middle\_name – фамилия;

- last\_name – отчество;

- drinker – пьющий/не пьющий;

- birthday – дата рождения;

- marital\_status – статус отношений;

- phone – контактный телефон.

**2.3 Заполненные таблицы**

Заполненная таблица «Пользователи» на рисунке 2

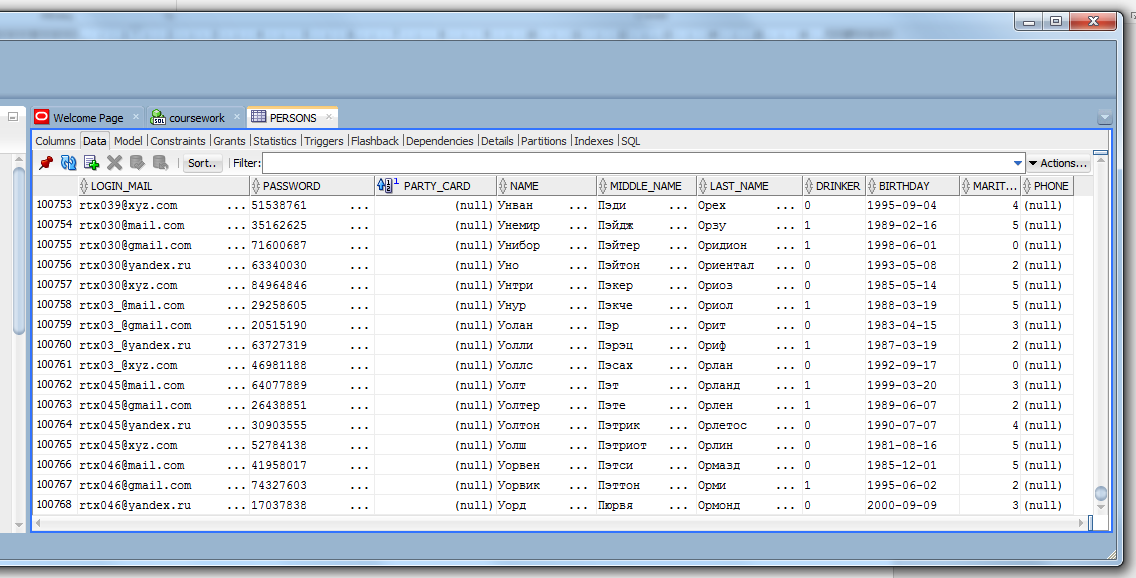


Рисунок 2 – PERSONS

Заполненная таблица «Членские взносы» на рисунке 3

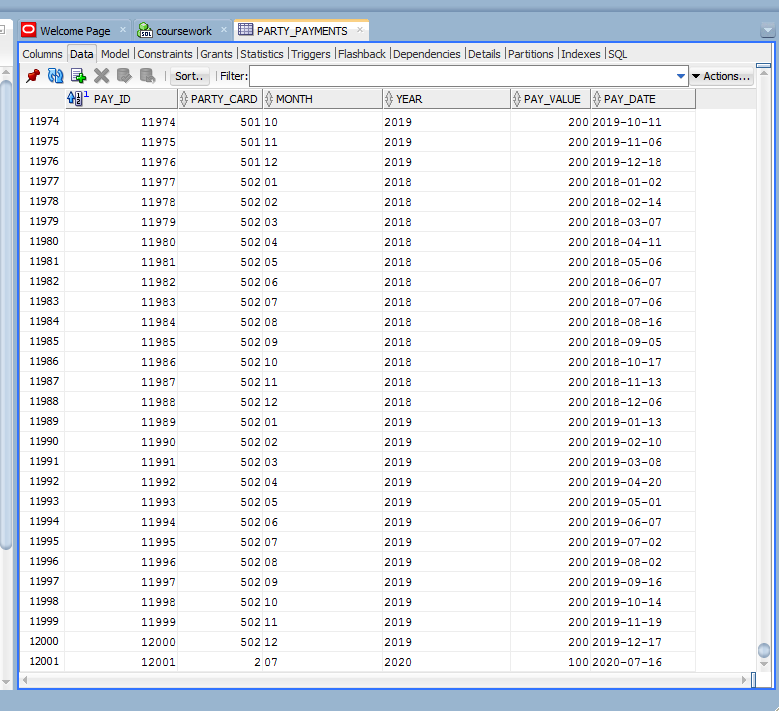


Рисунок 3 – PARTY\_PAYMENTS

Заполненная таблица «Партийные должности» на рисунке 4

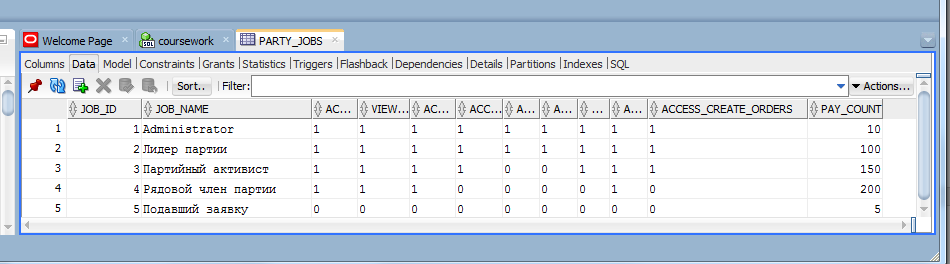


Рисунок 4 – PARTY\_JOBS

Заполненная таблица «Партийные билеты» на рисунке 5

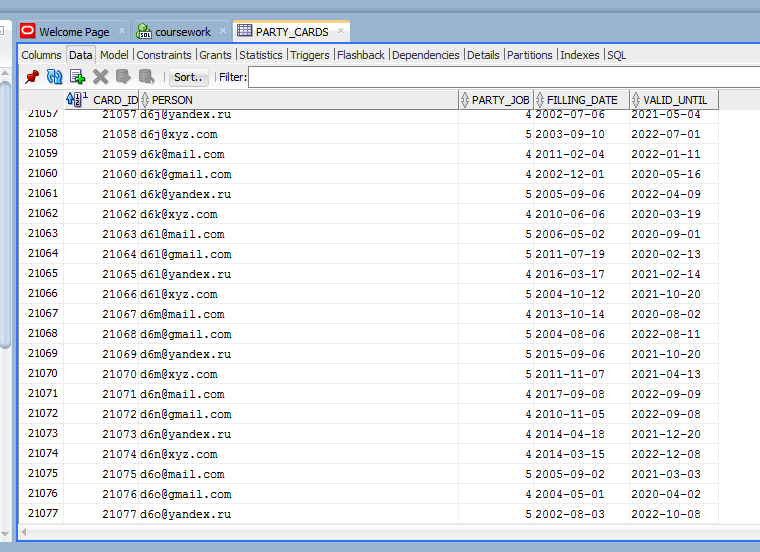


Рисунок 5 – PARTY\_CARDS

Заполненная таблица «Новости» на рисунке 6

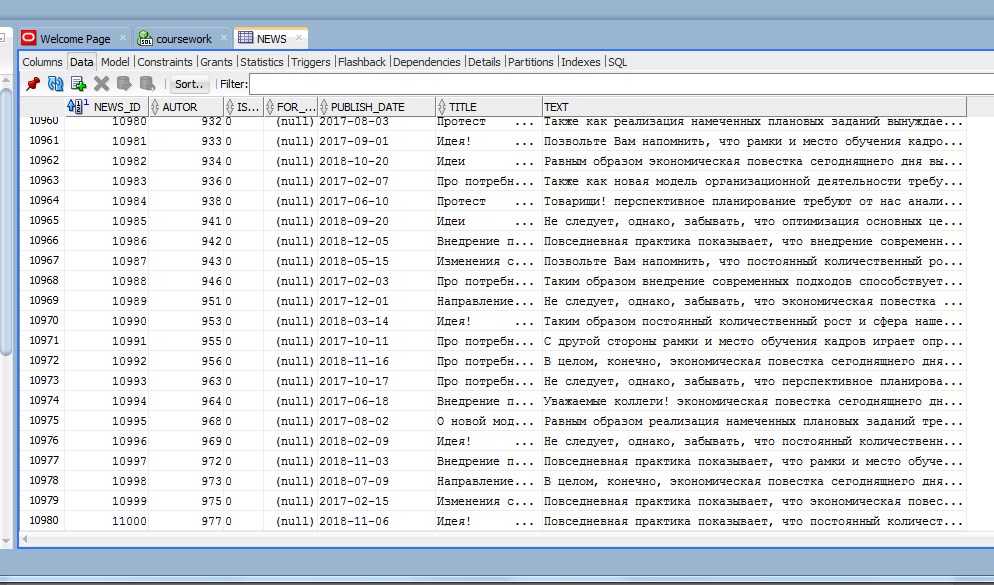


Рисунок 6 – NEWS

Заполненная таблица «Заявки на изменение или продление парт билета» на рисунке 7

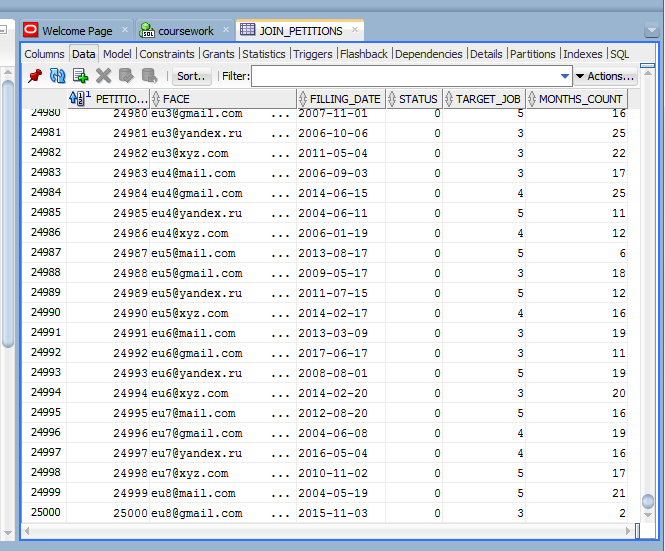


Рисунок 7 – JOIN\_PETITIONS

Заполненная таблица «Партийные заказы» на рисунке 8

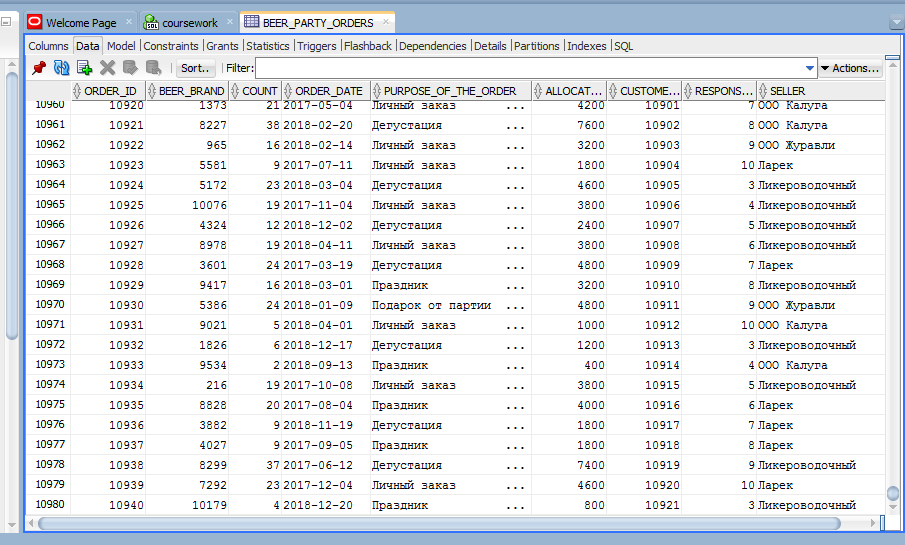


Рисунок 8 – BEER\_PARTY\_ORDERS

Заполненная таблица «Марки пива» на рисунке 9

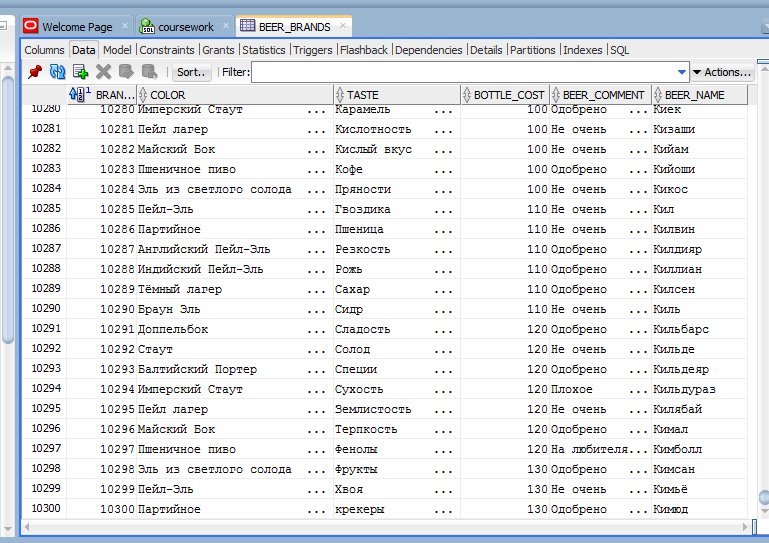


Рисунок 9 – BEER\_BRANDS

**2.4 Описание функций БД**

Были разработаны функции, помогающие осуществлять контроль и поддерживать целостность БД. Они представлены в приложении.

Система может подстраиваться под особенности устройства и взаимодействия между собой частей партии, так как она имеет в себе: списки пользователей, уникальный партийный билет для каждого пользователя выдаваемый после одобрения руководством и имеющий также возможность менять свою должность. Имеется настраиваемая система должностей варьирующая доступ к некоторым функциям и регулирующая членский взнос. Также реализована система рассылки глобальных новостей. Так как партия специализируется на марках пива, имеется также каталог, содержащий в себе наиболее полный список всех сортов с детальным описанием, а также есть возможность создания партийных заказов пива, которую в дальнейшем можно будет доработать как систему учета и других заказов для удовлетворения нужд партии.

**2.5 Исходные коды веб приложения APEX**

**Страница входа**

PageProcessing: **SetUserNameCookie** – помещает ссылку на текущего пользователя в куки браузера

apex\_authentication.send\_login\_username\_cookie( p\_username=>lower(:P101\_USERNAME)

);

PageProcessing: **Login** – вызывает из БД функцию авторизации, сама вызывается по нажатию кнопки «Войти»

Pkg\_Security.Process\_Login(

:P101\_USERNAME,

:P101\_PASSWORD,

:APP\_ID

);

**Страница регистрации**

DynamicAction(PL/SQL): **CreatePress** – вызывается при нажатии на кнопку «создать учетную запись»

declare

p\_count INTEGER(10);

answer\_string VARCHAR2(300);

error\_flag boolean;

begin

error\_flag := false;

-- логин

if(TRIM(:P102\_MAIL) IS NULL)then

answer\_string := answer\_string || 'Логин не может быть пустым; ';

error\_flag := true;

end if;

SELECT

COUNT(\*)

INTO

p\_count

FROM PERSONS p\_persons

WHERE TRIM(p\_persons.LOGIN\_MAIL) = TRIM(:P102\_MAIL);

--answer\_string := answer\_string || to\_char(p\_count);

if(p\_count <> 0)then

answer\_string := answer\_string || 'Такой логин уже занят; ';

error\_flag := true;

end if;

-- пароль

if(TRIM(:P102\_PASSWORD) IS NULL)then

answer\_string := answer\_string || 'Поле Пароль не может быть пустым; ';

error\_flag := true;

end if;

-- эта проверка выдает false, если хотябы один из операндов null

if(TRIM(:P102\_PASSWORD) <> TRIM(:P102\_PASSWORD\_CONFIRM))then

answer\_string := answer\_string || 'Пароли должны совпадать; ';

error\_flag := true;

end if;

-- Имя

if(TRIM(:P102\_NAME) IS NULL)then

answer\_string := answer\_string || 'Поле Имя не может быть пустым; ';

error\_flag := true;

end if;

-- Отчество

if(TRIM(:P102\_MIDDLE\_NAME) IS NULL)then

answer\_string := answer\_string || 'Поле Отчество не может быть пустым; ';

error\_flag := true;

end if;

-- Фамилия

if(TRIM(:P102\_LAST\_NAME) IS NULL)then

answer\_string := answer\_string || 'Поле Фамилия не может быть пустым; ';

error\_flag := true;

end if;

-- День рождения

if(TRIM(:P102\_BIRTHDAY) IS NULL)then

answer\_string := answer\_string || 'Поле День рождения не может быть пустым; ';

error\_flag := true;

end if;

-- создание аккаунта

if(not error\_flag)then

INSERT INTO "PERSONS" (

LOGIN\_MAIL,

PASSWORD,

NAME,

MIDDLE\_NAME,

LAST\_NAME,

DRINKER,

BIRTHDAY,

MARITAL\_STATUS,

PHONE

) VALUES (

TRIM(:P102\_MAIL),

TRIM(:P102\_PASSWORD),

TRIM(:P102\_NAME),

TRIM(:P102\_MIDDLE\_NAME),

TRIM(:P102\_LAST\_NAME),

TRIM(:P102\_DRINKER),

TRIM(:P102\_BIRTHDAY),

TRIM(:P102\_MATRIAL\_STATUS),

TRIM(:P102\_PHONE)

);

return 'Проверка пройдена успешно. Создаем аккаунт...';

end if;

return answer\_string;

end;

**Персональная страница**

SQL Query: **P\_MAIL** – запрос получения логина пользователя

select

PERSONS.LOGIN\_MAIL as "Mail"

from PERSONS PERSONS

where ( APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(PERSONS.LOGIN\_MAIL))

PL/SQL: **P1\_NAME** – запрос в базу данных для получения имени пользователя

declare

lastName PERSONS.LAST\_NAME%Type;

name PERSONS.NAME%Type;

middleName PERSONS.MIDDLE\_NAME%Type;

begin

Begin

Select

TRIM(person.LAST\_NAME),

TRIM(person.NAME),

TRIM(person.MIDDLE\_NAME)

Into lastName,

name,

middleName

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

return lastName||' '||name||' '||middleName;

Exception

When No\_Data\_Found Then

return '';

End;

end;

PL/SQL: **P1\_DRINKER**– запрос в базу данных для получения статуса отношения к алкоголю у пользователя

declare

drinker PERSONS.DRINKER%Type;

begin

Begin

Select

TRIM(person.DRINKER)

Into

drinker

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(drinker = 0)then

return 'Не пьющий';

else

return 'Любитель выпить';

end if;

Exception

When No\_Data\_Found Then

return '';

End;

end;

PL/SQL: **P1\_BIRTHDAY**– запрос в базу данных для получения даты рождения пользователя

select

PERSONS.BIRTHDAY

from PERSONS PERSONS

where ( APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(PERSONS.LOGIN\_MAIL))

PL/SQL: **P1\_MARITAL\_STATUS** – запрос в базу данных для получения статуса отношений пользователя

declare

status PERSONS.MARITAL\_STATUS%Type;

begin

Begin

Select

person.MARITAL\_STATUS

Into

status

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

return

CASE status

WHEN 1 THEN 'Не состою в браке'

WHEN 2 THEN 'Встречаюсь'

WHEN 3 THEN 'Помолвлен(а)'

WHEN 4 THEN 'Женат/Замужем'

WHEN 5 THEN 'В гражданском браке'

WHEN 6 THEN 'Влюблен(а)'

WHEN 7 THEN 'Все сложно'

WHEN 8 THEN 'В активном поиске'

ELSE 'Не выбрано'

END;

Exception

When No\_Data\_Found Then

return '';

End;

end;

PL/SQL: **P1\_PHONE** – запрос в базу данных для получения номера телефона пользователя

select

PERSONS.PHONE

from PERSONS PERSONS

where ( APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(PERSONS.LOGIN\_MAIL))

PL/SQL: **P1\_PARTYCARD\_ID** – запрос в базу данных для получения номера партийного билета пользователя

declare

card\_id PERSONS.PARTY\_CARD%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(card\_id IS NULL)then

return 'У вас пока нет парт. билета';

else

return card\_id;

end if;

Exception

When No\_Data\_Found Then

return '';

End;

end;

PL/SQL: **P1\_FILLING\_DATE** – запрос в базу данных для получения даты оформления партийного билета пользователя

declare

card\_id\_p PERSONS.PARTY\_CARD%Type;

card\_date\_p PARTY\_CARDS.FILLING\_DATE%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

card\_id\_p

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(card\_id\_p IS NULL) then

return '';

end if;

Select

PARTY\_CARDS.FILLING\_DATE

Into

card\_date\_p

From PARTY\_CARDS

Where PARTY\_CARDS.CARD\_ID = card\_id\_p;

if(card\_date\_p IS NULL)then

return '';

else

return card\_date\_p;

end if;

Exception

When No\_Data\_Found Then

return '';

End;

end;

PL/SQL: **P1\_VALID\_UNTIL** – запрос в базу данных для получения даты действия партийного билета пользователя

declare

card\_id\_p PERSONS.PARTY\_CARD%Type;

card\_date\_p PARTY\_CARDS.FILLING\_DATE%Type;

begin

Begin

Select

person\_p.PARTY\_CARD

Into

card\_id\_p

From PERSONS person\_p

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person\_p.LOGIN\_MAIL);

Select

p\_card.VALID\_UNTIL

Into

card\_date\_p

From PARTY\_CARDS p\_card

Where card\_id\_p = p\_card.CARD\_ID;

if(card\_id\_p IS NULL) then

return '';

end if;

if(card\_date\_p IS NULL)then

return '';

else

if(sysdate > card\_date\_p) then

return card\_date\_p||', not valid';

else

return card\_date\_p||', valid';

end if;

end if;

Exception

When No\_Data\_Found Then

return '';

End;

end;

PL/SQL: **P1\_JOB\_NAME** – запрос в базу данных для получения работы указанной в партийном билете пользователя

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_job\_name PARTY\_JOBS.JOB\_NAME%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return '';

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return '';

end if;

Select

jobs.JOB\_NAME

Into

p\_job\_name

From PARTY\_JOBS jobs

Where p\_job\_id = jobs.JOB\_ID;

if(p\_job\_name IS NULL)then

return '';

else

return p\_job\_name;

end if;

Exception

When No\_Data\_Found Then

return '';

End;

end;

PL/SQL: **P1\_PAY\_COUNT** – запрос в базу данных для получения размера членского взноса пользователя

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_pay\_count PARTY\_JOBS.PAY\_COUNT%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return '';

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return '';

end if;

Select

p\_job.PAY\_COUNT

Into

p\_pay\_count

From PARTY\_JOBS p\_job

Where p\_job\_id = p\_job.JOB\_ID;

if(p\_pay\_count IS NULL)then

return '';

else

return p\_pay\_count;

end if;

Exception

When No\_Data\_Found Then

return '';

End;

end;

PL/SQL: **P1\_PAY\_STATUS** – запрос в базу данных для получения статуса выплаты членского взноса пользователя в этом месяце

declare

p\_pays\_count integer;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_pay\_count PARTY\_JOBS.PAY\_COUNT%Type;

begin

select

count(PARTY\_PAYMENTS.PAY\_ID)

INTO

p\_pays\_count

FROM

PARTY\_PAYMENTS

JOIN PARTY\_CARDS ON PARTY\_CARDS.CARD\_ID = PARTY\_PAYMENTS.PARTY\_CARD

JOIN PARTY\_JOBS ON PARTY\_JOBS.JOB\_ID = PARTY\_CARDS.PARTY\_JOB

WHERE

PARTY\_PAYMENTS.YEAR = SUBSTR(TO\_CHAR(SYSDATE), 1, 4) AND

PARTY\_PAYMENTS.MONTH = SUBSTR(TO\_CHAR(SYSDATE), 6, 2) AND

PAY\_VALUE >= PARTY\_JOBS.PAY\_COUNT;

if(p\_pays\_count > 0)then

RETURN 'Этот месяц оплачен';

else

RETURN 'Этот месяц не оплачен';

end if;

end;

SQL запрос: **Таблица заявок, поданных пользователем** – запрос в базу данных для получения нерассмотренных заявок отправленных этим пользователем

select

j\_petitions.PETITION\_ID AS "Номер петиции",

j\_petitions.FILLING\_DATE AS "Дата оформления",

PARTY\_JOBS.JOB\_NAME AS "Должность",

j\_petitions.MONTHS\_COUNT || ' месяцев' AS "Время продления",

CASE j\_petitions.STATUS

WHEN 1 THEN 'Одобрено'

WHEN 2 THEN 'Отвергнуто'

ELSE 'На рассмотрении'

END

AS "Статус"

from JOIN\_PETITIONS j\_petitions

JOIN PARTY\_JOBS ON PARTY\_JOBS.JOB\_ID = j\_petitions.TARGET\_JOB

WHERE trim(j\_petitions.FACE) = TRIM(APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME'))

DynamicAction(PL/SQL): **HidePayFields** – отрабатывает при загрузке страницы, и скрывает поле оплаты членского взноса, если этот месяц оплачен

declare

p\_pays\_count integer;

card\_id PERSONS.PARTY\_CARD%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(card\_id IS NULL)then

return true;

end if;

Exception

When No\_Data\_Found Then

return true;

End;

select

count(PARTY\_PAYMENTS.PAY\_ID)

INTO

p\_pays\_count

FROM

PARTY\_PAYMENTS

JOIN PARTY\_CARDS ON PARTY\_CARDS.CARD\_ID = PARTY\_PAYMENTS.PARTY\_CARD

JOIN PARTY\_JOBS ON PARTY\_JOBS.JOB\_ID = PARTY\_CARDS.PARTY\_JOB

WHERE

PARTY\_PAYMENTS.YEAR = SUBSTR(TO\_CHAR(SYSDATE), 1, 4) AND

PARTY\_PAYMENTS.MONTH = SUBSTR(TO\_CHAR(SYSDATE), 6, 2) AND

PAY\_VALUE >= PARTY\_JOBS.PAY\_COUNT;

if(p\_pays\_count > 0)then

RETURN true;

else

RETURN false;

end if;

end;

DynamicAction(PL/SQL): **Filling** – при нажатии кнопки обозначающей подачу заявления на изменение или продление парт билета, эта программа проверяет введенные поля, и сохраняет данные о запросе в базу.

declare

begin

if(to\_number(:P1\_MONTHS\_COUNT)>=0

AND to\_number(:P1\_MONTHS\_COUNT) <= 60) then

INSERT INTO "JOIN\_PETITIONS" (

PETITION\_ID,

FACE,

FILLING\_DATE,

STATUS,

TARGET\_JOB,

MONTHS\_COUNT

) VALUES (

null,

trim(APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME')),

SYSTIMESTAMP,

0,

:P1\_JOB,

:P1\_MONTHS\_COUNT

);

return 'Заявка успешно отправлена';

else

return 'Количество месяцев должно быть в пределах от 0, до 60';

end if;

end;

DynamicAction(PL/SQL):**PayButonPress** – при нажатии кнопки оплатить членский взнос, программа проверяет введенные данные, а затем сохраняет данные об оплате в базу.

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_pay\_count PARTY\_JOBS.PAY\_COUNT%Type;

Begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return 'Парт билет не найден!';

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return 'Не найдена должность';

end if;

Select

p\_job.PAY\_COUNT

Into

p\_pay\_count

From PARTY\_JOBS p\_job

Where p\_job\_id = p\_job.JOB\_ID;

if(p\_pay\_count IS NULL)then

p\_pay\_count := 0;

end if;

Exception

When No\_Data\_Found Then

return 'Не найден парт билет';

End;

if(:P1\_PAY\_VALUE < p\_pay\_count)then

return 'Размер выплаты не может быть меньше чем размер членского взноса!';

end if;

if(:P1\_PAY\_FIELD\_CARD IS NULL)then

return 'Необходимо ввести номер карты для оплаты';

end if;

INSERT INTO "KR\_ADMIN"."PARTY\_PAYMENTS" (

PAY\_ID,

PARTY\_CARD,

MONTH,

YEAR,

PAY\_VALUE,

PAY\_DATE

) VALUES (

null,

p\_card\_id,

SUBSTR(TO\_CHAR(SYSDATE), 6, 2),

SUBSTR(TO\_CHAR(SYSDATE), 1, 4),

p\_pay\_count,

SYSTIMESTAMP

);

:P1\_PAY\_FIELD\_CARD := '';

return 'Оплата прошла успешно!';

end;

**Страница с новостями**

SQL запрос: **Таблица GlobalNewsPage** – запрос в базу данных для получения глобальных новостей

select

NEWS.PUBLISH\_DATE as "Дата публикации",

PERSONS.LAST\_NAME || PERSONS.NAME ||'('|| PERSONS.LOGIN\_MAIL || ')' as "Автор",

NEWS.TITLE as "Заголовок",

NEWS.TEXT as "Текст"

FROM NEWS

JOIN PARTY\_CARDS ON NEWS.AUTOR=PARTY\_CARDS.CARD\_ID

JOIN PERSONS ON PARTY\_CARDS.PERSON=PERSONS.LOGIN\_MAIL

SQL запрос: **Таблица Заявления** – запрос в базу данных для получения всех не рассмотренных заявлений пользователей, на изменение или продление парт билета

select

JOIN\_PETITIONS.PETITION\_ID as "Изменить",

JOIN\_PETITIONS.PETITION\_ID as "Номер заявления",

JOIN\_PETITIONS.FACE as "Подавшее лицо",

JOIN\_PETITIONS.FILLING\_DATE as "Дата подачи",

PARTY\_JOBS.JOB\_NAME as "На должность",

JOIN\_PETITIONS.MONTHS\_COUNT || ' месяцев' as "Продление на"

FROM JOIN\_PETITIONS

JOIN PARTY\_JOBS ON JOIN\_PETITIONS.TARGET\_JOB = PARTY\_JOBS.JOB\_ID

WHERE JOIN\_PETITIONS.STATUS = 0

SQL запрос: **Таблица партийные выплаты** – запрос в базу данных для получения сведений о всех членских взносах пользователей

select

PARTY\_PAYMENTS.PAY\_ID as "Уник. номер",

PERSONS.LOGIN\_MAIL || '(' || PERSONS.NAME || ' ' || PERSONS.MIDDLE\_NAME || ' ' || PERSONS.LAST\_NAME || ')' as "Владелец билета",

PARTY\_PAYMENTS.YEAR || '-' || PARTY\_PAYMENTS.MONTH as "Дата месяца",

PARTY\_PAYMENTS.PAY\_VALUE as "Сумма оплаты",

PARTY\_PAYMENTS.PAY\_DATE as "Дата оплаты"

FROM PARTY\_PAYMENTS

JOIN PERSONS ON (PARTY\_PAYMENTS.PARTY\_CARD = PERSONS.PARTY\_CARD)

DynamicAction(PL/SQL):**HideNews** – при отсутствии у пользователя прав на чтение новостей, программа скрывает блок «новости»

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_job\_access PARTY\_JOBS.ACCESS\_VIEW\_GLOBAL\_NEWS%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return true;

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return true;

end if;

Select

jobs.ACCESS\_VIEW\_GLOBAL\_NEWS

Into

p\_job\_access

From PARTY\_JOBS jobs

Where p\_job\_id = jobs.JOB\_ID;

if(p\_job\_access = 0)then

return true;

else

return false;

end if;

Exception

When No\_Data\_Found Then

return true;

End;

end;

DynamicAction(PL/SQL):**HideNotification** – если у пользователя есть разрешение на просмотр новостей партии, то программа скроет стоящее по умолчанию сообщение «у вас нет доступа к новостям»

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_job\_access PARTY\_JOBS.ACCESS\_VIEW\_GLOBAL\_NEWS%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return false;

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return false;

end if;

Select

jobs.ACCESS\_VIEW\_GLOBAL\_NEWS

Into

p\_job\_access

From PARTY\_JOBS jobs

Where p\_job\_id = jobs.JOB\_ID;

if(p\_job\_access = 0)then

return false;

else

return true;

end if;

Exception

When No\_Data\_Found Then

return false;

End;

end;

DynamicAction(PL/SQL):**HideCreateNews** – если у пользователя нет прав на создание собственных новостей, программа скроет блок отвечающий за это

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_job\_access PARTY\_JOBS.ACCESS\_EDIT\_OWN\_GLOBAL\_NEWS%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return true;

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return true;

end if;

Select

jobs.ACCESS\_EDIT\_OWN\_GLOBAL\_NEWS

Into

p\_job\_access

From PARTY\_JOBS jobs

Where p\_job\_id = jobs.JOB\_ID;

if(p\_job\_access = 0)then

return true;

else

return false;

end if;

Exception

When No\_Data\_Found Then

return true;

End;

end;

DynamicAction(PL/SQL): **HidePetitions** – если у пользователя нет прав на разбор петиций других пользователей, то программа скроет это сообщение

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_job\_access PARTY\_JOBS.ACCESS\_VIEW\_GLOBAL\_NEWS%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return true;

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return true;

end if;

Select

jobs.ACCESS\_APPROVAL\_OF\_PETITIONS

Into

p\_job\_access

From PARTY\_JOBS jobs

Where p\_job\_id = jobs.JOB\_ID;

if(p\_job\_access = 0)then

return true;

else

return false;

end if;

Exception

When No\_Data\_Found Then

return true;

End;

end;

DynamicAction(PL/SQL): **Create\_New** – отрабатывает при нажатии на кнопку «опубликовать новость»

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

access PARTY\_JOBS.ACCESS\_EDIT\_OWN\_GLOBAL\_NEWS%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return 'Не найден ваш парт. билет!';

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return 'Не найдена партийная должность!';

end if;

Select

p\_job.ACCESS\_EDIT\_OWN\_GLOBAL\_NEWS

Into

access

From PARTY\_JOBS p\_job

Where p\_job\_id = p\_job.JOB\_ID;

if(access = 0)then

return 'У вас недостаточно прав для публикации глобальных новостей';

else

if(:P2\_NEWS\_TEXT is not null) then

INSERT INTO "NEWS" (

AUTOR,

IS\_PERSONAL,

PUBLISH\_DATE,

TITLE,

TEXT

) VALUES (

p\_card\_id,

'0',

SYSTIMESTAMP,

:P2\_NEWS\_TITLE,

:P2\_NEWS\_TEXT

);

return 'Опубликовано в ' || SYSTIMESTAMP;

else

return 'Текст новости не может быть пустым!';

end if;

end if;

Exception

When No\_Data\_Found Then

return 'Ошибка публикации, одна из записей не найдена';

End;

end;

**Каталог пива**

SQL запрос: **Таблица** **BeerBrandsList** – запрос в базу данных для получения информации о марках пива хранящихся в базе данных.

select BRAND\_ID AS "Уник. номер",

BEER\_NAME AS "Название марки",

COLOR AS "Оттенок",

TASTE AS "Оттенки вкуса",

BOTTLE\_COST AS "Стоимость бут(₽)",

BEER\_COMMENT AS "Комментарий"

from BEER\_BRANDS

SQL запрос: **Таблица** **партийные заказы** – запрос в базу данных для получения информации о заказах пива, сделанных пользователями.

select BEER\_PARTY\_ORDERS.ORDER\_ID AS "Номер заказа",

BEER\_PARTY\_ORDERS.BEER\_BRAND AS "Номер бренда",

BEER\_BRANDS.BEER\_NAME AS "Название марки",

BEER\_PARTY\_ORDERS.COUNT AS "Количество бут.",

BEER\_PARTY\_ORDERS.ORDER\_DATE AS "Дата заказа",

BEER\_PARTY\_ORDERS.PURPOSE\_OF\_THE\_ORDER AS "Цель заказа",

BEER\_PARTY\_ORDERS.ALLOCATED\_MONEY AS "Выделенные сред.",

person1.LAST\_NAME || person1.NAME ||'('|| person1.LOGIN\_MAIL || ')' AS "Заказчик",

person1.LAST\_NAME || person1.NAME ||'('|| person1.LOGIN\_MAIL || ')' AS "Ответств. л.",

BEER\_PARTY\_ORDERS.SELLER AS "Продавец"

from BEER\_PARTY\_ORDERS

JOIN BEER\_BRANDS ON BEER\_PARTY\_ORDERS.BEER\_BRAND = BEER\_BRANDS.BRAND\_ID

JOIN PARTY\_CARDS card1 ON BEER\_PARTY\_ORDERS.CUSTOMER\_CARD = card1.CARD\_ID

JOIN PERSONS person1 ON card1.PERSON=person1.LOGIN\_MAIL

JOIN PARTY\_CARDS card2 ON BEER\_PARTY\_ORDERS.RESPONSIBLE\_PERSON\_CARD = card2.CARD\_ID

JOIN PERSONS person2 ON card2.PERSON=person2.LOGIN\_MAIL

DynamicAction(PL/SQL): **HideOrderCreate** – отрабатывает при загрузке страницы, в случае отсутствия у пользователя прав на создание партийного заказа, убирает блок заказов

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_job\_access PARTY\_JOBS.ACCESS\_CREATE\_ORDERS%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return true;

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return true;

end if;

Select

jobs.ACCESS\_CREATE\_ORDERS

Into

p\_job\_access

From PARTY\_JOBS jobs

Where p\_job\_id = jobs.JOB\_ID;

if(p\_job\_access = 0)then

return true;

else

return false;

end if;

Exception

When No\_Data\_Found Then

return true;

End;

end;

DynamicAction(PL/SQL): **HideOrders** – отрабатывает при загрузке страницы, в случае отсутствия у пользователя прав на просмотр партийных заказов, убирает блок просмотра заказов

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

p\_job\_access PARTY\_JOBS.ACCESS\_CREATE\_ORDERS%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return true;

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return true;

end if;

Select

jobs.ACCESS\_VIEW\_RESPONSIBLE\_ORDERS

Into

p\_job\_access

From PARTY\_JOBS jobs

Where p\_job\_id = jobs.JOB\_ID;

if(p\_job\_access = 0)then

return true;

else

return false;

end if;

Exception

When No\_Data\_Found Then

return true;

End;

end;

DynamicAction(PL/SQL): **CheckBottlesCost** – отрабатывает когда пользователь в партийном заказе изменяет количество или тип бутылок пива, и расчитывает общую стоимость заказа.

declare

p\_cost BEER\_BRANDS.BOTTLE\_COST%Type;

p\_count integer;

begin

Begin

Select

p\_brand.BOTTLE\_COST

Into

p\_cost

From BEER\_BRANDS p\_brand

Where TO\_NUMBER(:P3\_BRAND) = p\_brand.BRAND\_ID;

if(p\_cost IS NULL)then

return 'Ошибка выбора бренда пива!';

end if;

return ''||(p\_cost \* TO\_NUMBER(:P3\_BOTTLES\_COUNT));

Exception

When No\_Data\_Found Then

return 'Ошибка введенных значений!';

End;

end;

DynamicAction(PL/SQL): **CreateOrder** – отрабатывает при нажатии кнопки создать заказ, отправляет сведения о заказе в базу данных и изменяет статус заказа.

declare

p\_card\_id PERSONS.PARTY\_CARD%Type;

p\_job\_id PARTY\_CARDS.PARTY\_JOB%Type;

access PARTY\_JOBS.ACCESS\_CREATE\_ORDERS%Type;

p\_resp\_card\_id PERSONS.PARTY\_CARD%Type;

begin

Begin

Select

person.PARTY\_CARD

Into

p\_card\_id

From PERSONS person

Where APEX\_UTIL.GET\_SESSION\_STATE('SESSION\_USER\_NAME') = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return 'Не найден ваш парт. билет!';

end if;

Select

person.PARTY\_CARD

Into

p\_resp\_card\_id

From PERSONS person

Where :P3\_RESPONSIBLE\_PERSON = TRIM(person.LOGIN\_MAIL);

if(p\_card\_id IS NULL) then

return 'Не найден парт. билет ответственного за заказ человека!';

end if;

Select

p\_card.PARTY\_JOB

Into

p\_job\_id

From PARTY\_CARDS p\_card

Where p\_card\_id = p\_card.CARD\_ID;

if(p\_job\_id IS NULL)then

return 'Не найдена партийная должность!';

end if;

Select

p\_job.ACCESS\_CREATE\_ORDERS

Into

access

From PARTY\_JOBS p\_job

Where p\_job\_id = p\_job.JOB\_ID;

if(access = 0)then

return 'У вашей должности недостаточно прав для создания заказов';

else

if(

:P3\_BRAND IS NOT NULL AND

:P3\_BOTTLES\_COUNT IS NOT NULL AND

:P3\_ALLOCATED\_MONEY IS NOT NULL AND

:P3\_PURPOSE IS NOT NULL AND

:P3\_RESPONSIBLE\_PERSON IS NOT NULL AND

:P3\_SELLER IS NOT NULL

)then

INSERT INTO "BEER\_PARTY\_ORDERS" (

BEER\_BRAND,

COUNT,

ORDER\_DATE,

PURPOSE\_OF\_THE\_ORDER,

ALLOCATED\_MONEY,

CUSTOMER\_CARD,

RESPONSIBLE\_PERSON\_CARD,

SELLER

) VALUES (

:P3\_BRAND,

:P3\_BOTTLES\_COUNT,

SYSDATE,

:P3\_PURPOSE,

:P3\_ALLOCATED\_MONEY,

p\_card\_id,

p\_resp\_card\_id,

:P3\_SELLER

);

return 'Заказ добавлен в ' || SYSTIMESTAMP;

else

return 'Одно из полей пустое!';

end if;

end if;

Exception

When No\_Data\_Found Then

return 'Ошибка в заказе, одна из записей не найдена!';

End;

end;

**Страница обработки заявлений**

PL/SQL:**P4\_PETITION\_ID** – запрос получения номера петиции.

begin

return APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

end;

PL/SQL: **P4\_PETITION\_DATE** – запрос получения даты петиции.

Declare

temp\_p JOIN\_PETITIONS.FILLING\_DATE%TYPE;

begin

select

JOIN\_PETITIONS.FILLING\_DATE

INTO

temp\_p

FROM JOIN\_PETITIONS

WHERE JOIN\_PETITIONS.PETITION\_ID = APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

return temp\_p;

end;

PL/SQL: **P4\_FACE** – запрос получения почты пользователя подавшего петицию.

Declare

temp\_p JOIN\_PETITIONS.FACE%TYPE;

begin

select

JOIN\_PETITIONS.FACE

INTO

temp\_p

FROM JOIN\_PETITIONS

WHERE JOIN\_PETITIONS.PETITION\_ID = APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

return temp\_p;

end;

PL/SQL: **P4\_CURRENT\_UNTIL** – запрос получения срока годности текущего партийного билета.

Declare

face\_p JOIN\_PETITIONS.FACE%TYPE;

card\_id\_p PERSONS.PARTY\_CARD%TYPE;

valid\_p PARTY\_CARDS.VALID\_UNTIL%TYPE;

begin

-- получаем человека

select

JOIN\_PETITIONS.FACE

INTO

face\_p

FROM JOIN\_PETITIONS

WHERE JOIN\_PETITIONS.PETITION\_ID = APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

-- получаем id его билета

select

PERSONS.PARTY\_CARD

INTO

card\_id\_p

FROM PERSONS

WHERE PERSONS.LOGIN\_MAIL = face\_p;

-- если билет существует

if(card\_id\_p is not null)then

-- получаем срок годности

select

PARTY\_CARDS.VALID\_UNTIL

INTO

valid\_p

FROM PARTY\_CARDS

WHERE PARTY\_CARDS.CARD\_ID = card\_id\_p;

return valid\_p;

else

return '---';

end if;

end;

PL/SQL:**P4\_ADD\_UNTIL** – запрос получения времени продления партийного билета.

Declare

temp\_p JOIN\_PETITIONS.MONTHS\_COUNT%TYPE;

begin

select

JOIN\_PETITIONS.MONTHS\_COUNT

INTO

temp\_p

FROM JOIN\_PETITIONS

WHERE JOIN\_PETITIONS.PETITION\_ID = APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

return to\_char(temp\_p)||' месяцев';

end;

PL/SQL:**P4\_CURRENT\_JOB** – запрос получения текущей должности указанной в парт билете.

Declare

face\_p JOIN\_PETITIONS.FACE%TYPE;

card\_id\_p PERSONS.PARTY\_CARD%TYPE;

job\_id\_p PARTY\_CARDS.PARTY\_JOB%TYPE;

job\_name\_p PARTY\_JOBS.JOB\_NAME%TYPE;

begin

-- получаем человека

select

JOIN\_PETITIONS.FACE

INTO

face\_p

FROM JOIN\_PETITIONS

WHERE JOIN\_PETITIONS.PETITION\_ID = APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

-- получаем id его билета

select

PERSONS.PARTY\_CARD

INTO

card\_id\_p

FROM PERSONS

WHERE PERSONS.LOGIN\_MAIL = face\_p;

-- если билет существует

if(card\_id\_p is not null)then

-- получаем id работы

select

PARTY\_CARDS.PARTY\_JOB

INTO

job\_id\_p

FROM PARTY\_CARDS

WHERE PARTY\_CARDS.CARD\_ID = card\_id\_p;

-- получаем работу

select

PARTY\_JOBS.JOB\_NAME

INTO

job\_name\_p

FROM PARTY\_JOBS

WHERE PARTY\_JOBS.JOB\_ID = job\_id\_p;

return job\_name\_p;

else

return '---';

end if;

end;

PL/SQL: **P4\_NEW\_JOB** – запрос получения указанной в петиции должности.

Declare

temp\_p JOIN\_PETITIONS.TARGET\_JOB%TYPE;

job\_name\_p PARTY\_JOBS.JOB\_NAME%TYPE;

begin

-- id работы

select

JOIN\_PETITIONS.TARGET\_JOB

INTO

temp\_p

FROM JOIN\_PETITIONS

WHERE JOIN\_PETITIONS.PETITION\_ID = APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

-- получаем работу

select

PARTY\_JOBS.JOB\_NAME

INTO

job\_name\_p

FROM PARTY\_JOBS

WHERE PARTY\_JOBS.JOB\_ID = temp\_p;

return job\_name\_p;

end;

DynamicAction(PL/SQL): **CANCEL** – отрабатывает при нажатии кнопки «отклонить петицию», отправляет статус петиции в базу данных.

Declare

temp\_p JOIN\_PETITIONS.FILLING\_DATE%TYPE;

begin

UPDATE

"KR\_ADMIN"."JOIN\_PETITIONS"

SET

STATUS = '2'

WHERE

JOIN\_PETITIONS.PETITION\_ID = APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

end;

DynamicAction(PL/SQL): **ACCEPT** – отрабатывает при нажатии кнопки «принять петицию», отправляет статус петиции в базу данных.

Declare

job\_id\_p JOIN\_PETITIONS.TARGET\_JOB%TYPE;

month\_p JOIN\_PETITIONS.MONTHS\_COUNT%TYPE;

mail\_p PERSONS.LOGIN\_MAIL%TYPE;

card\_id\_p PERSONS.PARTY\_CARD%TYPE;

begin

-- ищем новую работу

select

JOIN\_PETITIONS.TARGET\_JOB

INTO

job\_id\_p

FROM JOIN\_PETITIONS

WHEREJOIN\_PETITIONS.PETITION\_ID= APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

-- ищем срок продления

select

JOIN\_PETITIONS.MONTHS\_COUNT

INTO

month\_p

FROM JOIN\_PETITIONS

WHEREJOIN\_PETITIONS.PETITION\_ID= APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

-- ищем парт. билет

select

PERSONS.LOGIN\_MAIL,

PERSONS.PARTY\_CARD

INTO

mail\_p,

card\_id\_p

FROM JOIN\_PETITIONS

JOIN PERSONS ON JOIN\_PETITIONS.FACE = PERSONS.LOGIN\_MAIL

WHERE JOIN\_PETITIONS.PETITION\_ID= APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

if(card\_id\_p is not null)then

-- если парт билет существует, просто меняем его данные

UPDATE

"KR\_ADMIN"."PARTY\_CARDS"

SET

PARTY\_JOB = job\_id\_p,

VALID\_UNTIL = ADD\_MONTHS(PARTY\_CARDS.VALID\_UNTIL, month\_p)

WHERE

PARTY\_CARDS.CARD\_ID = card\_id\_p;

else

-- если парт билет не существует, создаем его

INSERT INTO "PARTY\_CARDS" (

PERSON,

PARTY\_JOB,

FILLING\_DATE,

VALID\_UNTIL

) VALUES (

mail\_p,

job\_id\_p,

SYSTIMESTAMP,

ADD\_MONTHS(SYSTIMESTAMP, month\_p)

);

-- получаем id карты

select

PARTY\_CARDS.CARD\_ID

INTO

card\_id\_p

FROM PARTY\_CARDS

WHERE PARTY\_CARDS.PERSON = mail\_p;

-- записываем id пользователю

UPDATE

"KR\_ADMIN"."PERSONS"

SET

PARTY\_CARD = card\_id\_p

WHERE PERSONS.LOGIN\_MAIL = mail\_p;

end if;

UPDATE

"KR\_ADMIN"."JOIN\_PETITIONS"

SET

STATUS = '1'

WHERE

JOIN\_PETITIONS.PETITION\_ID= APEX\_UTIL.GET\_SESSION\_STATE('IN\_BUFFER\_1');

end;

**Глава 3. Результат**

**3.1 Интерфейс пользователя**

Разработанный графический интерфейс предназначен исключительно для пользователей и имеет 4 страницы:

- Страница входа и регистрации – позволяет пользователю создать аккаунт в системе и авторизоваться.

- Персональная страница – позволяет просматривать и редактировать сведения о текущем пользователе и его партийном билете

- Новости партии – позволяет получать информацию об актуальных новостях, а также, при наличии прав, изменять чужие партийные билеты и просматривать выплаты

- Каталог пива – содержит информацию обо всех марках пива известных партии любителей пива, при наличии необходимых прав, пользователь может на этой странице сделать заказ для партии.

На страницах используется много таблиц, которые пользователь может отфильтровать и настроить под вои конкретные нужды, что обеспечивает хорошую гибкость интерфейса.

Рассмотрим разработанный интерфейс.

Скриншоты страницы «Входа и регистрации» на рисунках 10 и 11

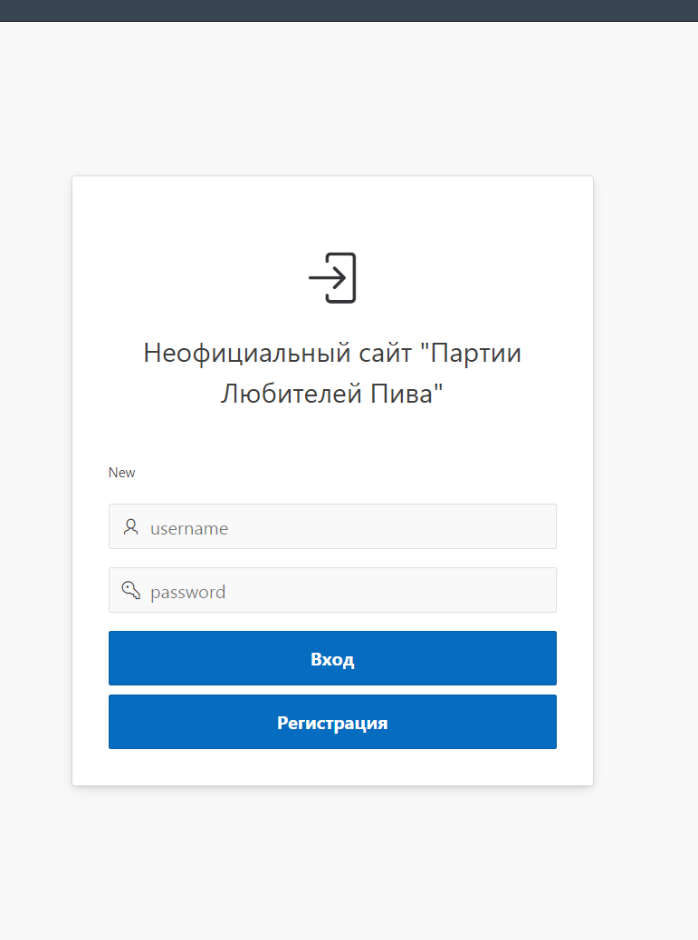


Рисунок 10 – Страница входа

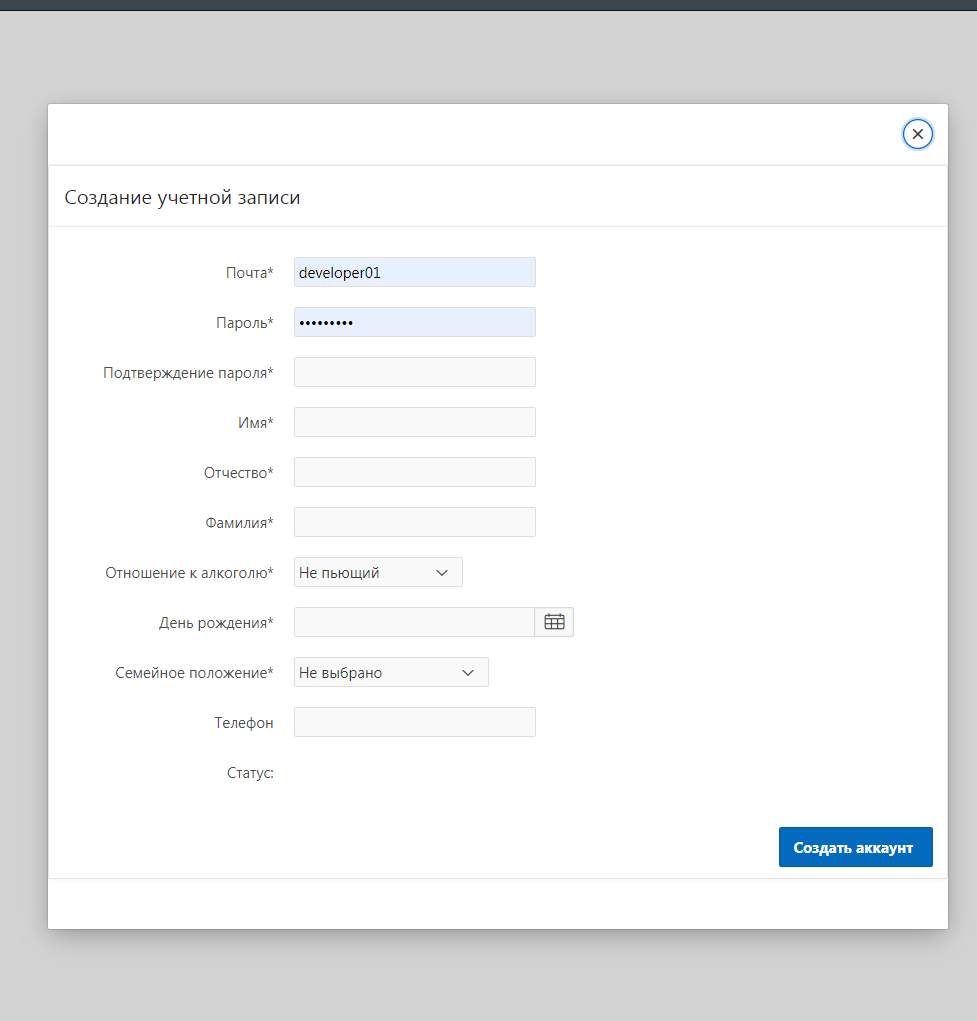


Рисунок 11 – Страница регистрации

Как только пользователь войдет в аккаунт, он будет перенаправлен на свою персональную страницу (рисунок 12). Здесь он может отредактировать свои данные, оплатить членские взносы, а также подать заявку на изменение или продление своего партийного билета.

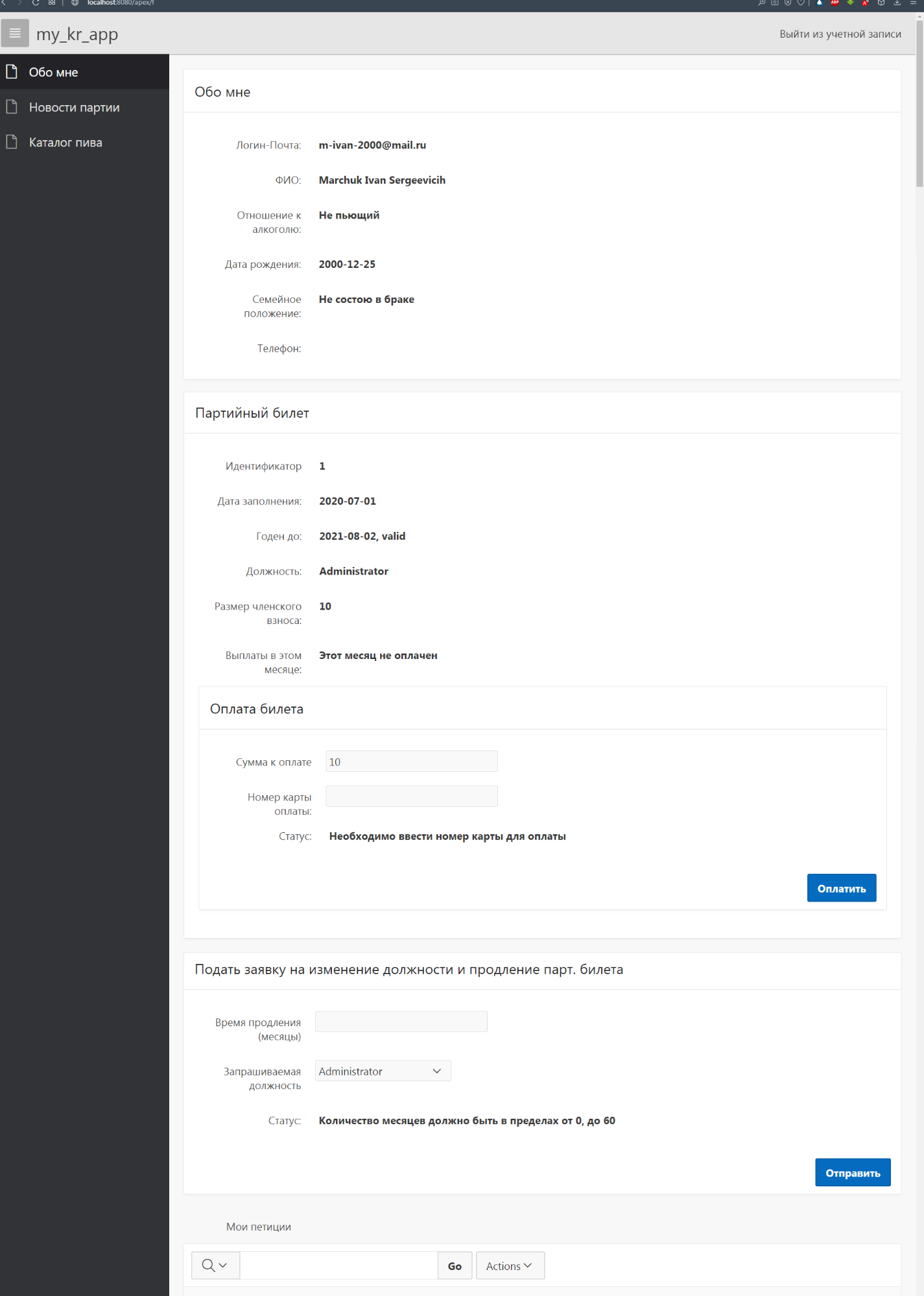


Рисунок 12 – «Персональная страница»

На странице новостей (рисунок 13) пользователь может читать актуальные новости партии, а также, при наличии прав, он может публиковать свои новости. Еще на этой странице пользователь может отвечать на заявки, на продление или изменение партийных билетов и просматривать членские взносы участников.

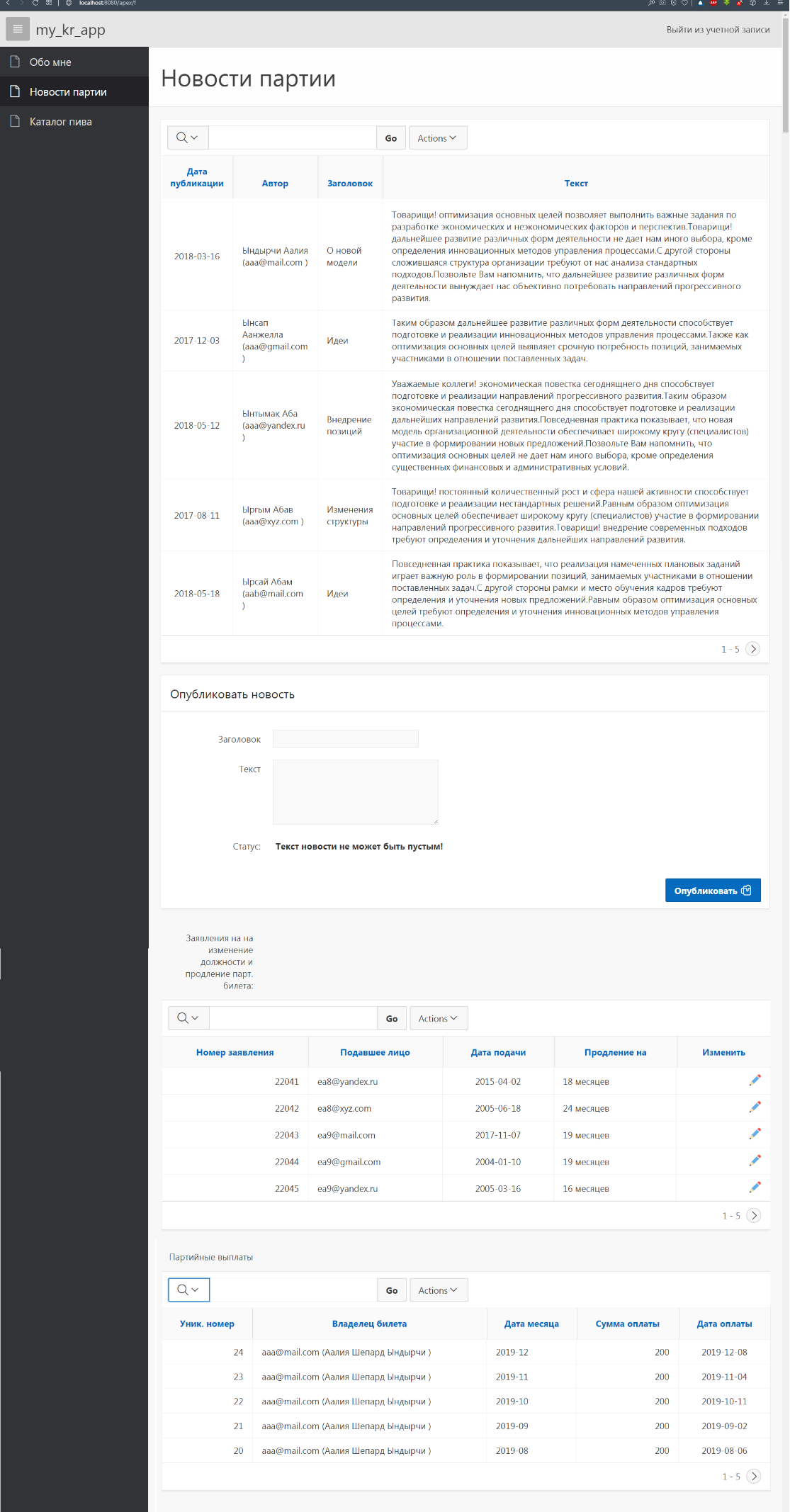


Рисунок 13 – «Новости партии»

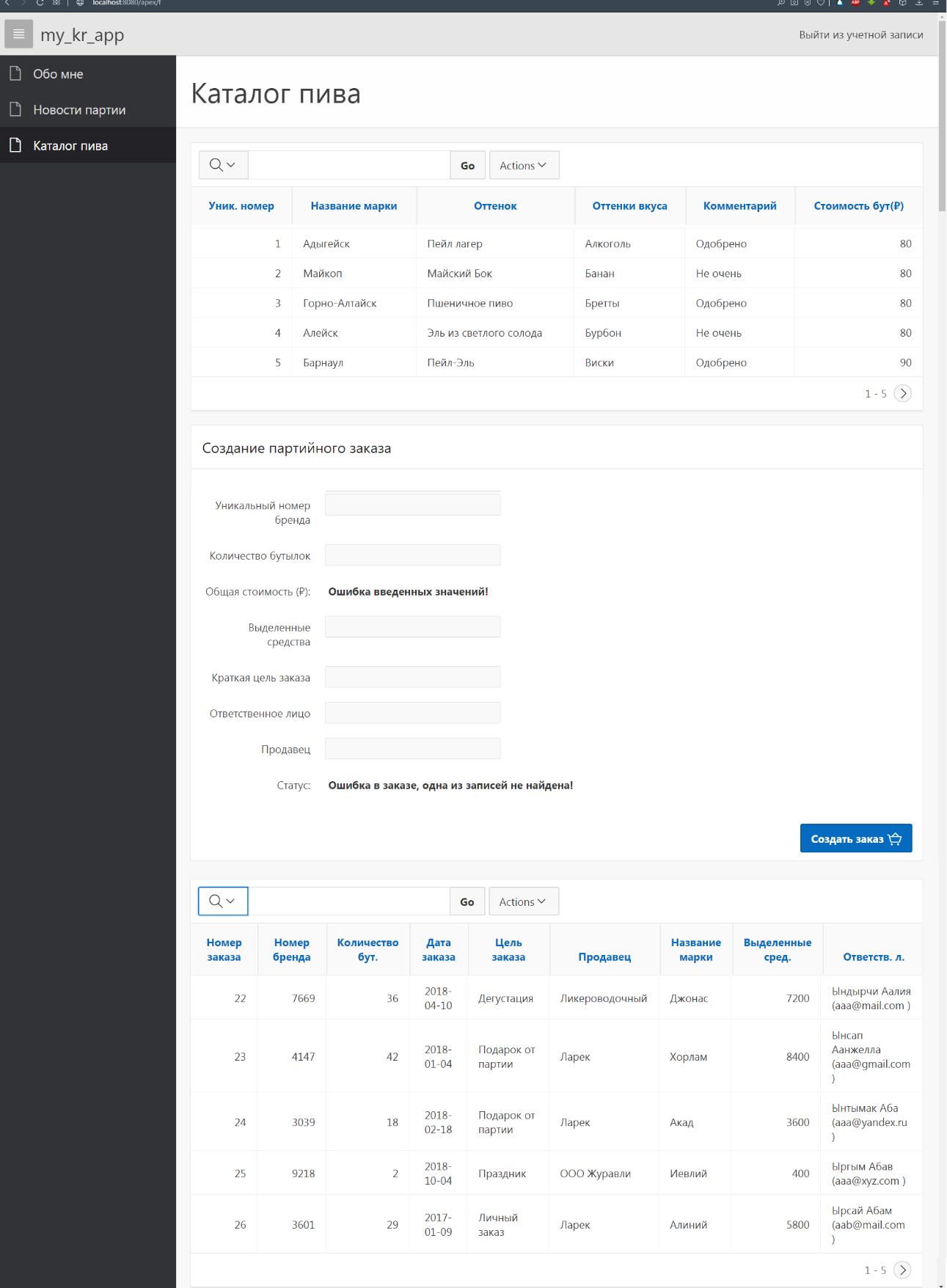


Рисунок 14 – «Каталог пива»

В каталоге пива (рисунок 14) пользователь может посмотреть информацию обо всех марках пива, известных партии. А также, при наличии необходимых прав, может сделать заказ для партийных нужд.

**Заключение**

Была разработана система, для партии любителей пива позволяющая вести учет партийных работников и их партийных билетов, публиковать общедоступные новости, делать партийные заказы и контролировать выплаты членских взносов. Система позволяет быстро и оперативно рассылать всем участникам партии самые важные новости. В качестве доработки можно добавить чаты, позволяющие пользователям общаться непосредственно друг с другом. А также, реализовать заказы не только пива но и любых других частых товаров из интернет магазинов сформированных в удобные списки. В таком случае, приложение сможет объединить в себе все необходимые функции. Возможна также разработка мобильного приложения (или десктопного) за счет использования простого API-интерфейса. С точки зрения графического интерфейса пользователя реализовано современное веб-приложение, оптимизированное как под ПК, так и под мобильные устройства и позволяющее вести рассылку партийных новостей, вести учет всех работников, осуществлять заказы необходимой для партии продукции и контролировать выплаты партийных взносов. Данное приложение призвано оптимизировать работу политических партий, сэкономить их время и ресурсы.

**Список источников**

1. Партия любителей пива (Россия) [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/Партия_любителей_пива_(Россия)> (дата обращения 16.05.2021)

2. Руководства Oracle APEX [Электронный ресурс] URL: <https://o7planning.org/ru/11003/oracle-apex> (дата обращения 16.05.2021)

3. Туториал по Oracle Application Express. Обзор IDE [Электронный ресурс] URL: <https://habr.com/ru/post/445128/> (дата обращения 16.05.2021)

4. Oracle PL/SQL учебник [Электронный ресурс] URL: <https://oracleplsql.ru/contents-oracle-plsql.html> (дата обращения 16.05.2021)

5. Oracle Database Online Documentation 12cRelease1(12.1) [Электронный ресурс] URL: <https://docs.oracle.com/database/121/nav/portal_4.htm> (дата обращения 16.05.2021)

**Приложение**

**Функции БД**

**1. Pkg\_Security** – функция авторизации пользователя

Procedure Process\_Login(pUserMail PERSONS.LOGIN\_MAIL%Type, pPassword PERSONS.PASSWORD%Type, p\_App\_Id Number);

create or replace Package Body Pkg\_Security Is

--------------------------------------

Procedure Process\_Login(pUserMail PERSONS.LOGIN\_MAIL%Type, pPassword PERSONS.PASSWORD%Type, p\_App\_Id Number) As

v\_Result Boolean := true;

v\_Password PERSONS.PASSWORD%Type;

v\_Email PERSONS.LOGIN\_MAIL%Type;

Begin

-- authenticate in data base

If pUserMail Is Null Or pPassword Is Null Then

-- Write to Session, Notification must enter a username and password

Apex\_Util.Set\_Session\_State('LOGIN\_MESSAGE'

,'Please enter Username and password.');

v\_Result := False;

End If;

Begin

Select TRIM(TRIM(person.PASSWORD)),

TRIM(person.LOGIN\_MAIL)

Into v\_Password,

v\_Email

From PERSONS person

Where TRIM(person.LOGIN\_MAIL) = pUserMail;

Exception

When No\_Data\_Found Then

-- Write to Session, User not found.

Apex\_Util.Set\_Session\_State('LOGIN\_MESSAGE'

,'User not found');

v\_Result := False;

End;

IF v\_Password <> pPassword THEN

-- Write to Session, Password incorrect.

Apex\_Util.Set\_Session\_State('LOGIN\_MESSAGE'

,'Password incorrect');

v\_Result := False;

End If;

If v\_Result = True Then -- authenticate is success

-- Write user information to Session.

Apex\_Util.Set\_Session\_State('SESSION\_USER\_NAME',pUserMail);

-- Redirect to Page 1 (Home Page).

Wwv\_Flow\_Custom\_Auth\_Std.Post\_Login(pUserMail,

pPassword,

v('APP\_SESSION'), -- p\_Session\_Id

p\_App\_Id || ':1'); -- p\_Flow\_page

Else

-- Login Failure, redirect to page 101 (Login Page).

Owa\_Util.Redirect\_Url('f?p=103:101:12345');

End If;

End;

End Pkg\_Security;

**Триггеры БД**

**1. BEER\_PARTY\_ORDERS\_id\_trg** – триггер инкрементирующий первичные ключи в таблице заказов пива

create or replace trigger BEER\_PARTY\_ORDERS\_id\_trg

before insert on BEER\_PARTY\_ORDERS

for each row

begin

if :new.ORDER\_ID is null then

select BEER\_PARTY\_ORDERS\_seq.nextval into :new.ORDER\_ID from dual;

end if;

end;

**2. bi\_apex\_access\_control** – триггер сохраняющий данные о пользователях которые работали с таблицей доступа

create or replace trigger bi\_apex\_access\_control

before insert or update on apex\_access\_control

for each row

begin

if inserting and :new.id is null then

select apex\_access\_control\_seq.nextval into :new.id from sys.dual;

end if;

if inserting then

:new.created\_by := v('USER');

:new.created\_on := sysdate;

end if;

if updating then

:new.updated\_by := v('USER');

:new.updated\_on := sysdate;

end if;

end;

**3. bi\_apex\_access\_setup** – триггер срабатывающий при начальной настройке доступа и инициализирующий начальные значения

create or replace trigger bi\_apex\_access\_setup

before insert or update on apex\_access\_setup

for each row

begin

if inserting and :new.id is null then

select apex\_access\_control\_seq.nextval into :new.id from sys.dual;

end if;

if :new.application\_id is null then

:new.application\_id := v('APP\_ID');

end if;

end;

**4.** **JOIN\_PETITIONS\_id\_trg** – триггер первичных ключей таблицы JOIN\_PETITIONS

create or replace trigger JOIN\_PETITIONS\_id\_trg

before insert on JOIN\_PETITIONS

for each row

begin

if :new.PETITION\_ID is null then

select JOIN\_PETITIONS\_seq.nextval into :new.PETITION\_ID from dual;

end if;

end;

**5.** **NEWS\_id\_trg** – триггер первичных ключей таблицы JOIN\_PETITIONS

create or replace trigger NEWS\_id\_trg

before insert on NEWS

for each row

begin

if :new.NEWS\_ID is null then

select NEWS\_seq.nextval into :new.NEWS\_ID from dual;

end if;

end;

**6.** **PARTY\_CARDS\_id\_trg** – триггер первичных ключей таблицы PARTY\_CARDS

create or replace trigger PARTY\_CARDS\_id\_trg

before insert on PARTY\_CARDS

for each row

begin

if :new.CARD\_ID is null then

select PARTY\_CARDS\_seq.nextval into :new.CARD\_ID from dual;

end if;

end;

**7.** **PARTY\_PAYMENTS\_id\_trg** – триггер первичных ключей таблицы PARTY\_PAYMENTS

create or replace trigger PARTY\_PAYMENTS\_id\_trg

before insert on PARTY\_PAYMENTS

for each row

begin

if :new.PAY\_ID is null then

select PARTY\_PAYMENTS\_seq.nextval into :new.PAY\_ID from dual;

end if;

end;