**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

**Отчёт по Домашнему заданию**

**“Проектирование веб-приложения с использованием фреймворка Django”**

Отчёт

(вид документа)

Листы А4

(вид носителя)

10

(количество листов)

|  |  |
| --- | --- |
| Исполнитель: | Студент группы ИУ5-54Б  Савельев Алексей Александрович |

Москва - 2021

**Цель домашнего задания:** изучение возможностей создания прототипа веб-приложения на основе базы данных с использованием фреймворка Django.

### Стандартное задание:

Создайте прототип веб-приложения с использованием фреймворка Django на основе базы данных, реализующий концепцию master/detail. Прототип должен содержать:

1. Две модели, связанные отношением один-ко-многим.
2. Стандартное средство администрирования Django позволяет редактировать данные моделей. Желательно настроить русификацию ввода и редактирования данных.
3. Веб-приложение формирует отчет в виде отдельного view/template, отчет выводит HTML-страницу, содержащую связанные данные из двух моделей.
4. Для верстки шаблонов используется фреймворк Bootstrap, или аналогичный фрейворк по желанию студента.

### Примечания:

1. Стандартный проект технопарка по Django приравнивается к стандартному варианту ДЗ.
2. По желанию студента для выполнения ДЗ может быть использован другой фреймворк, аналогичный Django, на любом языке программирования.

**Предметная область**

Предметной областью выступает тепа Поставщиков и Поставок. Реализованы связи 1-М, а также М-М (таблицы Поставщики-Партнеры). Так же на доп. баллы проект развернут на облачном сервере Heroku, где все еще функционирует.   
  
 Веб-приложение реализует авторизацию, и следовательно разделение функционала разных пользователей: админа и простого юзера. При входе на сайт пользователь попадает на страницу авторизации (так же реализована регистрация).   
 На главной странице оба пользователя могут начать взаимодействие с тремя таблицами, рядовой пользователь может лишь читать содержимое таблицы, а админ может удалять-добавлять-редактировать содержимое.   
 В панели навигации размещены кнопки, ведущие на главную страницу, и на страницу с отчетами, где пользователь может получить информацию по связанным таблицам. В одном из отчетов размещена js-диаграмма.

**Самые важные файлы:**

**Urls.py**

from django.urls import path  
from . import views  
  
  
urlpatterns = [  
 path('', views.loginPage, name='login'),  
 path('register/', views.regPage, name='register'),  
 path('logout/', views.logoutPage, name='logout'),  
  
 path('index/', views.index, name='home'),  
 path('details/<int:tab\_index>/', views.details, name='details'),  
 path(r'update/<int:tab\_index>/<int:pk>$', views.update, name='update'),  
 path(r'delete/<int:tab\_index>/<int:pk>$', views.delete, name='delete'),  
 path('review/', views.review, name='review'),  
 path('review/<int:tab\_index>/', views.review\_details, name='review\_details'),  
]

**Models.py**

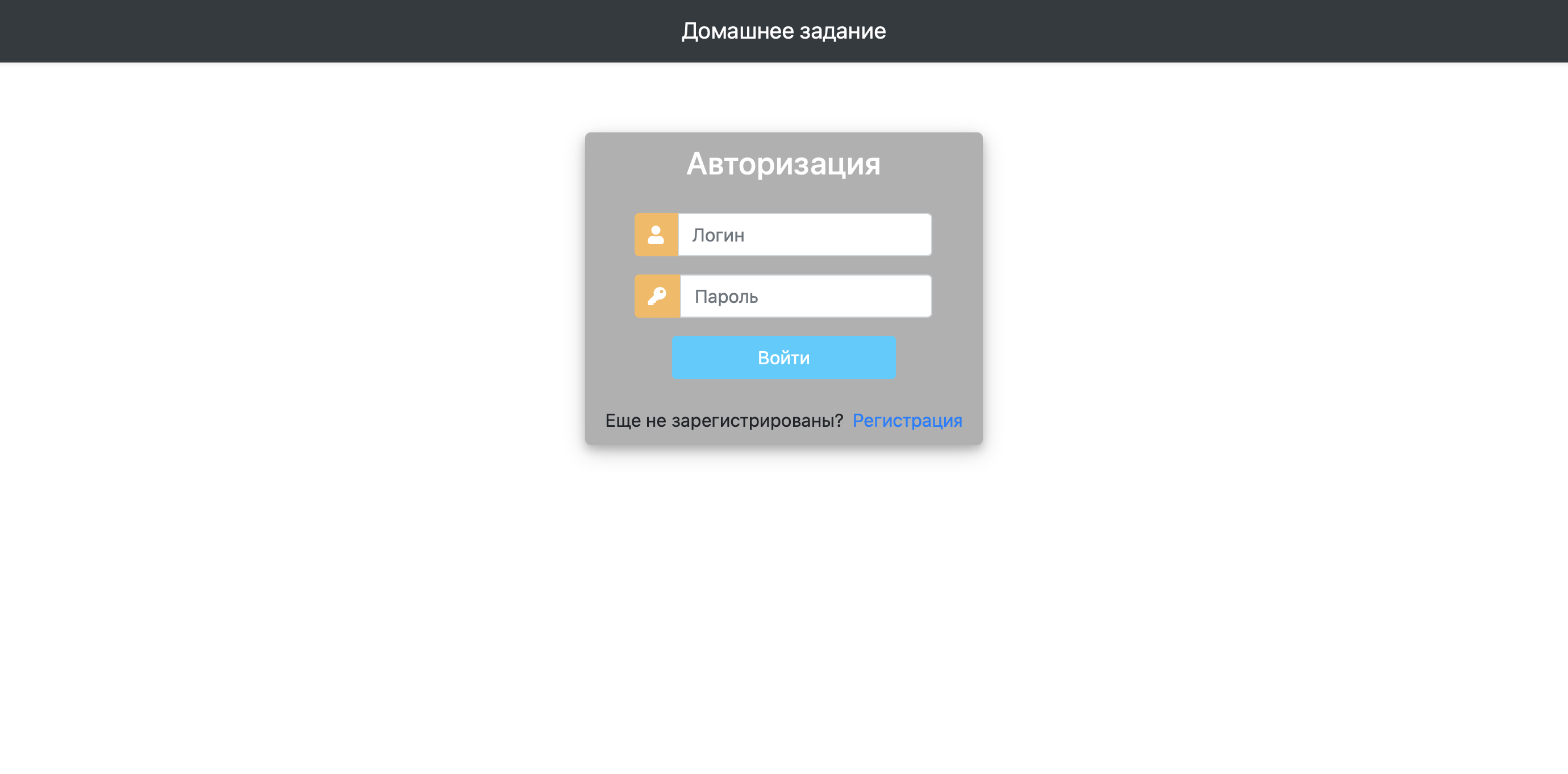
from django.db import models  
  
class Partner(models.Model):  
 name = models.CharField('Название', max\_length=50)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class Provider(models.Model):  
 name = models.CharField('Название', max\_length=50)  
 partners = models.ManyToManyField(Partner, blank=True, through='Partnership', through\_fields=('provider', 'partner'))  
 address = models.CharField('Адрес', max\_length=100)  
 phone = models.IntegerField('Телефон')  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class Partnership(models.Model):  
 provider = models.ForeignKey(Provider, on\_delete=models.CASCADE)  
 partner = models.ForeignKey(Partner, on\_delete=models.CASCADE)  
 designation = models.CharField('Цель партнерства', max\_length=200)  
  
 def \_\_str\_\_(self):  
 return str(self.provider.name + " - " + self.partner.name)  
  
class Product(models.Model):  
 product = models.CharField('Название продукта', max\_length=100)  
 expire\_time = models.IntegerField('Годен дней')  
  
 def \_\_str\_\_(self):  
 return self.product  
  
class Waybill(models.Model):  
 provider = models.ForeignKey(Provider, on\_delete=models.CASCADE)  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 quantity = models.IntegerField('Количество')  
 destination = models.CharField('Пункт назначения', max\_length=150)  
 date = models.DateField('Дата')  
  
 def \_\_str\_\_(self):  
 return str(self.provider.name + " - " + self.product.product)  
  
class Tables(models.Model):  
 name = models.CharField('Название таблицы', max\_length=100)  
 index = models.IntegerField('Индекс', default=1)  
 description = models.CharField('Описание', max\_length=300, default="")  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class TestProvider(models.Model):  
 name = models.CharField('Название', max\_length=50)  
 partners = models.ManyToManyField(Partner, blank=True)

**Views.py**

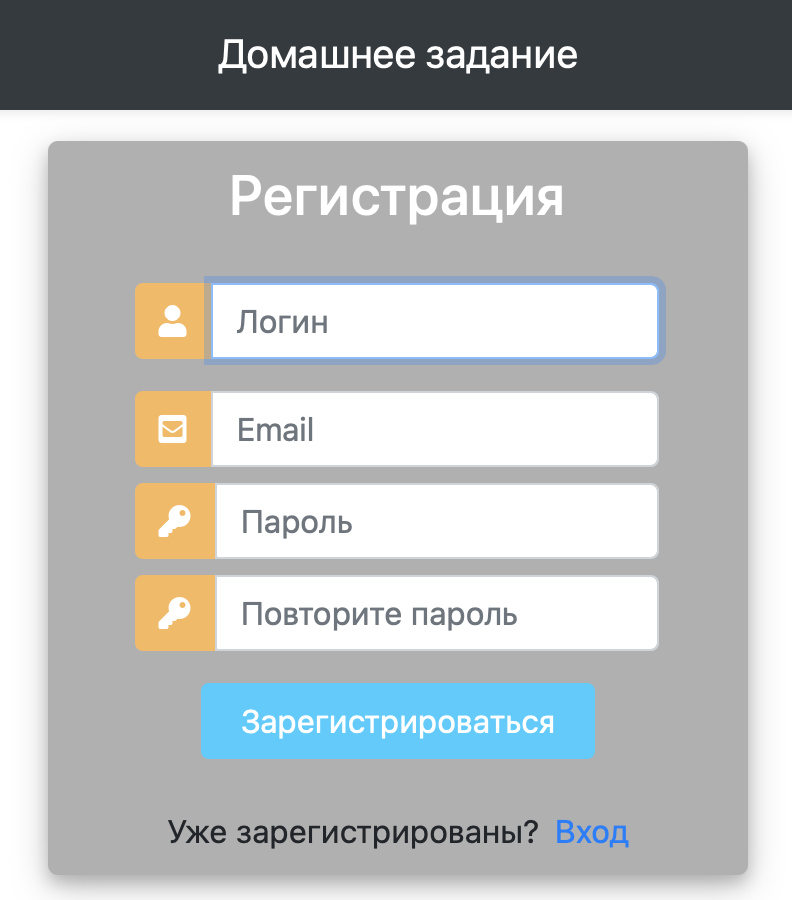
from django.shortcuts import get\_object\_or\_404, render, redirect  
from django.views.generic.edit import FormView  
from django.contrib.auth.forms import UserCreationForm  
from django.contrib.auth import authenticate, login, logout  
from django.urls import reverse  
  
from django.contrib import messages  
  
  
  
from .models import \*  
from .forms import CreateUserForm, PartnerForm, ProviderForm, ProductForm  
  
def index(request):  
 table\_list = Tables.objects.all()  
 context = {'title':'ДЗ', 'header':'Таблицы', 'table\_list':table\_list}  
 return render(request, 'DZApp/index.html', context)  
  
  
def details(request, tab\_index):  
 print(f"\033[34mtab index is: {tab\_index}\033[0m")  
 header = ""  
  
 if request.method == 'POST':  
 if tab\_index == 1:  
 form = PartnerForm(request.POST)  
 elif tab\_index == 3:  
 form = ProviderForm(request.POST)  
 elif tab\_index == 4:  
 form = ProductForm(request.POST)  
 if form.is\_valid():  
 form.save()  
  
 if tab\_index == 1:  
 table\_data = Partner.objects.all()  
 form = PartnerForm()  
 header = "Партнеры"  
 elif tab\_index == 3:  
 table\_data = Provider.objects.all()  
 form = ProviderForm()  
 header = "Поставщики"  
 elif tab\_index == 4:  
 table\_data = Product.objects.all()  
 form = ProductForm()  
 header = "Продукты"  
  
 if request.user.is\_authenticated:  
 username = request.user.username  
 else:  
 username = ""  
  
 return render(request, 'DZApp/details.html', {'title': 'Детали', 'header':header, 'table\_data':table\_data,  
 'tab\_index':tab\_index, 'form':form, 'username':username})  
  
def update(request, tab\_index, pk):  
  
 if tab\_index == 1:  
 to\_edit = Partner.objects.get(pk=pk)  
 form = PartnerForm(instance=to\_edit)  
 header = "Партнеры"  
 elif tab\_index == 3:  
 to\_edit = Provider.objects.get(pk=pk)  
 form = ProviderForm(instance=to\_edit)  
 header = "Поставщики"  
 elif tab\_index == 4:  
 to\_edit = Product.objects.get(pk=pk)  
 form = ProductForm(instance=to\_edit)  
 header = "Продукты"  
  
 if request.method == 'POST':  
 if tab\_index == 1:  
 form = PartnerForm(request.POST,instance=to\_edit)  
 elif tab\_index == 3:  
 form = ProviderForm(request.POST,instance=to\_edit)  
 elif tab\_index == 4:  
 form = ProductForm(request.POST,instance=to\_edit)  
 if form.is\_valid():  
 form.save()  
 return redirect('details', tab\_index)  
  
 return render(request, 'DZApp/details.html', {'header':header, 'update':True, 'to\_edit':to\_edit, 'form':form})  
  
def delete(request, tab\_index, pk):  
 to\_delete = None  
 if tab\_index == 1:  
 to\_delete = Partner.objects.get(pk=pk)  
 elif tab\_index == 3:  
 to\_delete = Provider.objects.get(pk=pk)  
 elif tab\_index == 4:  
 to\_delete = Product.objects.get(pk=pk)  
  
 to\_delete.delete()  
  
 return redirect('details', tab\_index)  
  
def review(request):  
 table\_list = Tables.objects.all()  
 context = {'title':'Отчет', 'header':'Отчет', 'table\_list':table\_list}  
 return render(request, 'DZApp/review.html', context)  
  
def review\_details(request, tab\_index):  
  
 if tab\_index == 2:  
 table\_data = Partnership.objects.all()  
 header = "Партнерство"  
 elif tab\_index == 4:  
 table\_data = Product.objects.all()  
 header = "Продукты"  
 elif tab\_index == 5:  
 table\_data = Waybill.objects.all().select\_related('provider', 'product')  
 header = "Поставки"  
  
 return render(request, 'DZApp/review\_details.html', {'title': 'Отчет', 'header': header, 'table\_data':table\_data,  
 'tab\_index':tab\_index})  
  
def loginPage(request):  
  
 if request.user.is\_authenticated:  
 return redirect('home')  
  
 if request.method == 'POST':  
 username = request.POST.get('username')  
 password = request.POST.get('password')  
  
 user = authenticate(request, username=username, password=password)  
  
 if user is not None:  
 login(request, user)  
 return redirect('home')  
 else:  
 messages.info(request, "Имя пользователя или пароль введены не верно")  
  
 context = {'title':'Авторизация'}  
 return render(request, 'DZApp/auth/login.html', context)  
  
def regPage(request):  
 form = CreateUserForm()  
  
 if request.method == 'POST':  
 form = CreateUserForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 user = form.cleaned\_data.get('username')  
 messages.success(request, "Пользователь " + user + " успешно зарегистрирован")  
  
 return redirect('login')  
  
 context = {'title':'Регистрация', 'form':form}  
 return render(request, 'DZApp/auth/registration.html', context)  
  
def logoutPage(request):  
 logout(request)  
 return redirect('login')

**Экранные формы сайта**

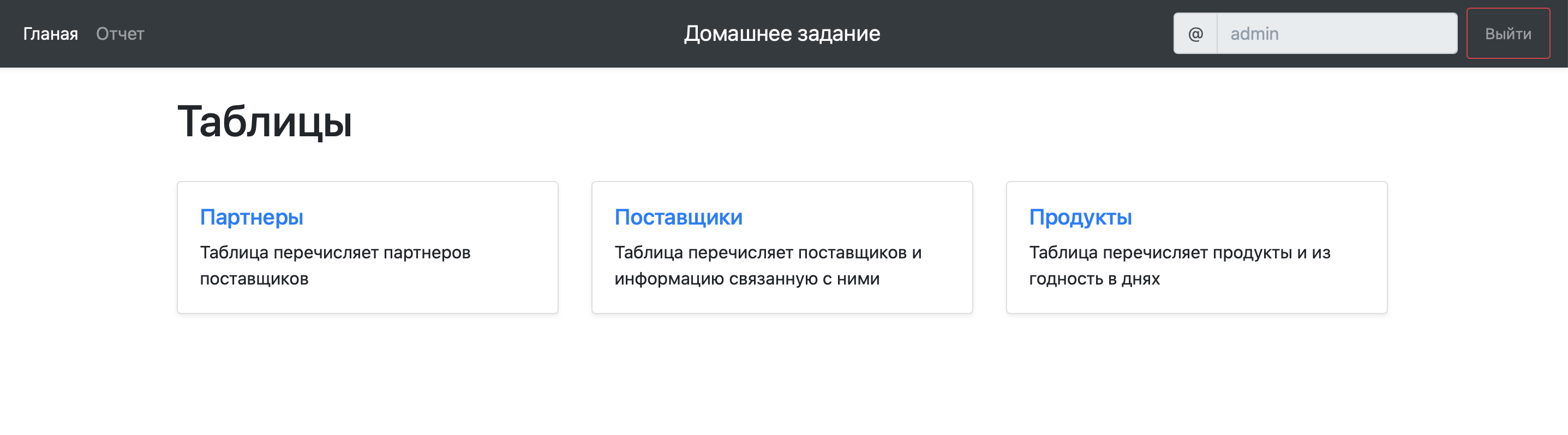
Заходя на сайт попадаем на страницу авторизации



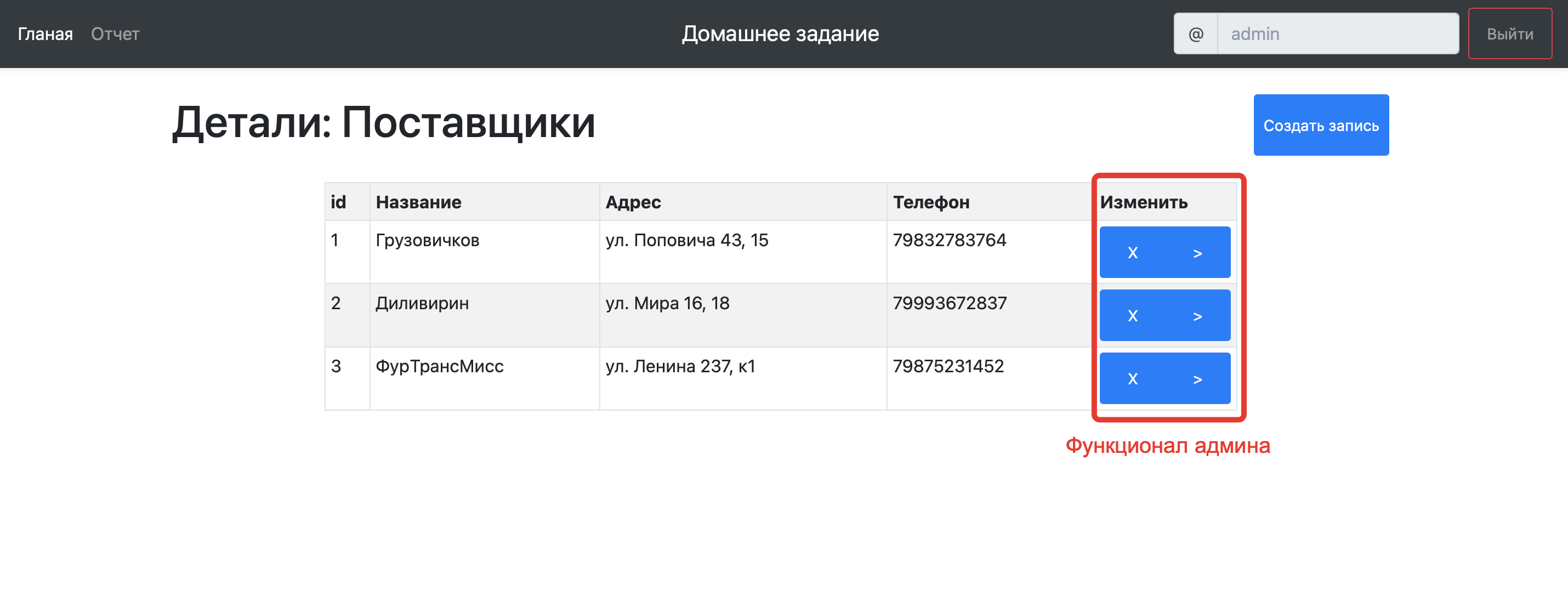
Так же предусмотрена форма регистрации новых пользователей



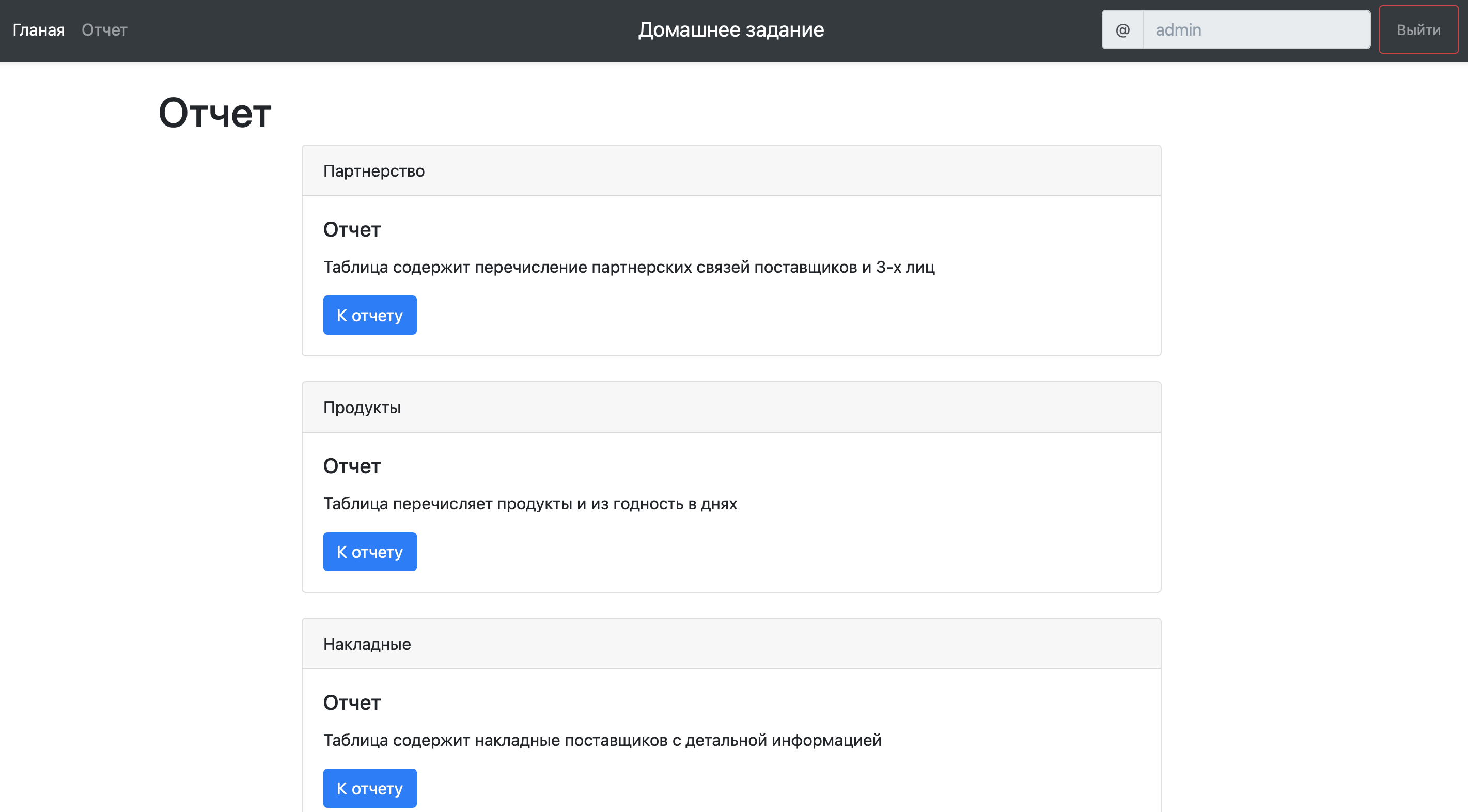
Авторизовавшись, попадаем на страницу с таблицами, и нажав на определённой кнопку мы переходим к определенной таблице, данные которой выводятся на странице браузера.



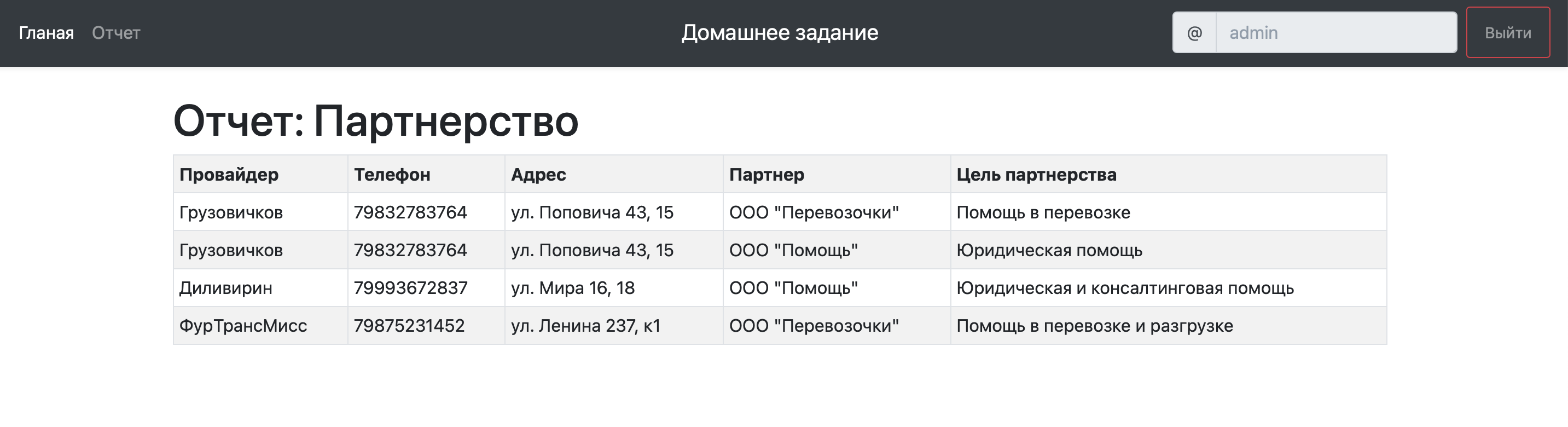
Предусмотрен функционал редактирования данных, их добавления или удаления записей, но только у пользователей, имеющих права администратора.



В панели навигации присутствует кнопки для возврата на главную страницу сайта, кнопка деавторизации и кнопка перехода к странице с отчетами.



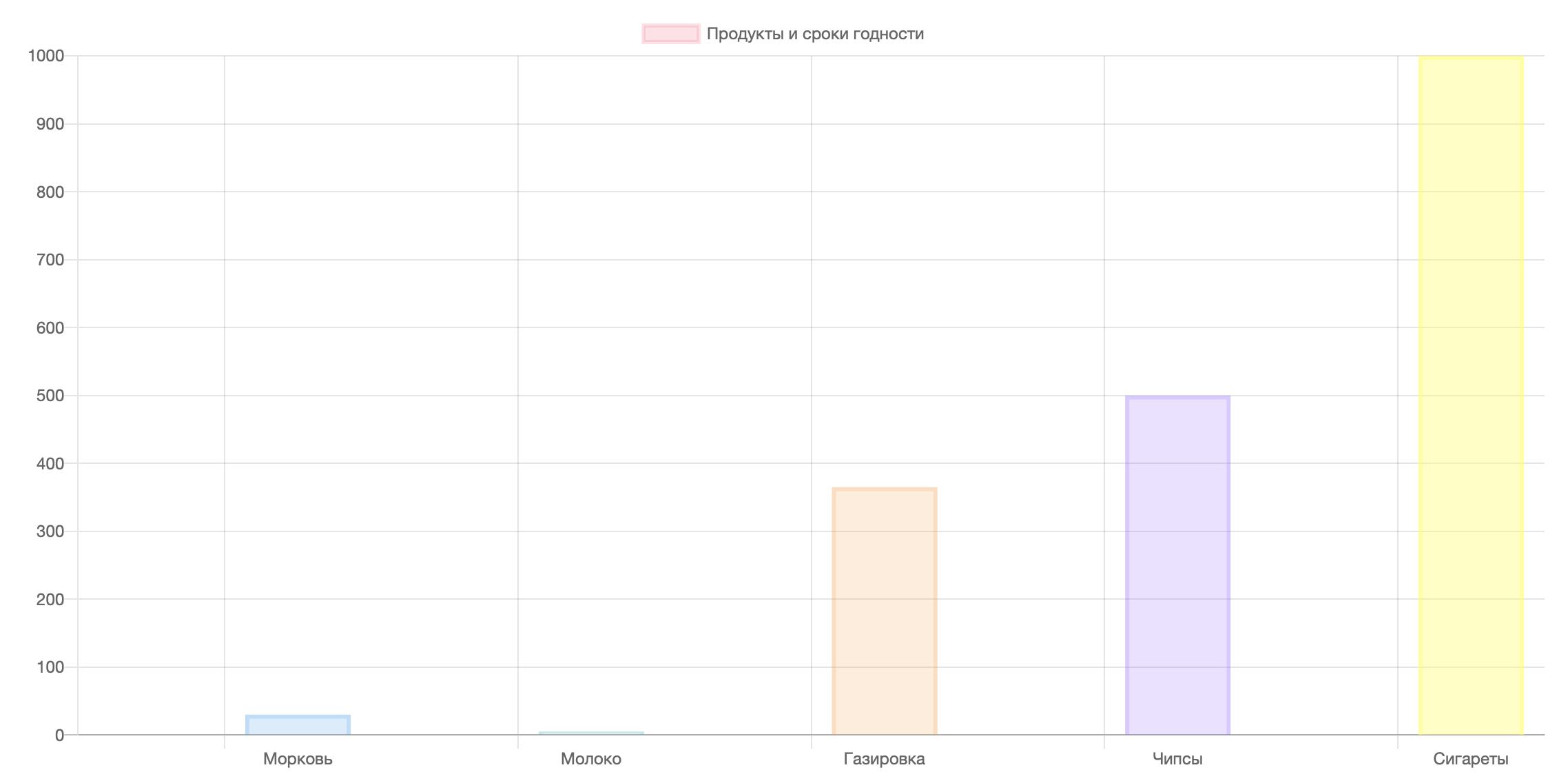
Отчет по партнёрству:



Отчет по продуктам:



Так же на странице отчета по продуктам выводится столбчатая диаграмма по срокам годности.



Отчет по накладным:



**Вывод:** в результате выполнения домашнего задания была выбрана предметная область, использованная мною ранее в лабораторном практикуме и курсовой работе по БД по теме «Магазин», а именно та ее часть, включающая поставки продукции и была реализована некая система управления базами данных, в виде веб-приложения, написанного на языке Python, и базирующее на фреймворке Django.