REPORT

MILESTONE 4

Zhassulan & akylbek

2021

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

So, here we started to create our database.

1. Administrator
   1. name - char
   2. login - char
   3. password - text

1. Customer
   1. name  - char
   2. order (FK)
   3. payment (FK)
   4. waiter (FK)

1. Order
   1. type - char
   2. price - int
   3. dishes(FK)

1. Payment
   1. price - int
   2. type - char

1. Waiter
   1. name - char
   2. phone number - char

1. Dishes
   1. name - char
   2. price - int
   3. ingredients - text
   4. cooking time  - int
   5. picture - image

1) ER

Изображение выглядит как текст, парковка

Автоматически созданное описание

2) Model to manipulate tables in DB

#models.py:

from django.conf import settings

from django.db import models

from django.utils import timezone

class Administator(models.Model):

name = models.CharField("Имя", max\_length=30)

login = models.CharField("Логин", max\_length=50)

password = models.TextField("Пароль")

def \_\_str\_\_(self):

return self.name

class Meta:

verbose\_name = "Администратор"

verbose\_name\_plural = "Администраторы"

class Dishes(models.Model):

name = models.CharField("Имя", max\_length=50)

price = models.IntegerField("Стоимость")

ingredients = models.TextField("Пароль")

cooking\_time = models.IntegerField("Стоимость")

picture = models.ImageField("Изображение")

def \_\_str\_\_(self):

return self.name

class Meta:

verbose\_name = "Блюда"

verbose\_name\_plural = "Блюда"

class Order(models.Model):

type = models.CharField("Вид заказа", max\_length=50)

price = models.IntegerField("Стоимость")

dishes = models.ForeignKey(Dishes, on\_delete=models.CASCADE,)

# dishes = models.ManyToManyField(Dishes, verbose\_name="Блюда", related\_name="dishes\_name")

def \_\_str\_\_(self):

return self.type

class Meta:

verbose\_name = "Заказ"

verbose\_name\_plural = "Заказы"

class Payment(models.Model):

price = models.IntegerField("Стоимость")

type = models.CharField("Способ оплаты", max\_length=50)

def \_\_str\_\_(self):

return self.type

class Meta:

verbose\_name = "Оплата"

verbose\_name\_plural = "Оплаты"

class Waiter(models.Model):

name = models.CharField("Имя", max\_length=50)

phone = models.CharField("Номер телефона", max\_length=20)

def \_\_str\_\_(self):

return self.name

class Meta:

verbose\_name = "Официант"

verbose\_name\_plural = "Официанты"

class Customer(models.Model):

name = models.CharField("Имя", max\_length=50)

order = models.ForeignKey(Order, on\_delete=models.CASCADE,)

# order = models.ManyToManyField(Order, verbose\_name="Заказ", related\_name="order\_name")

payment = models.ForeignKey(Payment, on\_delete=models.CASCADE,)

# payment = models.ManyToManyField(Payment, verbose\_name="Оплата", related\_name="payment\_type")

waiter = models.ForeignKey(Waiter, on\_delete=models.CASCADE,)

# waiter = models.ManyToManyField(Waiter, verbose\_name="Официант", related\_name="waiter\_name")

def \_\_str\_\_(self):

return self.name

class Meta:

verbose\_name = "Заказщик"

verbose\_name\_plural = "Заказщики"

#admin.py:

from django.contrib import admin

from .models import Administator, Dishes, Order, Payment, Waiter, Customer

admin.site.register(Administator)

admin.site.register(Dishes)

admin.site.register(Order)

admin.site.register(Payment)

admin.site.register(Waiter)

admin.site.register(Customer)

4) Select data from database and display ++

#views.py:

from django.shortcuts import render

from .models import Administator, Dishes, Order, Payment, Waiter, Customer

def db\_home(request):

administator = Administator.objects.all()

# administator = Administator.objects.order\_by('name')

# administator = Administator.objects.order\_by('-name')

# administator = Administator.objects.order\_by('name') #[:1] - ограничение исключительно 1 запись

dishes = Dishes.objects.all()

order = Order.objects.all()

payment = Payment.objects.all()

waiter = Waiter.objects.all()

customer = Customer.objects.all()

return render(request, 'db/db\_home.html', {

'administator': administator,

'dishes': dishes,

'order': order,

'payment': payment,

'waiter': waiter,

'customer': customer})

#db\_home.html:

{% if administator %}

{% for el in administator %}

<div class="alert alert-warning">

<h3>{{ el.name }}</h3>

<h6>{{ el.login }}</h6>

<h6>{{ el.password }}</h6>

</div>

{% endfor %}

{% else %}

<p>У нас нет записей!</p>

{% endif %}