Выполнила: Писукова Анастасия Вадимовна, 101 группа.

**Отчет по практической работе № 19**

**«Программная реализация системы».**

*Время выполнения – 14 часов*

**Тема:** «Программная реализация системы»

**Цель урока**:выполнение работы над реализацией информационной системы.

**Задание для выполнения**:

1. Выполните программную реализацию вашей информационной системы в соответствие с принятыми ранее проектными решения и концепцией решения.
2. С помощью команды startproject создадим проект Manic в среде разработки PyCharm на Python. Создадим приложения main для отображения основного сайта для клиента, и user, для отображения корпоративной системы сотрудников компании (см. Рисунок 1).

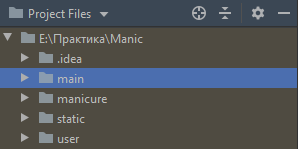


Рисунок 1 – Создание проекта и его приложений

1. Создадим элементы сайта для клиента – навигационную панель, главную страницу с основной информацией о компании, страницы новостей компании, оформления заказа, окончания оформления заказа, акций салона, прейскуранта цен, портфолио с фотографиями работ салона, отзывов клиентов (см. Рисунок 2). Также, добавим станицы в приложение user, такие как список услуг в виде таблицы, список заявок в виде таблицы, авторизация сотрудников, формы для создания заявки клиента, окончания создания заказа, добавления новой услуги, окончания добавления услуги, редактирования заявки, редактирования услуги (см. Рисунок 3). Оформим дизайн страниц согласно созданным ранее прототипам интерфейса. Создадим папку static и загрузим в нее изображения, которые будут использоваться на сайте.

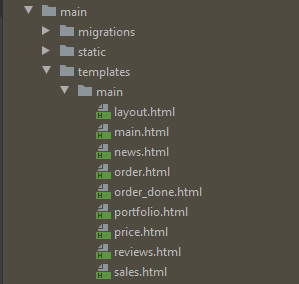


Рисунок 2 – Создание элементов приложения main

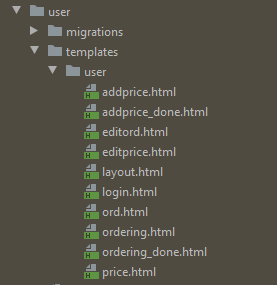


Рисунок 3 – Создание элементов приложения user

1. Исходя из базы данных, спроектированной ранее, создадим такие модели, как услуги, отзывы, заказы, а также дополнительную модель новостей (см. Рисунок 4):

class PriceList(models.Model):

name = models.CharField(max\_length=50)

price = models.CharField(max\_length=50)

picture = models.ImageField(upload\_to='static/main')

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

return f'{self.name.title()}'

class News(models.Model):

new = models.CharField(max\_length=50)

picture = models.ImageField(upload\_to='static/main')

text = models.TextField()

data = models.DateTimeField(auto\_now=True, blank = True)

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

return f'{self.new.title()}'

class Reviews(models.Model):

name = models.CharField(max\_length=50)

email = models.CharField(max\_length=50)

pricelist = models.ForeignKey(PriceList, related\_name='reviewpricelist', on\_delete=models.CASCADE, default="1")

text = models.TextField()

phone = models.CharField(max\_length=50)

data = models.DateTimeField(auto\_now=True)

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

return f'{self.name.title()}'

class Order(models.Model):

fullname = models.CharField(max\_length=50)

pricelist = models.ForeignKey(PriceList, related\_name='orderpricelist', on\_delete=models.CASCADE, default="1")

phone = models.CharField(max\_length=50)

data = models.DateTimeField(auto\_now=False)

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

return f'{self.fullname.title()}'

Произведем миграции с помощью команд makemigrations и migrate, создадим суперпользователя. Добавим модели в административную панель (см. Рисунок 5.1). В административной панели создадим новых пользователей admin и master, разграничим их возможности и зададим пароли для входа в систему (см. Рисунок 5.2).

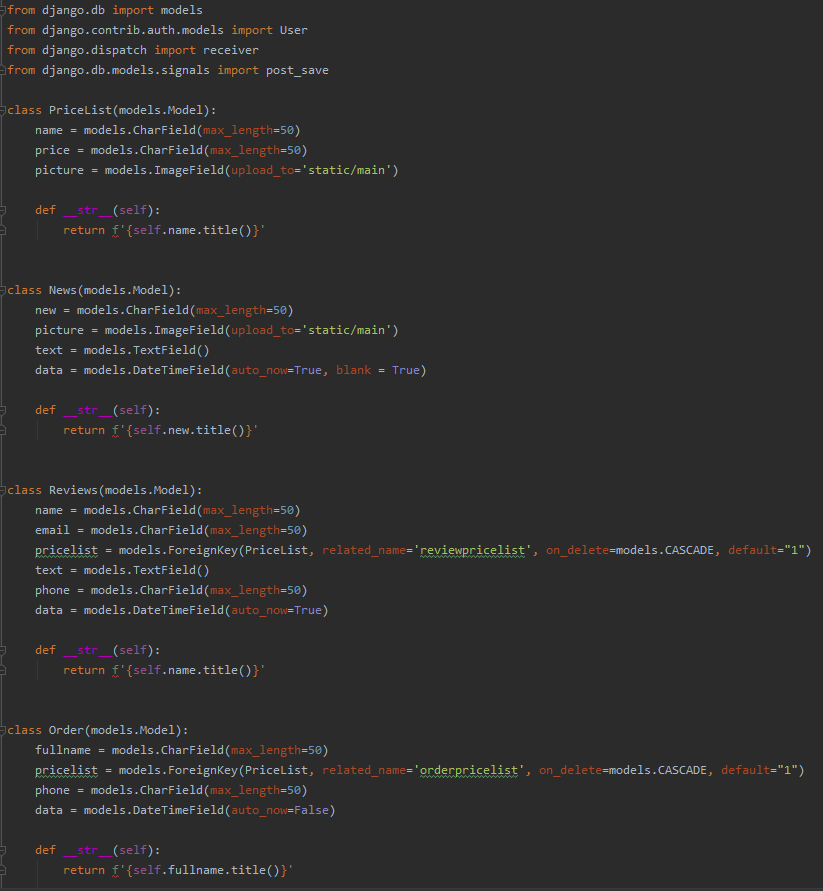


Рисунок 4 – Создание моделей

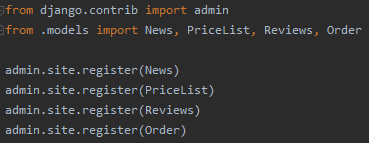


Рисунок 5.1 – Добавление моделей в административную панель



Рисунок 5.2 – Добавление пользователей

1. На основе созданных моделей в приложении main напишем форму создания заказа и форму создания отзыва клиента (см. Рисунок 6.1, 6.2), а в приложении user – форму авторизации, создания заказа клиента, добавления услуги, редактирования услуги и редактирования заказа (см. Рисунок 7.1, 7.2, 7.3). Пропишем формы на соответствующих им страницах.

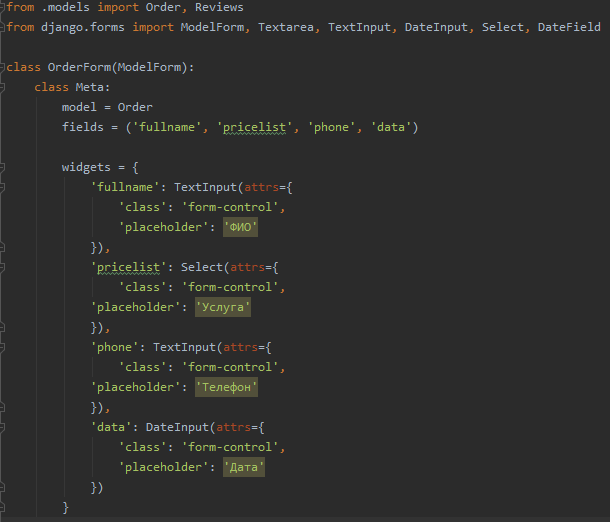


Рисунок 6.1 – Создание форм в приложении main



Рисунок 6.2 – Создание форм в приложении main

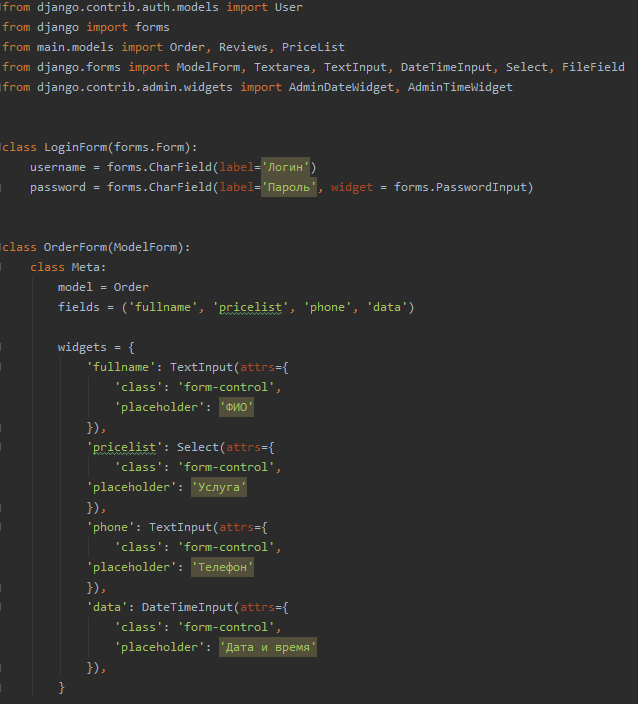


Рисунок 7.1 – Создание форм в приложении user



Рисунок 7.2 – Создание форм в приложении user

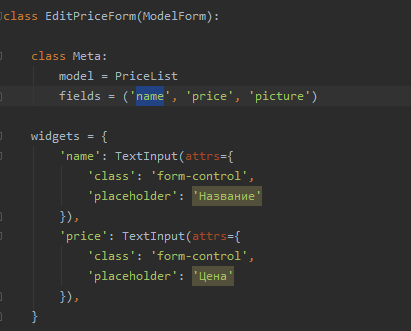


Рисунок 7.3 – Создание форм в приложении user

1. Пропишем представления страниц приложения main (см. Рисунок 8.1, 8.2, 8.3), и в приложения user (см. Рисунок 9.1, 9.2, 9.3, 9.4).



Рисунок 8.1 – views.py приложения main

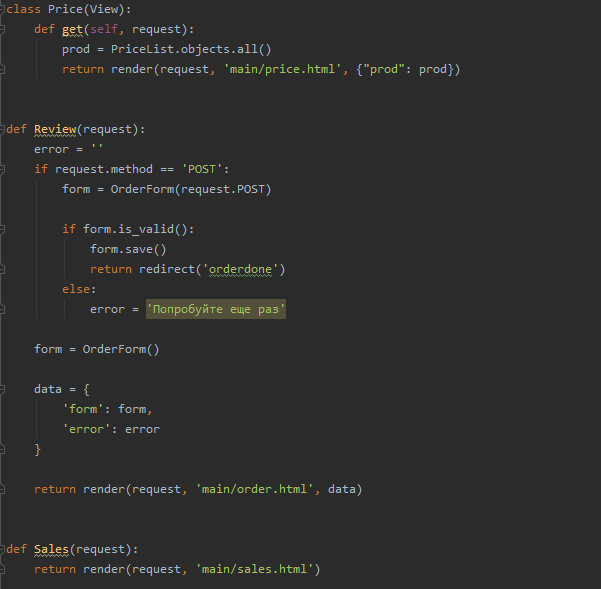


Рисунок 8.2 – views.py приложения main

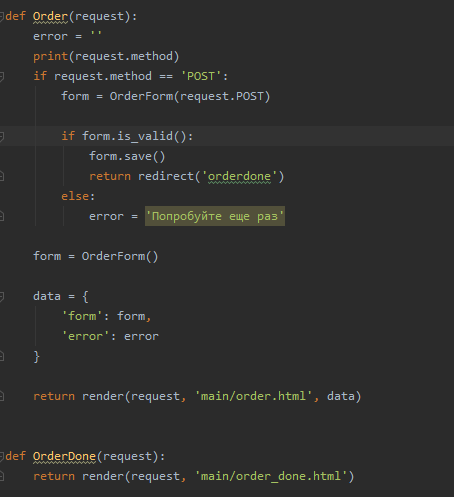


Рисунок 8.3 – views.py приложения main

def ReviewsView(request):

error = ''

prod = Reviews.objects.all()

if request.method == 'POST':

form = ReviewForm(request.POST)

if form.is\_valid():

form.save()

return redirect('reviews')

else:

error = 'Попробуйте еще раз'

form = ReviewForm()

data = {

'form': form,

'error': error,

"prod": prod

}

return render(request, 'main/reviews.html', data)

class NewsView(View):

def get(self, request):

prod = News.objects.all()

return render(request, 'main/news.html', {"prod": prod})

class Price(View):

def get(self, request):

prod = PriceList.objects.all()

return render(request, 'main/price.html', {"prod": prod})

def Review(request):

error = ''

if request.method == 'POST':

form = OrderForm(request.POST)

if form.is\_valid():

form.save()

return redirect('orderdone')

else:

error = 'Попробуйте еще раз'

form = OrderForm()

data = {

'form': form,

'error': error

}

return render(request, 'main/order.html', data)

def Order(request):

error = ''

print(request.method)

if request.method == 'POST':

form = OrderForm(request.POST)

if form.is\_valid():

form.save()

return redirect('orderdone')

else:

error = 'Попробуйте еще раз'

form = OrderForm()

data = {

'form': form,

'error': error

}

return render(request, 'main/order.html', data)

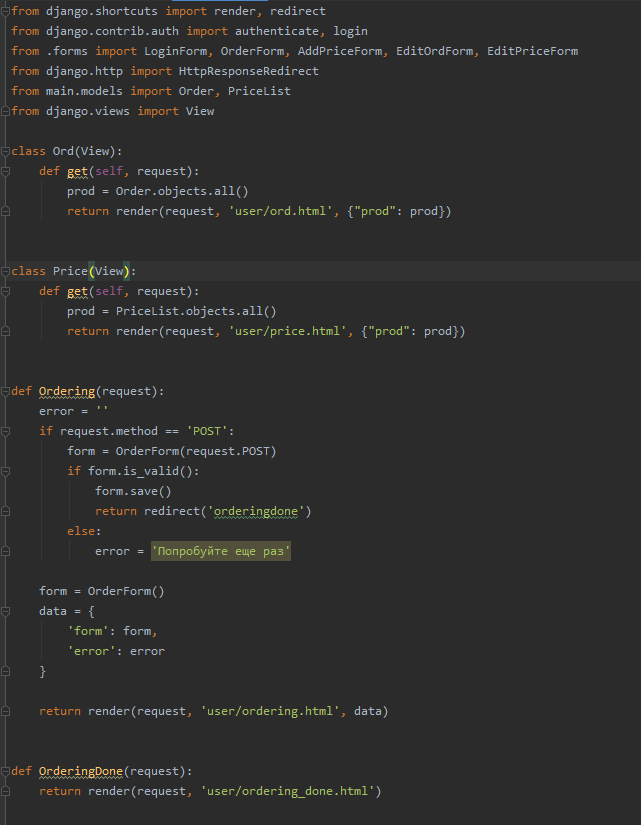


Рисунок 9.1 – views.py приложения user

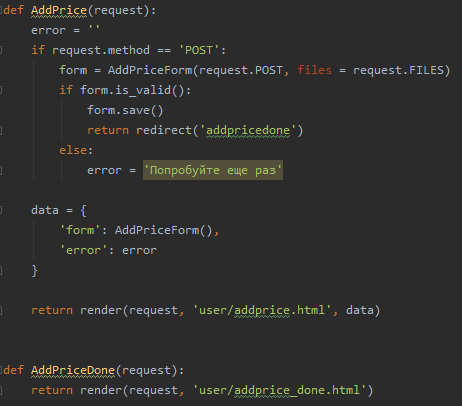


Рисунок 9.2 – views.py приложения user

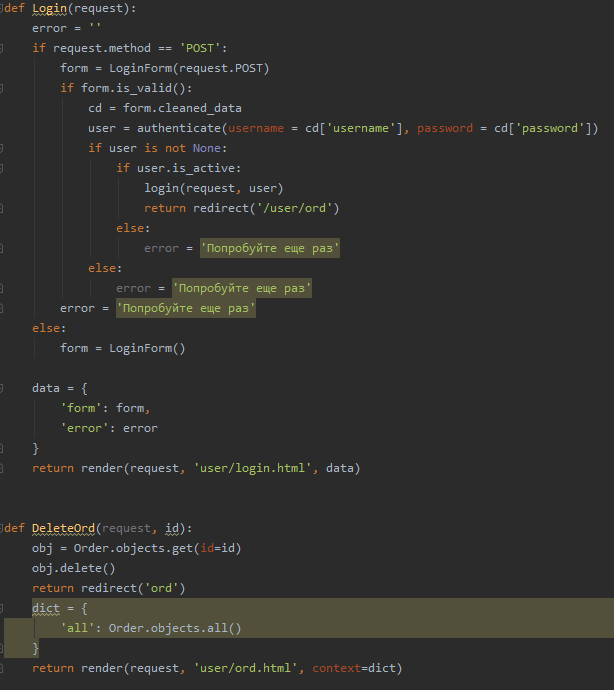


Рисунок 9.3 – views.py приложения user

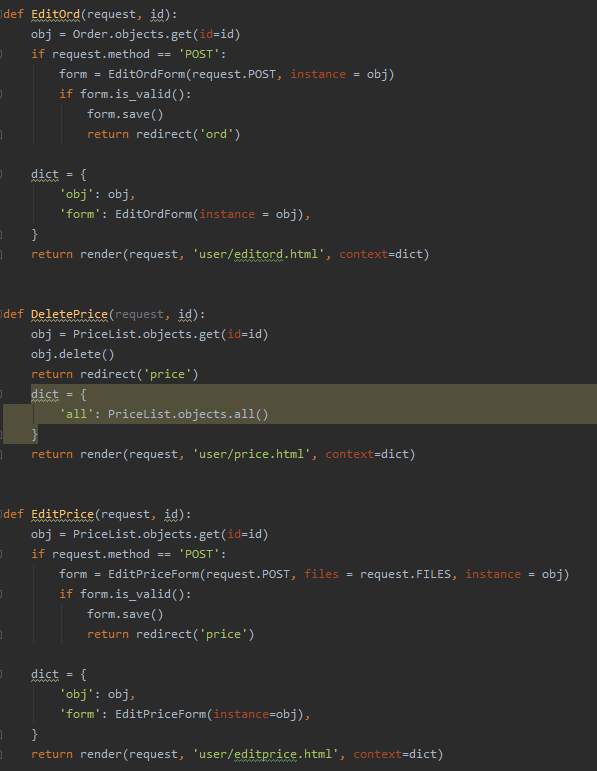


Рисунок 9.4 – views.py приложения user

class Ord(View):

def get(self, request):

prod = Order.objects.all()

return render(request, 'user/ord.html', {"prod": prod})

class Price(View):

def get(self, request):

prod = PriceList.objects.all()

return render(request, 'user/price.html', {"prod": prod})

def Ordering(request):

error = ''

if request.method == 'POST':

form = OrderForm(request.POST)

if form.is\_valid():

form.save()

return redirect('orderingdone')

else:

error = 'Попробуйте еще раз'

form = OrderForm()

data = {

'form': form,

'error': error

}

return render(request, 'user/ordering.html', data)

def AddPrice(request):

error = ''

if request.method == 'POST':

form = AddPriceForm(request.POST, files = request.FILES)

if form.is\_valid():

form.save()

return redirect('addpricedone')

else:

error = 'Попробуйте еще раз'

data = {

'form': AddPriceForm(),

'error': error

}

return render(request, 'user/addprice.html', data)

def Login(request):

error = ''

if request.method == 'POST':

form = LoginForm(request.POST)

if form.is\_valid():

cd = form.cleaned\_data

user = authenticate(username = cd['username'], password = cd['password'])

if user is not None:

if user.is\_active:

login(request, user)

return redirect('/user/ord')

else:

error = 'Попробуйте еще раз'

else:

error = 'Попробуйте еще раз'

error = 'Попробуйте еще раз'

else:

form = LoginForm()

data = {

'form': form,

'error': error

}

return render(request, 'user/login.html', data)

def DeleteOrd(request, id):

obj = Order.objects.get(id=id)

obj.delete()

return redirect('ord')

dict = {

'all': Order.objects.all()

}

return render(request, 'user/ord.html', context=dict)

def EditOrd(request, id):

obj = Order.objects.get(id=id)

if request.method == 'POST':

form = EditOrdForm(request.POST, instance = obj)

if form.is\_valid():

form.save()

return redirect('ord')

dict = {

'obj': obj,

'form': EditOrdForm(instance = obj),

}

return render(request, 'user/editord.html', context=dict)

def DeletePrice(request, id):

obj = PriceList.objects.get(id=id)

obj.delete()

return redirect('price')

dict = {

'all': PriceList.objects.all()

}

return render(request, 'user/price.html', context=dict)

def EditPrice(request, id):

obj = PriceList.objects.get(id=id)

if request.method == 'POST':

form = EditPriceForm(request.POST, files = request.FILES, instance = obj)

if form.is\_valid():

form.save()

return redirect('price')

dict = {

'obj': obj,

'form': EditPriceForm(instance=obj),

}

return render(request, 'user/editprice.html', context=dict)

1. Пропишем ссылки страниц приложения main (см. Рисунок 10), и приложения user (см. Рисунок 11).

urlpatterns = [

path('', views.Main, name = 'main'),

path('reviews', views.ReviewsView, name = 'reviews'),

path('sales', views.Sales, name = 'sales'),

path('news', NewsView.as\_view(), name = 'news'),

path('portfolio', views.Portfolio, name = 'portfolio'),

path('price', Price.as\_view(), name = 'price'),

path('order', views.Order, name = 'order'),

path('orderdone', views.OrderDone, name = 'orderdone'),

] + static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

urlpatterns = [

url(r'^login/$', views.Login, name='login'),

path('ord', Ord.as\_view(), name = 'ord'),

path('price', Price.as\_view(), name = 'price'),

path('ordering', views.Ordering, name='ordering'),

path('orderingdone', views.OrderingDone, name='orderingdone'),

path('addprice', views.AddPrice, name='addprice'),

path('addpricedone', views.AddPriceDone, name='addpricedone'),

path('deleteord/<int:id>', views.DeleteOrd, name = 'deleteord'),

path('editord/<int:id>', views.EditOrd, name = 'editord'),

path('deleteprice/<int:id>', views.DeletePrice, name = 'deleteprice'),

path('editprice/<int:id>', views.EditPrice, name = 'editprice'),

]

if settings.DEBUG:

urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

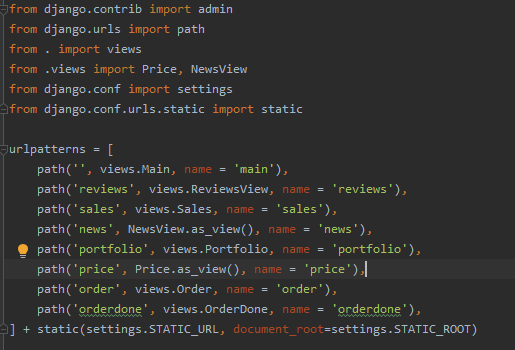


Рисунок 10 – urls.py приложения main

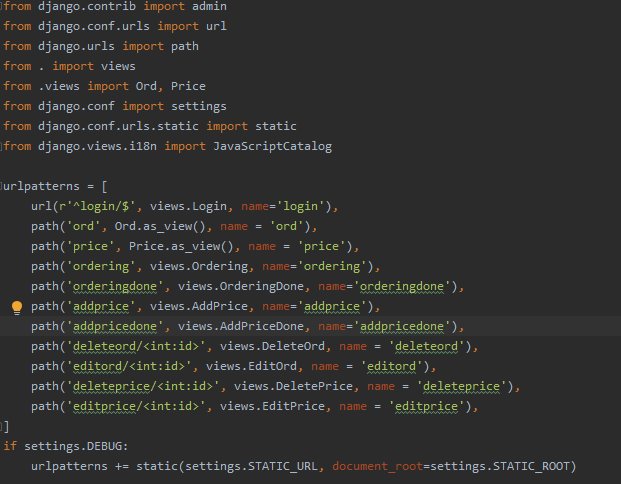


Рисунок 11 – urls.py приложения user

1. Поверим работоспособность всех страниц сайта, а также базы данных.
2. **Оформить отчет и предоставить его на веб-платформу** (файлом или ссылкой на репозиторий GitHub, с загруженным отчетом). В отчёте необходимо включить скриншоты вашего проекта, в качестве приложения загрузите архив с файлами вашей разработки.

**Критерии оценивания:**

|  |  |
| --- | --- |
| **Оценка** | **Критерии оценивания выполнения задач** |
| 5 | Ход решения верный, все его шаги выполнены правильно. |
| 4 | Ход решения верный, все его шаги выполнены правильно, есть ошибки. |
| 3 | Ход решения верный, задание решено не полностью. |
| 2 | Другие случаи, не соответствующие указанным критериям. |