**PYTHON SCRIPT**

|  |  |
| --- | --- |
| DATE | 14 NOVEMBER 2022 |
| PROJECT ID | PNT2022TMID24781 |
| PROJECT NAME | SMART SOLUTION FOR RAILWAYS |

**CODE:**

from django.contrib.auth.base\_user import AbstractBaseUser

from django.db import models

class User(AbstractBaseUser):

"""

User model.

"""

USERNAME\_FIELD = "email"

REQUIRED\_FIELDS = ["first\_name", "last\_name"]

email = models.EmailField(

verbose\_name="E-mail",

unique=True

)

first\_name = models.CharField(

verbose\_name="First name",

max\_length=30

)

last\_name = models.CharField(

verbose\_name="Last name",

max\_length=40

)

city = models.CharField(

verbose\_name="City",

max\_length=40

)

stripe\_id = models.CharField(

verbose\_name="Stripe ID",

unique=True,

max\_length=50,

blank=True,

null=True

)

objects = UserManager()

@property

def get\_full\_name(self):

return f"{self.first\_name} {self.last\_name}"

class Meta:

verbose\_name = "User"

verbose\_name\_plural = "Users"

class Profile(models.Model):

"""

User's profile.

"""

phone\_number = models.CharField(

verbose\_name="Phone number",

max\_length=15

)

date\_of\_birth = models.DateField(

verbose\_name="Date of birth"

)

postal\_code = models.CharField(

verbose\_name="Postal code",

max\_length=10,

blank=True

)

address = models.CharField(

verbose\_name="Address",

max\_length=255,

blank=True

)

class Meta:

abstract = True

class UserProfile(Profile):

"""

User's profile model.

"""

user = models.OneToOneField(

to=User, on\_delete=models.CASCADE, related\_name="profile",

)

group = models.CharField(

verbose\_name="Group type",

choices=GroupTypeChoices.choices(),

max\_length=20,

default=GroupTypeChoices.EMPLOYEE.name,

)

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

return self.user.email

class Meta:

# user 1 - employer

user1, \_ = User.objects.get\_or\_create(

email="foo@bar.com",

first\_name="Employer",

last\_name="Testowy",

city="Białystok",

)

user1.set\_unusable\_password()

group\_name = "employer"

\_profile1, \_ = UserProfile.objects.get\_or\_create(

user=user1,

date\_of\_birth=datetime.now() - timedelta(days=6600),

group=GroupTypeChoices(group\_name).name,

address="Myśliwska 14",

postal\_code="15-569",

phone\_number="+48100200300",

)

# user2 - employee

user2, \_ = User.objects.get\_or\_create()

email="bar@foo.com",

first\_name="Employee",

last\_name="Testowy",

city="Białystok",

)

user2.set\_unusable\_password()

group\_name = "employee"

\_profile2, \_ = UserProfile.objects.get\_or\_create()

user=user2,

date\_of\_birth=datetime.now() - timedelta(days=7600),

group=GroupTypeChoices(group\_name).name,

address="Myśliwska 14",

postal\_code="15-569",

phone\_number="+48200300400",

)

response\_customer = stripe.Customer.create()

email=user.email,

description=f"EMPLOYER - {user.get\_full\_name}",

name=user.get\_full\_name,

phone=user.profile.phone\_number,

)

user1.stripe\_id = response\_customer.stripe\_id

user1.save()

mcc\_code, url = "1520", "https://www.softserveinc.com/"

response\_ca = stripe.Account.create()

type="custom",

country="PL",

email=user2.email,

default\_currency="pln",

business\_type="individual",

settings={"payouts": {"schedule": {"interval": "manual", }}},

requested\_capabilities=["card\_payments", "transfers", ],

business\_profile={"mcc": mcc\_code, "url": url},

individual={

"first\_name": user2.first\_name,

"last\_name": user2.last\_name,

"email": user2.email,

"dob": {

"day": user2.profile.date\_of\_birth.day,

"month": user2.profile.date\_of\_birth.month,

"year": user2.profile.date\_of\_birth.year,

},

"phone": user2.profile.phone\_number,

"address": {

"city": user2.city,

"postal\_code": user2.profile.postal\_code,

"country": "PL",

"line1": user2.profile.address,

},

},

)

user2.stripe\_id = response\_ca.stripe\_id

user2.save()

tos\_acceptance = {"date": int(time.time()), "ip": user\_ip},

stripe.Account.modify(user2.stripe\_id, tos\_acceptance=tos\_acceptance)

passport\_front = stripe.File.create(

purpose="identity\_document",

file=\_file, # ContentFile object

stripe\_account=user2.stripe\_id,

)

individual = {

"verification": {

"document": {"front": passport\_front.get("id"),},

"additional\_document": {"front": passport\_front.get("id"),},

}

}

stripe.Account.modify(user2.stripe\_id, individual=individual)

new\_card\_source = stripe.Customer.create\_source(user1.stripe\_id, source=token)

stripe.SetupIntent.create(

payment\_method\_types=["card"],

customer=user1.stripe\_id,

description="some description",

payment\_method=new\_card\_source.id,

)

payment\_method = stripe.Customer.retrieve(user1.stripe\_id).default\_source

payment\_intent = stripe.PaymentIntent.create(

amount=amount,

currency="pln",

payment\_method\_types=["card"],

capture\_method="manual",

customer=user1.stripe\_id, # customer

payment\_method=payment\_method,

application\_fee\_amount=application\_fee\_amount,

transfer\_data={"destination": user2.stripe\_id}, # connect account

description=description,

metadata=metadata,

)

payment\_intent\_confirm = stripe.PaymentIntent.confirm(

payment\_intent.stripe\_id, payment\_method=payment\_method

)

stripe.PaymentIntent.capture(

payment\_intent.id, amount\_to\_capture=amount

)

stripe.Balance.retrieve(stripe\_account=user2.stripe\_id)

stripe.Charge.create(

amount=amount,

currency="pln",

source=user2.stripe\_id,

description=description

)

stripe.PaymentIntent.cancel(payment\_intent.id)

unique\_together = ("user", "group")