# 用户

## 用户表

### 用户表1

class User(AbstractUser):  
 # 用户模型  
 nickname = models.CharField('密码', max\_length=32)  
 avatar = models.ImageField('头像', upload\_to='avatar/%Y%m', null=True, blank=True)  
  
 class Meta:  
 db\_table = 'account\_user'  
 verbose\_name = '用户表'  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.username

### 用户表2

class User(CommonModel):  
 *'''用户信息'''* USER\_STATUS = (('0', '删除'), ('1', '正常'))  
 username = models.CharField('用户名', max\_length=20, unique=True)  
 password = models.CharField('密码', max\_length=256)  
 nickname = models.CharField('昵称', max\_length=20, null=True)  
 avatar = models.ImageField('头像', null=True, upload\_to='avatar')  
 status = models.CharField('用户状态', max\_length=3, choices=USER\_STATUS, default=1)  
 is\_super = models.BooleanField('超级用户', default=0)  
 is\_delete = models.BooleanField('已删除', default=0)  
  
 class Meta:  
 db\_table = 'accounts\_user'  
  
 def \_\_str\_\_(self):  
 return self.username

## 用户详情表

### 用户详情表1

class Profile(models.Model):  
 # 用户详细信息表  
 SEX\_CHOICES = ((0, '男'), (1, '女'), (2, '保密'))  
 username = models.CharField('用户名', max\_length=64, editable=True, null=True, blank=True)  
 user = models.OneToOneField(to='User', related\_name='profile', on\_delete=models.CASCADE, verbose\_name='关联用户')  
 sex = models.SmallIntegerField('性别', choices=SEX\_CHOICES, default=2)  
 age = models.SmallIntegerField('年龄', null=True, blank=True)  
 real\_name = models.CharField('真实姓名', max\_length=20, null=True, blank=True)  
 email = models.CharField('邮箱', max\_length=128, null=True, blank=True)  
 is\_email\_valid = models.BooleanField('邮箱验证', default=False)  
 phone\_no = models.CharField('手机号', max\_length=11, null=True, blank=True)  
 is\_phone\_valid = models.BooleanField('手机号验证', default=False)  
  
 source = models.CharField('登录来源', max\_length=16, null=True, blank=True)  
 version = models.CharField('版本', max\_length=16, null=True, blank=True)  
 updated\_at = models.DateTimeField('更新时间', auto\_now=True)  
 created\_at = models.DateTimeField('创建时间', auto\_now\_add=True)  
  
 class Meta:  
 db\_table = 'user\_profile'  
 verbose\_name = '用户详情表'  
 verbose\_name\_plural = verbose\_name

### 用户详情表2

class UserProfile(CommonModel):  
 *'''用户详情'''* SEX = (('0', '女'), ('1', '男'), ('2', '隐藏'))  
 user = models.OneToOneField(verbose\_name='关联用户', to='User', on\_delete=models.CASCADE, related\_name='profile')  
 username = models.CharField('用户名', max\_length=20, unique=True)  
 real\_name = models.CharField('真实姓名', max\_length=20, null=True)  
 sex = models.CharField('性别', max\_length=2, choices=SEX, default=2)  
 maxim = models.CharField('用户格言', max\_length=128, null=True)  
 address = models.CharField('用户地址', max\_length=128, null=True)  
  
 class Meta:  
 db\_table = 'accounts\_user\_profile'  
  
 def \_\_str\_\_(self):  
 return self.username

## 用户登录日志表

class LoginRecord(models.Model):  
 # 用户登录日志  
 user = models.ForeignKey(to='User', related\_name='login\_record', on\_delete=models.CASCADE, verbose\_name='关联用户')  
 username = models.CharField('用户名', max\_length=64, null=True, blank=True)  
 ip = models.CharField('IP', max\_length=32, null=True, blank=True)  
 address = models.CharField('地址', max\_length=32, null=True, blank=True)  
 source = models.CharField('登录来源', max\_length=16, null=True, blank=True)  
 version = models.CharField('版本', max\_length=16, null=True, blank=True)  
  
 created\_at = models.DateTimeField('登录时间', auto\_now\_add=True)  
  
 class Meta:  
 db\_table = 'login\_record'  
 verbose\_name = '用户日志表'  
 verbose\_name\_plural = verbose\_name

# 景点

## 景点信息表

class Sight(CommonModel):  
 # 景点基础信息  
 name = models.CharField('名称', max\_length=64)  
 desc = models.CharField('描述', max\_length=256, null=True, blank=True)  
 main\_img = models.ImageField('主图', max\_length=256, upload\_to='%Y%m/sight/')  
 banner\_img = models.ImageField('详情主图', max\_length=256, upload\_to='%Y%m/sight/')  
 content = models.TextField('详细', null=True, blank=True)  
 score = models.FloatField('评分', default=5)  
 province = models.CharField('省份', max\_length=32, null=True, blank=True)  
 city = models.CharField('城市', max\_length=32, null=True, blank=True)  
 area = models.CharField('区/县', max\_length=32, null=True, blank=True)  
 town = models.CharField('乡镇', max\_length=32, null=True, blank=True)  
 min\_price = models.FloatField('最低价格', default=0)  
 is\_top = models.BooleanField('是否精选', default=False)  
 is\_hot = models.BooleanField('是否热门', default=False)  
  
 images = GenericRelation(to=ImageRelated, verbose\_name='关联图片', related\_query\_name='rel\_sight\_images',  
 related\_name='rel\_sight\_ima')  
  
 class Meta:  
 db\_table = 'sight'  
 ordering = ['-updated\_at']  
 verbose\_name = '景点信息表'  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name

## 景点详情表

class Info(models.Model):  
 # 景点详情  
 sight = models.OneToOneField(to=Sight, on\_delete=models.CASCADE, verbose\_name='关联景点')  
 entry\_explain = models.TextField('入园参考', null=True, blank=True)  
 play\_way = models.TextField('特色玩法', null=True, blank=True)  
 tips = models.TextField('温馨提示', null=True, blank=True)  
 traffic = models.TextField('交通到达', null=True, blank=True)  
  
 class Meta:  
 db\_table = 'sight\_info'  
 verbose\_name = '景点详情表'  
 verbose\_name\_plural = verbose\_name

## 门票表

class Ticket(CommonModel):  
 # 门票  
 sight = models.ForeignKey(to=Sight, related\_name='tickets', on\_delete=models.CASCADE, verbose\_name='景点门票')  
 name = models.CharField('名称', max\_length=128)  
 desc = models.CharField('描述', max\_length=64, null=True, blank=True)  
 type = models.SmallIntegerField('类型', help\_text='默认成人票', choices=TicketTypes.choices, default=TicketTypes.ADULT)  
 price = models.FloatField('价格(原价)')  
 discount = models.FloatField('折扣', default=10)  
 total\_stock = models.PositiveIntegerField('总库存', default=0)  
 remain\_stock = models.PositiveIntegerField('剩余库存', default=0)  
 expire\_date = models.IntegerField('有效期', default=0)  
 return\_policy = models.CharField('退改政策', max\_length=64, default='条件退')  
 has\_invoice = models.BooleanField('是否提供发票', default=True)  
 entry\_way = models.SmallIntegerField('入园方式', choices=EnterWay.choices, default=EnterWay.BY\_TICKET)  
 tips = models.TextField('预定须知', null=True, blank=True)  
 remark = models.TextField('其他说明', null=True, blank=True)  
 status = models.SmallIntegerField('状态', choices=TicketStatus.choices, default=TicketStatus.OPEN)  
  
 class Meta:  
 db\_table = 'sirht\_ticket'  
 verbose\_name = '门票表'  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name

## 景点评论及回复表

class Comment(CommonModel):  
 # 评论和回复  
 user = models.ForeignKey(to=User, related\_name='user\_comments', on\_delete=models.CASCADE, verbose\_name='评论人')  
 sight = models.ForeignKey(to=Sight, related\_name='sight\_comments', on\_delete=models.CASCADE, verbose\_name='景点')  
 content = models.TextField('评论内容', null=True, blank=True)  
 is\_top = models.BooleanField('是否置顶', default=False)  
 love\_count = models.IntegerField('点赞数', default=0)  
 score = models.FloatField('评分', default=0)  
  
 ip\_address = models.CharField('IP地址', max\_length=64, null=True, blank=True)  
 is\_public = models.SmallIntegerField('是否公开', default=1)  
 reply = models.ForeignKey(to='self', related\_name='related\_name', on\_delete=models.CASCADE, verbose\_name='回复',  
 blank=True, null=True)  
  
 images = GenericRelation(to=ImageRelated, verbose\_name='关联图片', related\_query\_name='rel\_sight\_images',  
 related\_name='rel\_sight\_ima')  
  
 class Meta:  
 db\_table = 'sight\_comment'  
 ordering = ['-love\_count', '-created\_at']  
 verbose\_name = '评论表'  
 verbose\_name\_plural = verbose\_name

# 订单

## 订单表

class Order(CommonModel):  
 # 订单  
 sn = models.CharField('订单编号', max\_length=32)  
 user = models.ForeignKey(to=User, on\_delete=models.CASCADE, related\_name='order\_user', verbose\_name='关联用户')  
 buy\_count = models.IntegerField('购买数量', default=1)  
 buy\_amount = models.FloatField('总价', default=0.0)  
  
 to\_user = models.CharField('收货人', max\_length=32)  
 to\_area = models.CharField('省市区', max\_length=32, null=True, blank=True)  
 to\_address = models.CharField('详细地址', max\_length=256, null=True, blank=True)  
 to\_phone = models.CharField('手机号码', max\_length=16)  
 remark = models.CharField('备注', max\_length=255, null=True, blank=True)  
  
 # 快递信息  
 express\_type = models.CharField('快递', max\_length=32, null=True, blank=True)  
 express\_no = models.CharField('单号', max\_length=32, null=True, blank=True)  
  
 status = models.SmallIntegerField('订单状态', choices=OrderStatus.choices, default=OrderStatus.SUBMIT)  
 types = models.SmallIntegerField('订单类型', choices=OrderTypes.choices, default=OrderTypes.SIGHT\_TICKET)  
  
 class Meta:  
 db\_table = 'order'  
 verbose\_name = '订单表'  
 verbose\_name\_plural = verbose\_name

## 订单明细表

class OrderItem(CommonModel):  
 *""" 订单明细 """* user = models.ForeignKey(to=User, related\_name='order\_items', on\_delete=models.CASCADE, verbose\_name='关联用户')  
 order = models.ForeignKey(to=Order, on\_delete=models.CASCADE, related\_name='order\_items', null=True,  
 verbose\_name='关联订单')  
 # 商品快照  
 flash\_name = models.CharField('商品名称', max\_length=128, null=True, blank=True)  
 flash\_img = models.ImageField('商品的主图', null=True, blank=True)  
 flash\_price = models.FloatField('价格', default=0.0)  
 flash\_origin\_price = models.FloatField('原价', default=0.0)  
 flash\_discount = models.FloatField('折扣', default=0.0)  
 count = models.PositiveIntegerField('购买数量', default=1)  
 amount = models.FloatField('总额', default=0.0)  
  
 status = models.SmallIntegerField('订单状态', choices=OrderStatus.choices, default=OrderStatus.SUBMIT)  
 remark = models.CharField('备注', max\_length=255, null=True, blank=True)  
  
 # 复合关联  
 content\_type = models.ForeignKey(ContentType, on\_delete=models.CASCADE)  
 object\_id = models.PositiveIntegerField()  
 content\_object = GenericForeignKey('content\_type', 'object\_id')  
  
 class Meta:  
 db\_table = 'order\_item'  
 verbose\_name = '订单明细表'  
 verbose\_name\_plural = verbose\_name

## 支付凭证表

class Payment(CommonModel):  
 *""" 支付凭证 """* user = models.ForeignKey(to=User, on\_delete=models.CASCADE, related\_name='payments')  
 order = models.ForeignKey(to=Order, on\_delete=models.CASCADE, related\_name='payments')  
 amount = models.FloatField('金额', default=0.0, help\_text='支付实际的金额')  
 sn = models.CharField('流水号', max\_length=32, null=True, blank=True)  
 third\_sn = models.CharField('第三方订单号', max\_length=128, null=True, blank=True)  
  
 status = models.SmallIntegerField('支付状态', default=1)  
  
 meta = models.CharField('其他数据', max\_length=128, null=True, blank=True)  
 remark = models.CharField('备注信息', max\_length=128, null=True, blank=True)  
  
 class Meta:  
 db\_table = 'order\_payment'  
 verbose\_name = '支付信息表'  
 verbose\_name\_plural = verbose\_name

# 系统

## 图片表

class ImageRelated(CommonModel):  
 img = models.ImageField('图片', upload\_to='%Y%m/file/')  
 summary = models.CharField('说明', max\_length=32, null=True, blank=True)  
 user = models.ForeignKey(to=User, verbose\_name='上传用户', related\_name='upload\_images', on\_delete=models.CASCADE,  
 null=True, blank=True)  
 content\_type = models.ForeignKey(to=ContentType, on\_delete=models.CASCADE)  
 object\_id = models.IntegerField('关联模型')  
 content\_object = GenericForeignKey('content\_type', 'object\_id')  
  
 class Meta:  
 db\_table = 'system\_image\_related'

## 轮播图片表

class Slider(CommonModel):  
 name = models.CharField('名称', max\_length=64)  
 desc = models.CharField('描述', max\_length=256, null=True, blank=True)  
 typrs = models.SmallIntegerField('展现位置', default=10)  
 img = models.ImageField('图片地址', max\_length=255, upload\_to='%Y%m/slider')  
 reorder = models.SmallIntegerField('排序字段', default=0, help\_text='数字越大越靠前')  
 start\_time = models.DateTimeField('生效时间', null=True, blank=True)  
 end\_time = models.DateTimeField('结束时间', null=True, blank=True)  
 target\_url = models.CharField('目标地址', max\_length=255, null=True, blank=True)  
  
 class Meta:  
 db\_table = 'system\_slider'  
 ordering = ['-reorder']  
 verbose\_name = '系统表'  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name