"بسم الله الرحمن الرحیم"

عنوان پروژه :پیاده‌سازی ORM با پایتون وSQLite

دانشجو :ملیکا محبوبی

دانشگاه :دانشگاه گیلان، رشت

دانشکده :فنی و مهندسی شرق

ترم :۶

استاد دکتر سید دانش

سال تحصیلی :۱۴۰۴-۱۴۰۵

مستند پروژه ORM با پایتون

۱ .مقدمه

این پروژه یک ORM (Object Relational Mapping) ساده با استفاده از **زبان Python** و پایگاه داده **SQLite**پیاده‌سازی می‌کند. هدف اصلی پروژه، آشنایی با:

- برنامه‌نویسی شیءگرا (OOP) درPython
- ارتباط بین کلاس‌ها و جداول دیتابیس
- انجام عملیات پایه‌ای پایگاه داده(CRUD)
- طراحی و پیاده‌سازی سیستم تست خودکار برای تضمین کیفیت

ORMساخته‌شده به کاربران اجازه می‌دهد کلاس‌های Python را به جداول پایگاه داده نگاشت کنند و بدون نوشتن مستقیمSQL ، داده‌ها را مدیریت نمایند.

۲ .ساختار پروژه

```
OrmProject
├── app.py                فایل اصلی اجرای Flask UI #
├── orm/
│   ├── database.py       مدیریت اتصال و اجرای کوئری‌ها #
```

```
│   ├── fields.py          # تعریف فیلدها (IntegerField, CharField, BooleanField)
│   └── basemodel.py       # کلاس BaseModel پایه ORM
├── models/
│   ├── user.py            # مدل User به عنوان نمونه
│   ├── post.py            # مدل Post (اختیاری برای تست‌های اضافی)
│   └── comment.py         # مدل Comment (اختیاری برای تست‌های اضافی)
├── tests/
│   ├── test_user.py       # تست‌های واحد مدل User
│   ├── test_post.py       # تست‌های مدل Post
│   └── test_comment.py    # تست‌های مدل Comment
└── templates/
    └── index.html         # رابط وب (UI)
```

۳ .شرح کلیORM

کلاس Fieldو زیرکلاس‌ها

- **Field**:پایه برای تعریف ویژگی‌های ستون‌ها

ویژگی‌ها:

  - primary_key : مشخص می‌کند ستون کلید اصلی است
  - unique : محدودیت یکتا
  - null : مقدار NULL مجاز است یا خیر
  - default : مقدار پیش‌فرض
  - max_length : حداکثر طول برای رشته‌ها

- **IntegerField**:برای اعداد صحیح
- **CharField**:برای رشته‌ها با طول قابل تنظیم
- **BooleanField**:برای مقادیر بولین) ذخیره‌شده به صورت ۱/۰ در SQLite)

ویژگی امتیازی :اعتبارسنجی نوع داده و طول رشته پیش از ذخیره.

۳ .کلاس Database

- مدیریت **اتصال امن به**SQLite
- متدها:
  - connect() : اتصال به فایل دیتابیس و ایجادcursor
  - execute(query, params) : اجرای کوئری و commit خودکار
  - executemany(query, seq_of_params) : اجرای چندگانه
  - close() : بستن اتصال

ویژگی امتیازی check_same_thread=False :برای استفاده در چند thread بدون ارور.

کلاس BaseModel

- پایه برای همه مدل‌ها(User, Post, Comment)
- متدها:
  - iter_fields() : پیمایش فیلدهای کلاس
  - create_table() : ایجاد جدول در دیتابیس به صورت خودکار
  - __init__(**kwargs) : مقداردهی اولیه با مقادیر پیش‌فرض
  - validate() : اعتبارسنجی مقادیر قبل از ذخیره

- save() : ذخیره یا بروزرسانی رکورد(Insert/Update)
- get(**kwargs) : گرفتن اولین رکورد مطابق شرط
- filter(**kwargs) : گرفتن لیست رکوردها با فیلتر
- delete() : حذف رکورد با کلید اصلی
- all() : گرفتن همه رکوردها

ویژگی‌های امتیازی:

- اعتبارسنجی null ، نوع داده و طول رشته
- مدیریت PK خودکار
- پشتیبانی از فیلتر و جستجوی منعطف

مدل User نمونه

```
class User(BaseModel):
table_name = "users"
id = IntegerField(primary_key=True)
name = CharField(max_length=127, null=True)
email = CharField(max_length=255, unique=True)
```

ویژگی امتیازی : مدل کاملاً قابل توسعه، قابلیت اضافه کردن فیلدهای بیشتر، یا مدل‌های دیگر.

۴ UI و تعامل با کاربر

- رابط تحت وب با **FlaskوBootstrap**
- فرم افزودن کاربر، جستجو و حذف رکوردها
- نمایش جدول کاربران در همان صفحه بدون رفرش کامل
- ویژگی امتیازی:
  - هایلایت رکوردهای جستجو شده
  - حذف رکورد با کلید روی همان صفحه

۵ .تست‌های واحد(Unit Tests)

- تمام متدهای مدل User تست شده‌اند:
  - ایجاد رکورد(test_create_user)
  - بروزرسانی(test_update_user)
  - حذف(test_delete_user)
  - فیلتر(test_filter_all_and_by_field)
  - محدودیت(test_unique_email_constraint) unique
  - مقدار null مجاز(test_null_name_allowed)

اجرای تست‌ها:

#اجرای تک تک فایل‌ها

```
python -m unittest tests/test_user.py

python -m unittest tests/test_post.py

python -m unittest tests/test_comment.py
```

#اجرای همه تست‌ها

```
python -m unittest discover -s tests
```

پس دوباره اشاره میکنم که :

تست‌ها رو می‌تونی **تک‌تک یا همه با هم** اجرا کنی:

- **تک‌تک فایل تست:**

```
python -m unittest tests.test_user
```

یا اگر فایل دیگه‌ای بود، مثل :tests.test_post

```
python -m unittest tests.test_post
```

- **همه تست‌ها با یک دستور:**

```
python -m unittest discover -s tests
```

گزارش پوشش کد:(coverage)

pip install coverage

coverage run -m unittest discover -s tests

coverage html

#باز کردن گزارش در مرورگر

بعد می توان فایل htmlcov/index.htmlرو باز کرد تا گزارش پوشش تست‌ها رو دید.

start htmlcov/index.html

ویژگی امتیازی :تست جامع و پوشش کامل متدها، اطمینان از صحت عملکرد و جلوگیری از ارور در زمان اجرا.

۶ .نکات امتیازی و امنیتی

- مدیریت امن اتصال به SQLite وthread-safe
- اعتبارسنجی داده‌ها قبل از ذخیره
- پشتیبانی از ویژگی‌های پیشرفته Field (Unique, Default, Null, Max length)
- الاتعاملی و بدون رفرش کامل برای بهبود تجربه کاربری
- طراحی قابل توسعه برای افزودن مدل‌های جدید بدون تغییر ساختار اصلی

این پروژه یک **ORMکامل و قابل توسعه با پایتون** ایجاد کرده است که:

- عملیات CRUD را به صورت امن و آسان فراهم می‌کند
- اعتبارسنجی داده‌ها و مدیریت اتصال دیتابیس را انجام می‌دهد
- رابط وب زیبا و تعاملی دارد
- تمام تست‌های واحد با گزارش پوشش کد ارائه شده‌اند

ویژگی‌ها و قابلیت‌های پروژه:

۱. **مدل‌سازی داده‌ها به صورت کلاس‌هایPython**

هر جدول در دیتابیس توسط یک کلاس Python نمایندگی می‌شود و هر فیلد جدول به صورت **شیء Field** تعریف می‌شود. این رویکرد باعث می‌شود که مدیریت داده‌ها و تغییر ساختار جداول بسیار ساده و خوانا باشد.

۲. **عملیات پایه‌ای پایگاه داده(CRUD)**

- **Create:** ایجاد رکورد جدید با اعتبارسنجی اتوماتیک
- **Read / Get / Filter:** خواندن رکوردها با امکان فیلتر کردن بر اساس فیلدها
- **Update:** به‌روزرسانی رکوردهای موجود با تشخیص هوشمند تغییرات
- **Delete:** حذف رکوردها با کنترل کلید اصلی و جلوگیری از خطای احتمالی

۳. **ویژگی‌های فیلدها:**

- نوع داده(Integer, Char, Boolean)
- کلید اصلی (Primary Key) با auto-increment
- یکتا بودن(Unique)
- مقدار پیش‌فرض(Default)
- مجاز بودن مقدار تهی(Null)
- محدودیت طول برای رشته‌ها(Max Length)

۴. **ایجاد خودکار جداول در پایگاه داده:**

با متد ()create_table، جداول متناظر با مدل‌ها به صورت خودکار ساخته می‌شوند، بدون نیاز به نوشتن دستورات SQL دستی.

۵. **اعتبارسنجی پیشرفته داده‌ها:**

قبل از ذخیره، تمام داده‌ها بررسی می‌شوند تا مطمئن شویم از لحاظ نوع، محدودیت طول و Null/Uniques مطابقت دارند.

۶. **مدیریت اتصال ایمن به دیتابیس:**

- اتصال به SQLite تنها در صورت نیاز برقرار می‌شود
- امکان استفاده در **چند Thread** فراهم شده تا با Flask و وب‌اپلیکیشن هماهنگی داشته باشد
- مدیریت commit و rollback به صورت خودکار

۷. **پشتیبانی از تست‌های واحد:(Unit Tests)**

- پوشش کامل متدهایCRUD
- تست محدودیت‌های unique وnull
- تست فیلتر و بازیابی داده‌ها
- گزارش پوشش تست با **coverage**برای اطمینان از صحت عملکردORM

۸. **رابط کاربری تحت وب:**

- امکان اضافه کردن کاربران، جستجو و حذف رکوردها بدون تغییر در UI اصلی
- نتایج جستجو به صورت **هایلایت** نمایش داده می‌شوند
- طراحی زیبا و رنگی با استفاده از **Bootstrap 5**
- همه عملیات بدون رفرش کامل صفحه و با JSON و Ajax سازگار است

۹. **ویژگی‌های پیشرفته و امتیازی:**

- امکان **گسترش مدل‌ها** مانند Post و Comment بدون تغییر در ساختار اصلیORM
- مدیریت استثناها و جلوگیری از خطاهای رایجSQLite
- رعایت اصول **SOLIDو Clean Code** در طراحی کلاس‌ها و متدها
- قابلیت **اعمال محدودیت‌ها و قواعد داده‌ای پیچیده** به راحتی با اضافه کردن فیلد وvalidation
- قابلیت استفاده مجدد در پروژه‌های دیگر یا گسترش به دیتابیس‌های بزرگتر