AppStore program

Design

Database Design

app_store table

Listings	type	describe
id	INTEGER	Self-growing ID
app_id	INTEGER	Application ID
track_name	TEXT	Application Name
size_bytes	INTEGER	Application Size (Bytes)
currency	TEXT	Currency Type
price	REAL	price

Listings	type	describe
rating_count_tot	INTEGER	Total number of ratings for all versions
rating_count_ver	INTEGER	Total number of ratings for the current version
user_rating	REAL	User ratings for all versions
user_rating_ver	REAL	User ratings for current version
ver	TEXT	Application Versions
cont_rating	TEXT	Content Ratings
prime_genre	TEXT	Main Type
sup_devices_num	INTEGER	Number of supported devices
ipad5c_urls_num	INTEGER	Number of iPad screenshots
lang_num	INTEGER	Number of languages
vpp_lic	INTEGER	VPP Licensing

app_store_desc Table

Listings	type	describe
id	INTEGER	Self-growing ID
app_id	INTEGER	Application ID

Listings	type	describe
app_desc	TEXT	Application describe

Front-end design

The application was designed using the Bootstrap front-end framework and includes the following pages:

index.html：Used to display the list of applications.

detail.html：Used to display application details.

Back-end design

The application is designed using Python's Flask framework and includes the following components.

Flask application: used to handle HTTP requests and responses.

SQLite database: for storing application and application description data.

Flask Paginate library: used for pagination processing.

Faker library: for random data generation.csv

Library: for reading csv files.

Development

Installing dependencies

Before starting development, the following dependencies need to be installed.

pyenv local 3.7.0

python3 -m venv. venv source. venv/bin/activate

pip install --upgrade pip pip installs Faker

pip install flask pip install Flask-Paginate

pip install behave pip install selenium

pip install gunicorn pip freeze > requirements.txt

1. front-end: Use the Bootstrap front-end framework to design the data contained in the embed template. And realized the front-end logic of paging.

2. back-end: Imported the CSV data into the db file and used the Flask framework to implement the back-end logic, connected to the SQLite database to display the data, and implement the paging back-end logic. Including handling HTTP requests and responses. Write test cases and perform testing.

Implementation

During implementation: first you need to download the dependencies, install, and configure Flask, Flask Paginate, Behave's python dependencies using pyenv and virtual environment management tools. Initialize the SQLite database and import the application data, then start the application, and finally visit the application URL in the browser to view the page.

Test

Development and testing based on front-end, and back-end design the project uses Behave and Selenium for automated testing, and tests can be run using the following commands: behave

The test cases are located under the features folder.