Project Skeleton Structure

```
PythonCode/
├── pyproject.toml              # (опционально) или requirements.txt
├── requirements.txt
├── README.md
├── src/
│   ├── app.py                  # точка входа приложения
│   ├── domain/
│   │   ├── __init__.py
│   │   └── entities.py         # dataclasses сущностей
│   ├── data/
│   │   ├── __init__.py
│   │   ├── connection.py       # менеджер соединения SQLite
│   │   └── repositories/
│   │       ├── __init__.py
│   │       ├── items_repo.py
│   │       ├── suppliers_repo.py
│   │       ├── specifications_repo.py
│   │       └── categories_repo.py
│   ├── services/
│   │   ├── __init__.py
│   │   ├── item_service.py
│   │   ├── supplier_service.py
│   │   ├── specification_service.py
│   │   └── file_service.py
│   └── ui/
│       ├── __init__.py
│       └── qt_models/          # QAbstractListModel / QAbstractTableModel
│           ├── items_model.py
│           ├── suppliers_model.py
│           └── specification_table_model.py
├── qml/
│   └── ...                     # QML UI файлы
├── tests/
│   ├── __init__.py
│   ├── test_items_repo.py
│   ├── test_spec_services.py
│   └── conftest.py
└── .github/
    └── workflows/
        └── ci.yml
```

---

src/domain/entities.py

```
from dataclasses import dataclass
from typing import Optional

@dataclass
class Item:
    id: Optional[int]
    name: str
    article: str
    price: float
    measure: str
```

---

src/data/connection.py

```
from contextlib import contextmanager
import sqlite3

class DBConnection:
    def __init__(self, path: str):
        self.path = path

    @contextmanager
    def session(self):
        conn = sqlite3.connect(self.path)
        try:
            conn.execute('PRAGMA foreign_keys = ON')
            yield conn
            conn.commit()
        except Exception:
            conn.rollback()
            raise
        finally:
            conn.close()
```

---

src/data/repositories/items_repo.py

```
from typing import List, Optional
from ...domain.entities import Item
from ..connection import DBConnection

class ItemsRepository:
    def __init__(self, conn: DBConnection):
        self.conn = conn
```

```
    def get_by_id(self, item_id: int) -> Optional[Item]:
        with self.conn.session() as c:
            row = c.execute(
                'SELECT id, name, article, price, measure FROM items WHERE
id=?',
                (item_id,),
            ).fetchone()
            if not row:
                return None
            return Item(*row)

    def list(self) -> List[Item]:
        with self.conn.session() as c:
            rows = c.execute(
                'SELECT id, name, article, price, measure FROM items'
            ).fetchall()
        return [Item(*r) for r in rows]

    def create(self, item: Item) -> int:
        with self.conn.session() as c:
            cur = c.execute(
                'INSERT INTO items(name, article, price, measure) VALUES
(?, ?, ?, ?)',
                (item.name, item.article, item.price, item.measure),
            )
            return cur.lastrowid
```

---

src/services/item_service.py

```
from ..data.repositories.items_repo import ItemsRepository
from ..domain.entities import Item

class ItemService:
    def __init__(self, repo: ItemsRepository):
        self.repo = repo

    def create_item(self, name: str, article: str, price: float, measure:
str) -> int:
        item = Item(id=None, name=name, article=article, price=price,
measure=measure)
        return self.repo.create(item)

    def get_item(self, item_id: int) -> Item:
        return self.repo.get_by_id(item_id)
```

.github/workflows/ci.yml

```yaml
name: CI

on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.11'

      - name: Install dependencies
        run: |
          pip install -r requirements.txt
          pip install pytest black ruff

      - name: Lint
        run: |
          black --check src
          ruff src

      - name: Tests
        run: pytest
```

---

tests/test_items_repo.py

```python
from src.data.connection import DBConnection
from src.data.repositories.items_repo import ItemsRepository
from src.domain.entities import Item
import tempfile
import os

def test_create_and_get_item():
    db_path = tempfile.NamedTemporaryFile(delete=False).name
    conn = DBConnection(db_path)

    with conn.session() as c:
        c.execute('CREATE TABLE items (id INTEGER PRIMARY KEY, name TEXT,
article TEXT, price REAL, measure TEXT)')
```

```
    repo = ItemsRepository(conn)

    item_id = repo.create(Item(None, "Test", "A1", 10.0, "pcs"))
    item = repo.get_by_id(item_id)

    assert item.name == "Test"
    os.remove(db_path)
```

README.md (шаблон)

```
# PythonCode (skeleton)

## Quick start
```

pip install -r requirements.txt python src/app.py

```

```