51

Flask + SQLite
projekts MiniVeikals

Flask + SQLite

Lai mājaslapā **dati būtu dinamiski** – piemēram, produktu saraksts, ziņas, komentāri u.c. – tos var **ielasīt no datubāzes** un attēlot HTML veidnē.

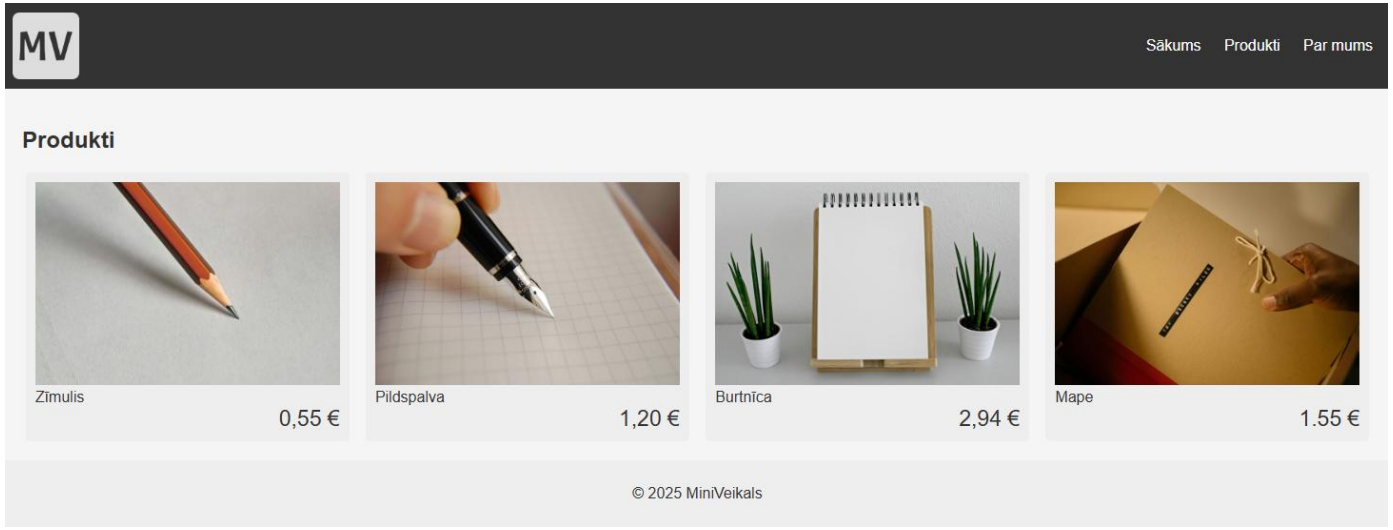
Kā tas darbojas?

- Flask uzņem pieprasījumu no pārlūka (http://127.0.0.1:5000/produkti)
- Flask izveido savienojumu ar datubāzi (datubāzē glabājas dati par produktiem)
- Izpilda SQL vaicājumu, piemēram: SELECT * FROM products
- Rezultāts tiek padots uz HTML veidni (products.html)
- Lietotājs redz rezultātu pārlūkā

Dati datubāzē

id	name	price	image
1	Zīmulis	0,55	pencil.jpg
2	Pildspalva	1,20	pen.jpg
3	Burtnīca	2,94	notebook.jpg
4	Mape	1.55	folder.jpg

Dati mājaslapā



Flask + SQLite

Papildināsim projektu MiniVeikals

1. Mapē `miniVeikals` izveidot jaunu datubāzi `miniVeikals.db`
2. Datubāzē izveidot un aizpildīt tabulu `products`

Tabulas `products` lauki

Column Name	#	Data Type	Length	Not Null	Auto Increment
123 id	1	INTEGER		[v]	[v]
A-Z name	2	TEXT		[v]	[ ]
123 price	3	REAL		[v]	[ ]
A-Z image	4	TEXT		[ ]	[ ]

Tabulas `products` ieraksti

123 id	A-Z name	123 price	A-Z image
1	Zīmulis	0,55	pencil.jpg
2	Pildspalva	1,20	pen.jpg
3	Burtnīca	2,94	notebook.jpg
4	Mape	1.55	folder.jpg

Flask + SQLite

3. Mapē `miniVeikals/static/` izveidot jaunu mapi `images`
4. Mapē `miniVeikals/static/images` izveidot jaunu mapi `products`
5. Mapē `miniVeikals/static/images/products` iekopēt produktu attēlus


folder.jpg


notebook.jpg


pen.jpg


pencil.jpg

Flask + SQLite

6. Failā `miniVeikals/app.py` papildus importēt `sqlite3, pathlib` un izveidot jaunu funkciju `get_db_connection()`

```
import sqlite3
from pathlib import Path
```

```
def get_db_connection():
    """
    Izveido un atgriež savienojumu ar SQLite datubāzi.
    """
    # Atrod ceļu uz datubāzes failu (tas atrodas tajā pašā mapē, kur šis fails)
    db = Path(__file__).parent / "miniVeikals.db"

    # Izveido savienojumu ar SQLite datubāzi
    conn = sqlite3.connect(db)

    # Nodrošina, ka rezultāti būs pieejami kā vārdnīcas (piemēram: product["name"])
    conn.row_factory = sqlite3.Row

    # Atgriež savienojumu
    return conn
```

Flask + SQLite

7. Failā `miniVeikals/app.py` papildināt funkciju `products()`

```python
# Maršruts, kas atbild uz pieprasījumu /produkti
@app.route("/produkti")
def products():
    conn = get_db_connection()  # Pieslēdzas datubāzei

    # Izpilda SQL vaicājumu, kas atlasa visus produktus
    products = conn.execute("SELECT * FROM products").fetchall()

    conn.close()  # Aizver savienojumu ar datubāzi

    # Atgriežam HTML veidni "products.html", padodot produktus veidnei
    return render_template("products.html", products=products)
```

Flask + SQLite

8. Failā `miniVeikals/templates/products.html`

```
{% extends "base.html" %}
{% block title %}Produkti - MiniVeikals{% endblock %}
{% block content %}

<h2>Produkti</h2>
<div class="products-list">
    {% for product in products %}
    <a class="products-item" href="{{ url_for('products_show', product_id=product['id']) }}">
        <img src="{{ url_for('static', filename='images/products/' ~ product['image']) }}"
                                                                alt="{{ product['name'] }}">
        <div class="product-name">{{ product["name"] }}</div>
        <div class="product-price">{{ product["price"] }} €</div>
    </a>
    {% endfor %}
</div>

{% endblock %}
```

jinja2: cikls "skrien" cauri sarakstam **products**, kas tika padots no maršruta @app.route("/produkti")

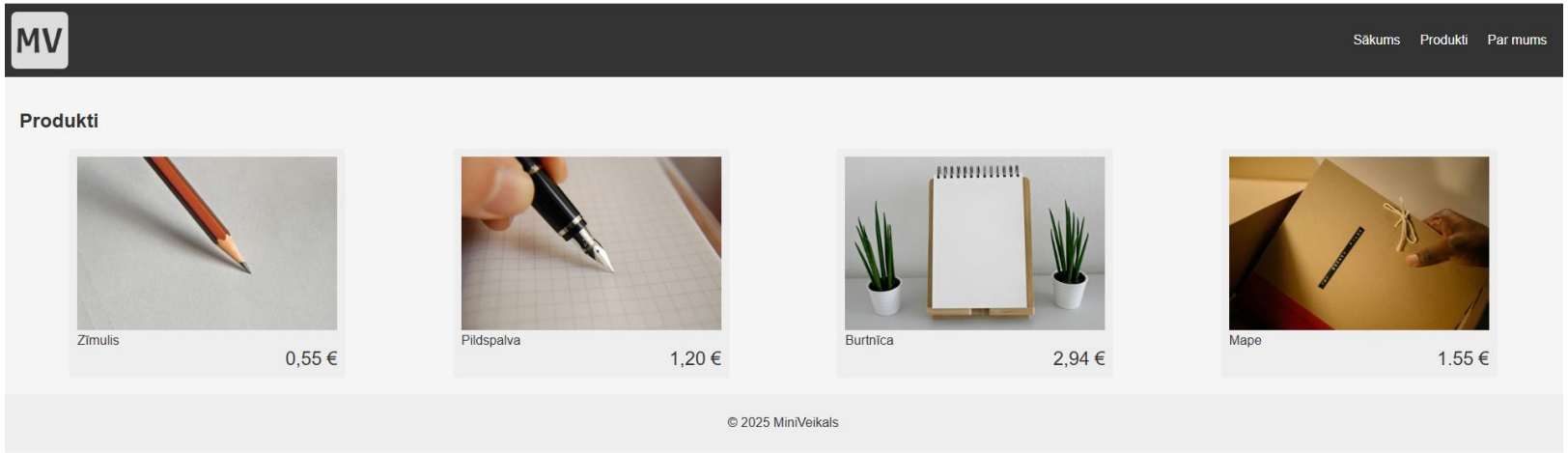
jinja2: izvada katra **product** datus

Flask + SQLite

9. Papildināt `miniVeikals/static/style.css`

```css
/* products-list */
.products-list {
    display: flex;
    gap: 10px;
    justify-content: space-around;
    flex-wrap: wrap;
}
.products-item {
    max-width: 320px;
    background-color: #EEE;
    padding: 10px;
    border-radius: 5px;
    color: #333;
    text-decoration: none;
}
.products-item:hover {
    background-color: #DDD;
}
.products-item .product-price {
    text-align: right;
    font-size: 23px;
}
.products-item img {
    max-width: 100%;
}
```

10. Pārbaudīt `http://127.0.0.1:5000/produkti`



Flask + SQLite

11. Failā `miniVeikals/app.py` izveidot jaunu maršrutu

```
# Maršruts, kas atbild uz pieprasījumu, piemēram: /produkti/3
# Šeit <int:product_id> nozīmē, ka URL daļā gaidāms produkta ID kā skaitlis
@app.route("/produkti/<int:product_id>")
def products_show(product_id):
    conn = get_db_connection()  # Pieslēdzas datubāzei

    # Izpilda SQL vaicājumu, kurš atgriež tikai vienu produktu pēc ID
    product = conn.execute(
        "SELECT * FROM products WHERE id = ?",
        (product_id,),
    ).fetchone()
    # ? ir vieta, kur tiks ievietota vērtība – šajā gadījumā product_id

    conn.close()  # Aizver savienojumu ar datubāzi

    # Atgriežam HTML veidni 'products_show.html', padodot konkrēto produktu veidnei
    return render_template("products_show.html", product=product)
```

Flask + SQLite

12. Izveidot jaunu failu `miniVeikals/templates/products_show.html`

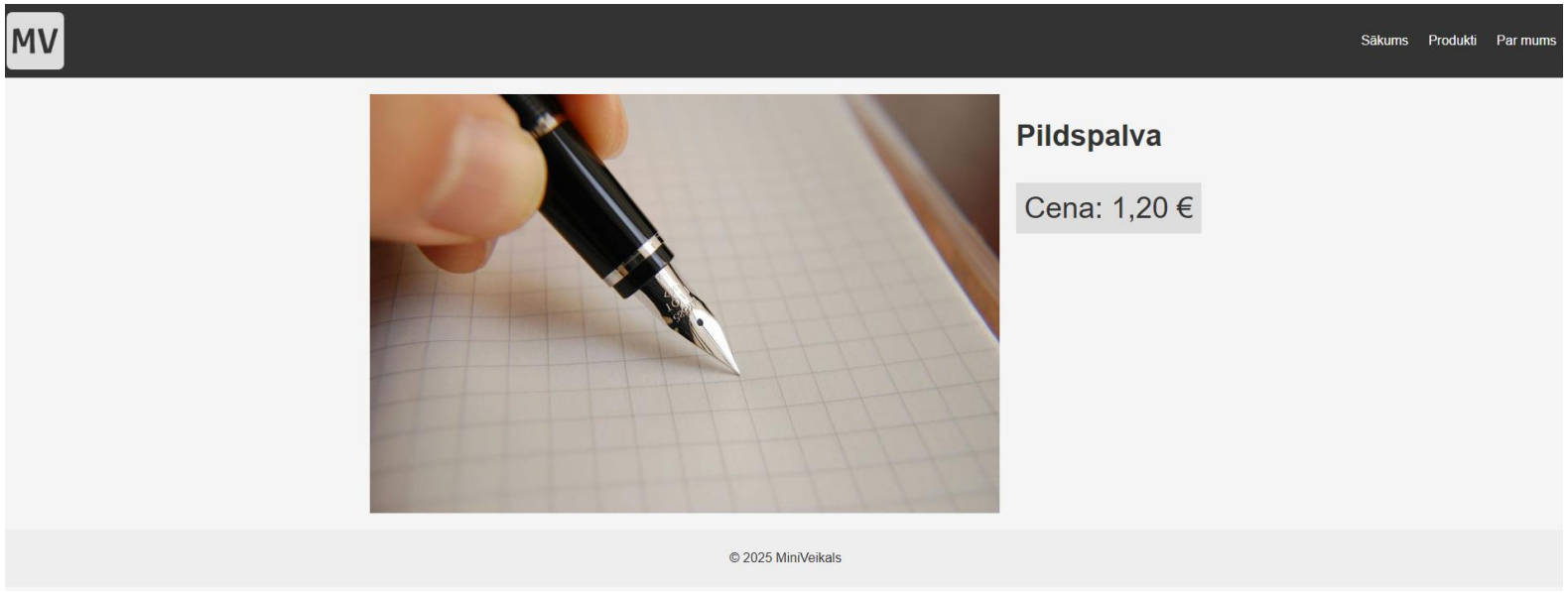
```
{% extends "base.html" %}
{% block title %}{{product.name}} - MiniVeikals{% endblock %}
{% block content %}

<div class="products-show">
    <img src="{{ url_for('static', filename='images/products/' ~ product['image']) }}"
                                                    alt="{{ product['name'] }}">
    <div>
        <h2>{{product["name"]}}</h2>
        <p>Cena: {{product["price"]}} €</p>
    </div>
</div>

{% endblock %}
```

Flask + SQLite

13. Papildināt `miniVeikals/static/style.css`

```css
/* products-show */
.products-show {
    display: flex;
    justify-content: center;
    gap: 20px;
}
.products-show h2{
    font-size: 36px;
}
.products-show p{
    font-size: 36px;
    background-color: #DDD;
    padding: 10px;
}
```

14. Pārbaudīt `http://127.0.0.1:5000/produkti/2`



Flask + SQLite

15. Failā `miniVeikals/templates/products.html`

```
<a class="products-item" href="#">
```

aizstāt ar

```
<a class="products-item" href="{{ url_for('products_show', product_id=product.id) }}">
```

url_for('products_show', product_id=product.id) veido **URL** uz noteiktu maršrutu, kurš ir definēts `app.py` failā.
Piemēram, ja product.id=2, tad tiek izveidots URL **`http://127.0.0.1:5000/produkti/2`**

```
@app.route("/produkti/<int:product_id>")
def products_show(product_id):
```

vērtība **2** tiek padota funkcijas `products_show` parametram `product_id`

Flask + SQLite

Projekta MiniVeikals failu struktūra