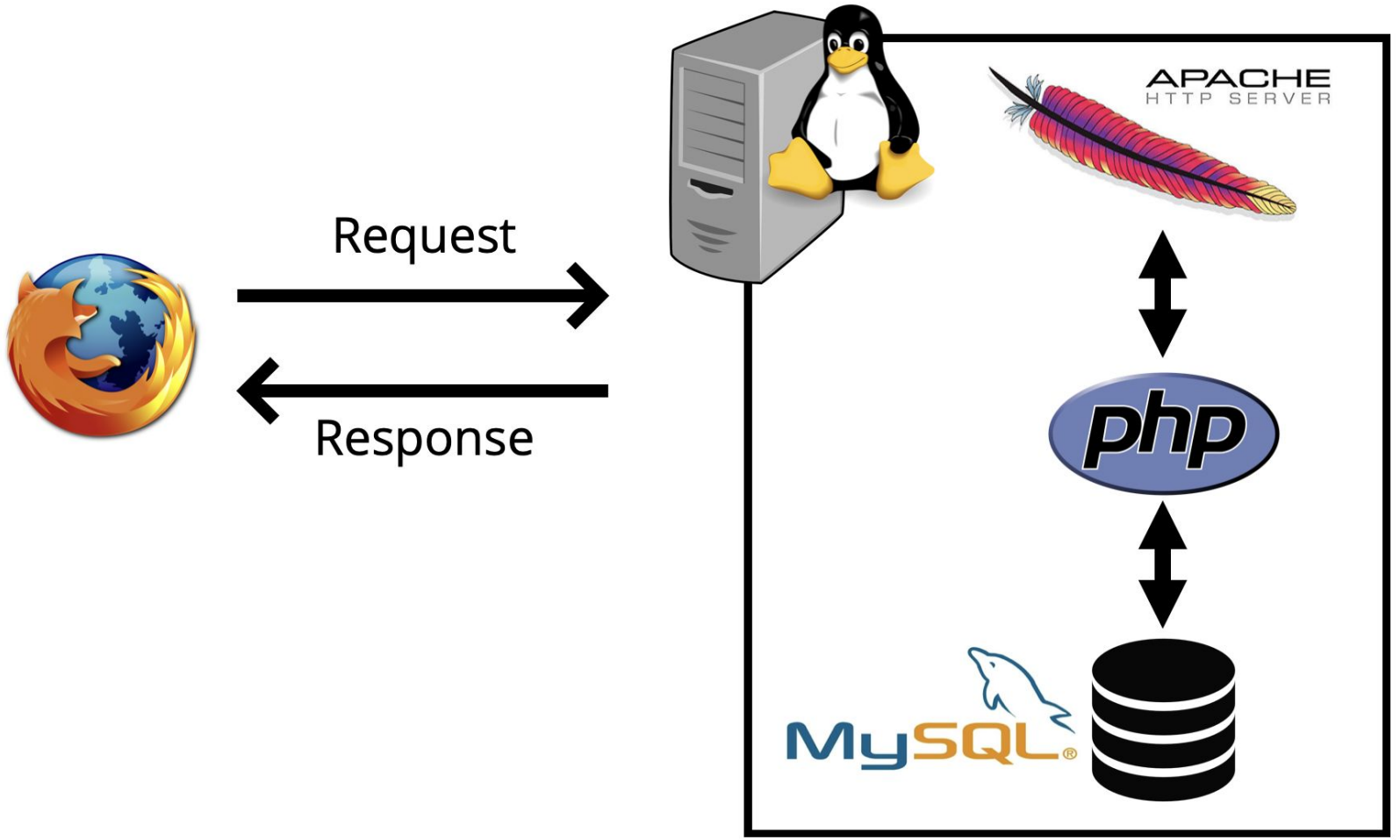
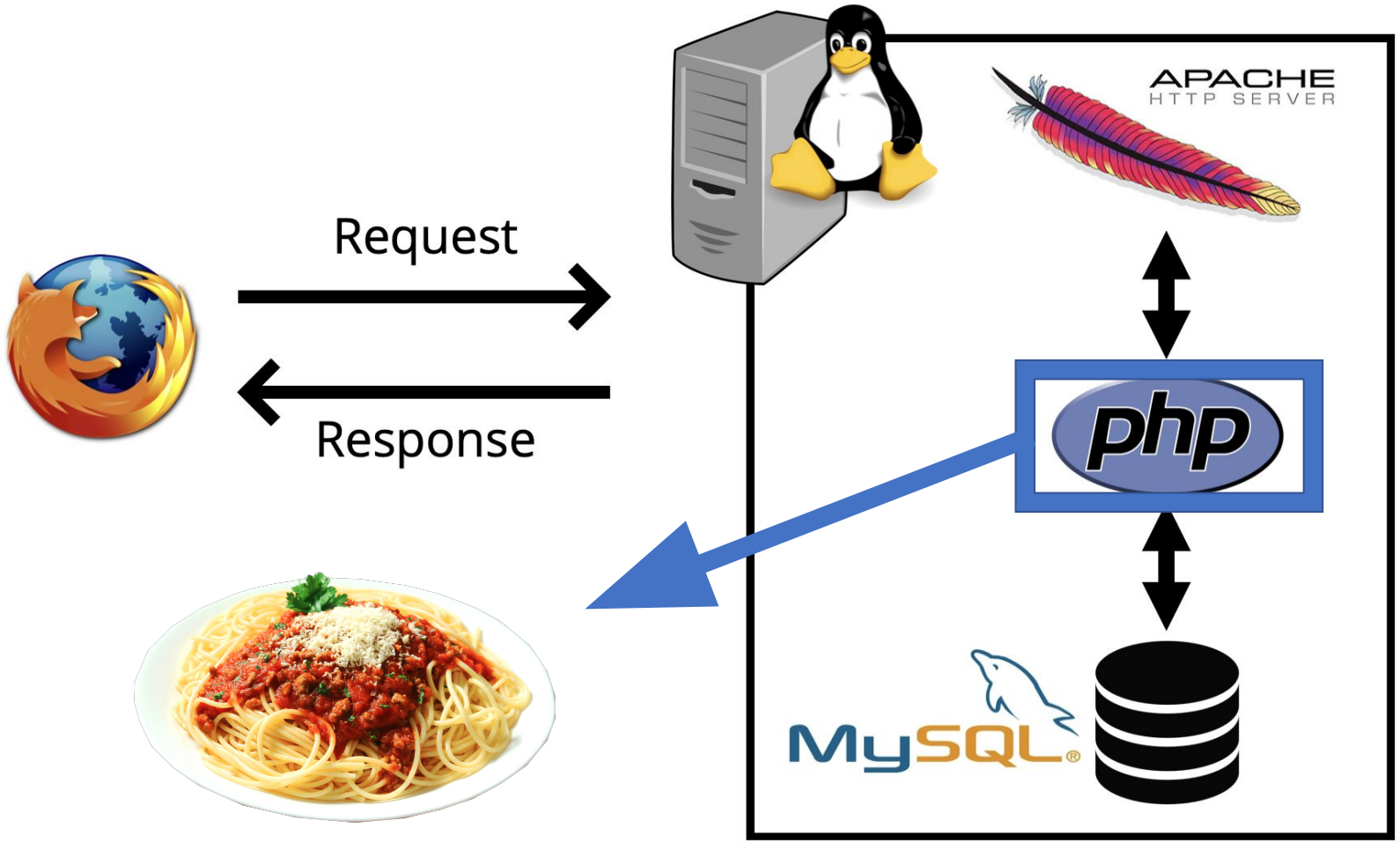
Developing a web application using Python & Flask

Pau Andrio [pau.andrio@bsc.es]

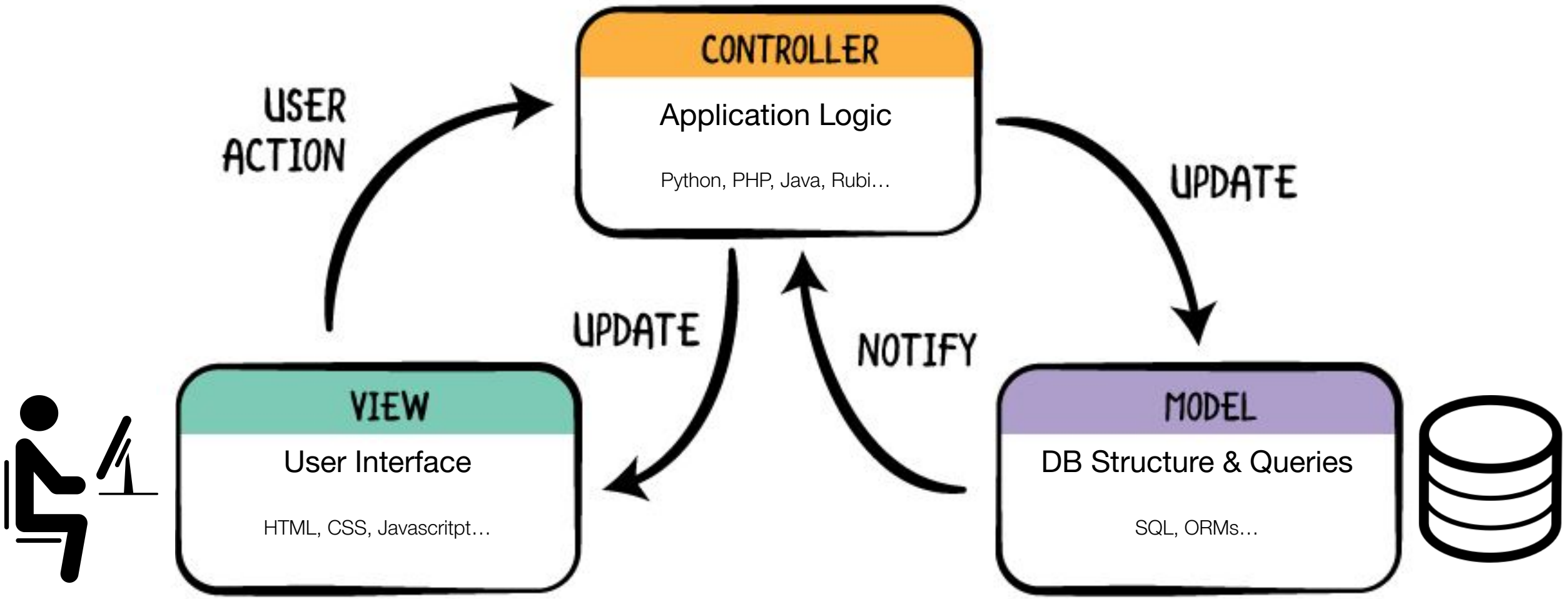
LAMP: Linux, Apache, MySQL, PHP



LAMP: Linux, Apache, MySQL, PHP



MVC: Model View Controller

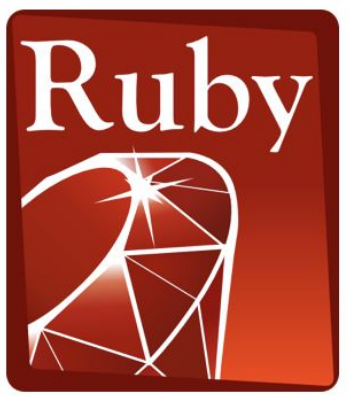


Questions

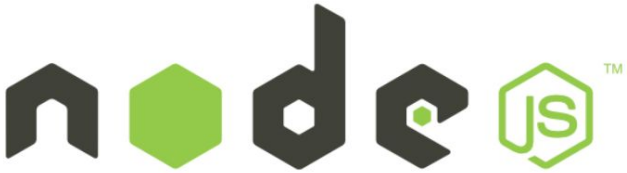
Please answer the following questions in the chat using raising your hands.

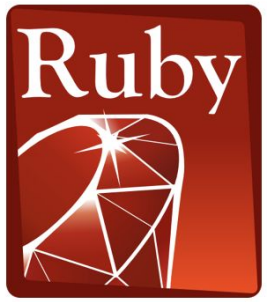
1. Are you familiar with the MVC software design pattern?
2. Do you understand the benefits of dividing the code in components?
3. Have you ever heard the term Spaghetti code before?
4. How many of you have coded anything in any programming language before?

(Big) **Frameworks**



(Micro μ) **Frameworks**





Questions

Please answer the following questions in the chat using raising your hands.

1. Have you ever used a developing framework of any kind?

2. Are you going to use PHP/Slim for your project?

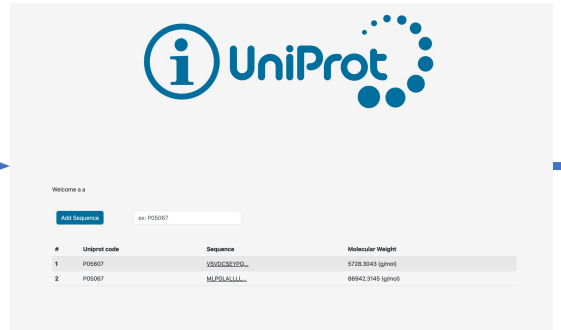
3. Are you going to use Python/Flask for your project?

python

Flask

web development,
one drop at a time

Sign Up



Home

Login

User Space

Sequence

Questions

Please answer the following questions in the chat using raising your hands.

1. Do you understand what uniprotinfo does?

Download & Execute UniprotInfo

Download and install Miniconda

- `[Only LINUX] wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda.sh`
  `[Only MACOS] wget https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh -O ~/miniconda.sh`
- `bash ~/miniconda.sh -b -p $HOME/miniconda`
- `~/miniconda/bin/conda init ${SHELL##*/}  # bash or zsh`

Create an environment install depencies and activate it

- `conda create -n uniprotinfoenv -c conda-forge python==3.11.0 biopython email-validator flask flask-bcrypt flask-login flask-sqlalchemy flask-wtf git`
- `conda activate uniprotinfoenv`

Clone the repository and change directory

- `git clone git@github.com:PauAndrio/uniprotinfo.git`
- `cd uniprotinfo`

Create DB and launch the test server

- `python -c "from app import db, app; app.app_context().push(); db.create_all();"`
- `python app.py`

Test the app using your favorite browser

- `http://localhost:5000/`

The view: base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}{% endblock %}</title>
    <link rel="icon" type="image/x-icon" href="{{ url_for('static', filename='img/favicon.png') }}">
    <!-- Include Bootstrap -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD" crossorigin="anonymous">
    <!-- Include custom CSS-->
    <link href="{{ url_for('static', filename='css/custom.css') }}" rel="stylesheet">
</head>
<body>
    <div class="container-md d-grid gap-5">
        <div class="row mb-5">
            <div class="text-center">
                <a href="{{ url_for('home') }}"><img class="logo" src="{{ url_for('static', filename='img/Infobox_info_icon.svg') }}" /></a>
                <a href="{{ url_for('home') }}"><img class="logo" src="{{ url_for('static', filename='img/uniprot-logo.img.svg') }}" /></a>
            </div>
        </div>
        {% block body %}
        {% endblock %}
    </div>
</body>
</html>
```

Why call url_for instead of a plain URL?

{{ Python Code }}
{% Jinja2 Code %}

Here goes the code defined in other templates

The view: home.html

```
{% extends 'base.html' %}
{% block title %}Home{% endblock %}

{% block body %}
<div class="row mb-5">
    <div class="text-center">
        <a class="btn btn-primary btn-lg me-5" href="{{ url_for('login') }}" role="button">Login</a>
        <a class="btn btn-primary btn-lg" href="{{ url_for('signup') }}" role="button">Sign up</a>
    </div>
</div>
<div class="row">
    <div class="text-center">
        <img class="logo-sm" src="{{ url_for('static', filename='img/github-mark.svg') }}" />
        <a class="link-primary mt-2"
            href="https://github.com/PauAndrio/uniprotinfo">https://github.com/PauAndrio/uniprotinfo</a>
    </div>
</div>

...

{% endblock %}
```

The view: login.html

```
{% extends 'base.html' %}
{% block title %}Login{% endblock %}

{% block body %}
<form class="form-signin text-center" action="{{ url_for('login') }}" method="post">
 {{ form.hidden_tag() }}
   <h1 class="h3 mb-3 fw-normal">Please Log in</h1>
   <div class="form-floating mb-1">
     {{ form.email(class_="form-control") }}
     <label for="email">Email address</label>
     {% set emailerror = form.errors.get('email') %}
       {% if emailerror %}
          <div class="text-danger h6">{{ emailerror[0] }}</div>
       {% endif %}
 </div>
 <div class="form-floating mb-3">
     {{ form.password(class_="form-control") }}
     <label for="password">Password</label>
 </div>
 {% if loginerror %}
    <div class="text-danger h6">{{ loginerror }}</div>
 {% endif %}
   {{ form.submit(class_="btn btn-lg btn-primary") }}
</form>
{% endblock %}
```

{% ... %} for Statements

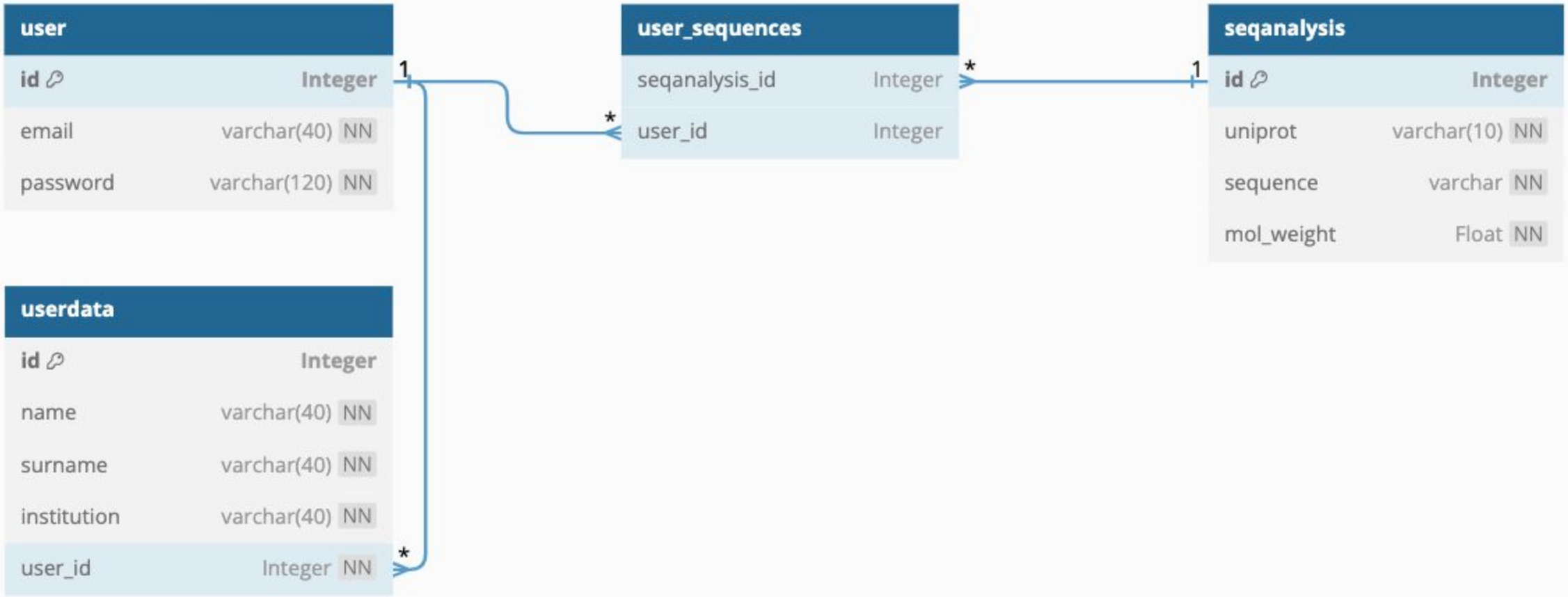
{{ ... }} for Expressions to print to the template output

The view: seqanalysis.html

```
{% extends 'base.html' %}
{% block title %}Sequence Analysis{% endblock %}

{% block body %}
<div class="container-md">
    <div class="row">
        <div class="col-2 h5">Uniprot Code:</div>
        <div class="col-2 mb-5">{{ seqanalysis.uniprot }}</div>
    </div>
    <div class="row">
        <div class="col-2 h5">Molecular Weight:</div>
        <div class="col-2 mb-5">{{ "%.4f"|format(seqanalysis.mol_weight) }} (g/mol)</div>
    </div>
    <div class="row">
        <div class="col-2 h5">Sequence:</div>
        <div class="col-8 text-wrap break-all">{{ seqanalysis.sequence }}</div>
    </div>
</div>
{% endblock %}
```

The Database Diagram



**user_sequences is an auxiliary table to implement a many-to-many relation*

The Database Diagram: SQL version

```sql
CREATE TABLE user_sequences (
    seqanalysis_id INTEGER,
    user_id INTEGER,
    FOREIGN KEY(seqanalysis_id) REFERENCES seqanalysis(id),
    FOREIGN KEY(user_id) REFERENCES user(id)
);

CREATE TABLE user (
    id INTEGER PRIMARY KEY UNIQUE,
    email VARCHAR(40) NOT NULL UNIQUE,
    password VARCHAR(120) NOT NULL
);
```

```sql
CREATE TABLE userdata (
    id INTEGER PRIMARY KEY UNIQUE,
    name VARCHAR(40) NOT NULL,
    surname VARCHAR(40) NOT NULL,
    institution VARCHAR(40) NOT NULL,
    user_id INTEGER NOT NULL,
    FOREIGN KEY(user_id) REFERENCES user(id)
);

CREATE TABLE seqanalysis (
    id INTEGER PRIMARY KEY UNIQUE,
    uniprot VARCHAR(10) NOT NULL,
    sequence TEXT NOT NULL,
    mol_weight REAL NOT NULL
);
```

The Database Python Model

```python
import sqlite3
conn = sqlite3.connect('my_database.db')
cursor = conn.cursor()

class User:
    def __init__(self, id, email, password):
        self.id = id
        self.email = email
        self.password = password


def get_user(email, cursor):
    cursor.execute("SELECT * FROM user WHERE email=?", (email,))
    user_row = cursor.fetchone()
    return User(id=user_row[0], email=user_row[1], password=user_row[2])

def get_userdata(user_id, cursor):
    cursor.execute("SELECT * FROM userdata WHERE user_id=?", (user_id,))
    userdata_row = cursor.fetchone()
    return Userdata(id=userdata_row[0], name=userdata_row[1], surname=userdata_row[2], institution=userdata_row[3], user_id=userdata_row[4])

def get_seqanalysis(id, cursor):
    cursor.execute("SELECT * FROM seqanalysis WHERE id=?", (id,))
    seqanalysis_row = cursor.fetchone()
    return Seqanalysis(id=seqanalysis_row[0], uniprot=seqanalysis_row[1], sequence=seqanalysis_row[2], mol_weight=seqanalysis_row[3])
```

```python
class Userdata:
    def __init__(self, id, name, surname, institution, user_id):
        self.id = id
        self.name = name
        self.surname = surname
        self.institution = institution
        self.user_id = user_id


class Seqanalysis:
    def __init__(self, id, uniprot, sequence, mol_weight):
        self.id = id
        self.uniprot = uniprot
        self.sequence = sequence
        self.mol_weight = mol_weight
```

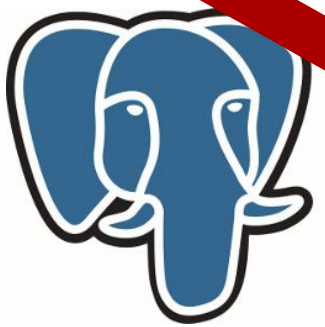
The Database Diagram: SQL version

```
CREATE TABLE use      quences (
    seqanalysis_id INT
    user_id INTEGER,
    FOREIGN KEY(seqanalysis_id)      ENCES seqanalysis(id),
    FOREIGN KEY(user_id) REFERENCES us      d)
);

CREATE TABLE user (
    id INTEGER PRIMARY KEY UNIQUE,
    email VARCHAR(40) NOT NULL UNIQUE,
    password VARCHAR(120) NOT NULL
);
```

```
CREATE TABLE userdata (
    id INTEGER PRIMARY KEY UNIQUE,
    name VARCHAR(40) NOT NULL,
    surname VARCHAR(40) NOT      ,
    institution VARCHAR      NOT NULL,
    user_id INTEG      T NULL,
    FOREIGN      user_id) REFERENCES user(id)
);

CREATE TABLE seqanalysis (
    id INTEGER PRIMARY KEY UNIQUE,
      iprot VARCHAR(10) NOT NULL,
    seq      e TEXT NOT NULL,
    mol_weig      EAL NOT NULL
);
```



The Database Python Model

```
import sqli
conn = sqlite3.co    t('my_database.db')
cursor = conn.cursor()

class User:
    def __init__(self, id, email, passw
        self.id = id
        self.email = email
        self.password = password


def get_user(email, cursor):
    cursor.execute("SELECT * FROM user WHERE email=?", (email,))
    user_row = cursor.fetchone()
    return User(id=user_row[0], email=user_row[1], pas    =user_row[2])

def get_userdata(user_id, cursor):
    cursor.execute("SELECT * FROM user     WHERE user_id=?", (user_id,))
    userdata_row = cursor.fetchon
    return Userdata(id=user     ow[0], name=userdata_row[1], surname=userdata_row[2], institution=userdata_row[3]     r_id=userdata_row[4])

def get_seqanal     d, cursor):
    cursor.e     e("SELECT * FROM seqanalysis WHERE id=?", (id,))
    se     ysis_row = cursor.fetchone()
    return Seqanalysis(id=seqanalysis_row[0], uniprot=seqanalysis_row[1], sequence=seqanalysis_row[2], mol_weight=seqanalysis_row[3])
```

```
class Userdata:
    def __init__(self, id, name, surname, institut     user_id):
        self.id = id
        self.name = name
        self.surname = surname
        self.institution = in     tion
        self.user_id = u     d

class Seqanal     :
    def     __init__(self, id, uniprot, sequence, mol_weight):
        self.id = id
        self.uniprot = uniprot
        self.sequence = sequence
        s     ol_weight = mol_weight
```

The Model: model.py

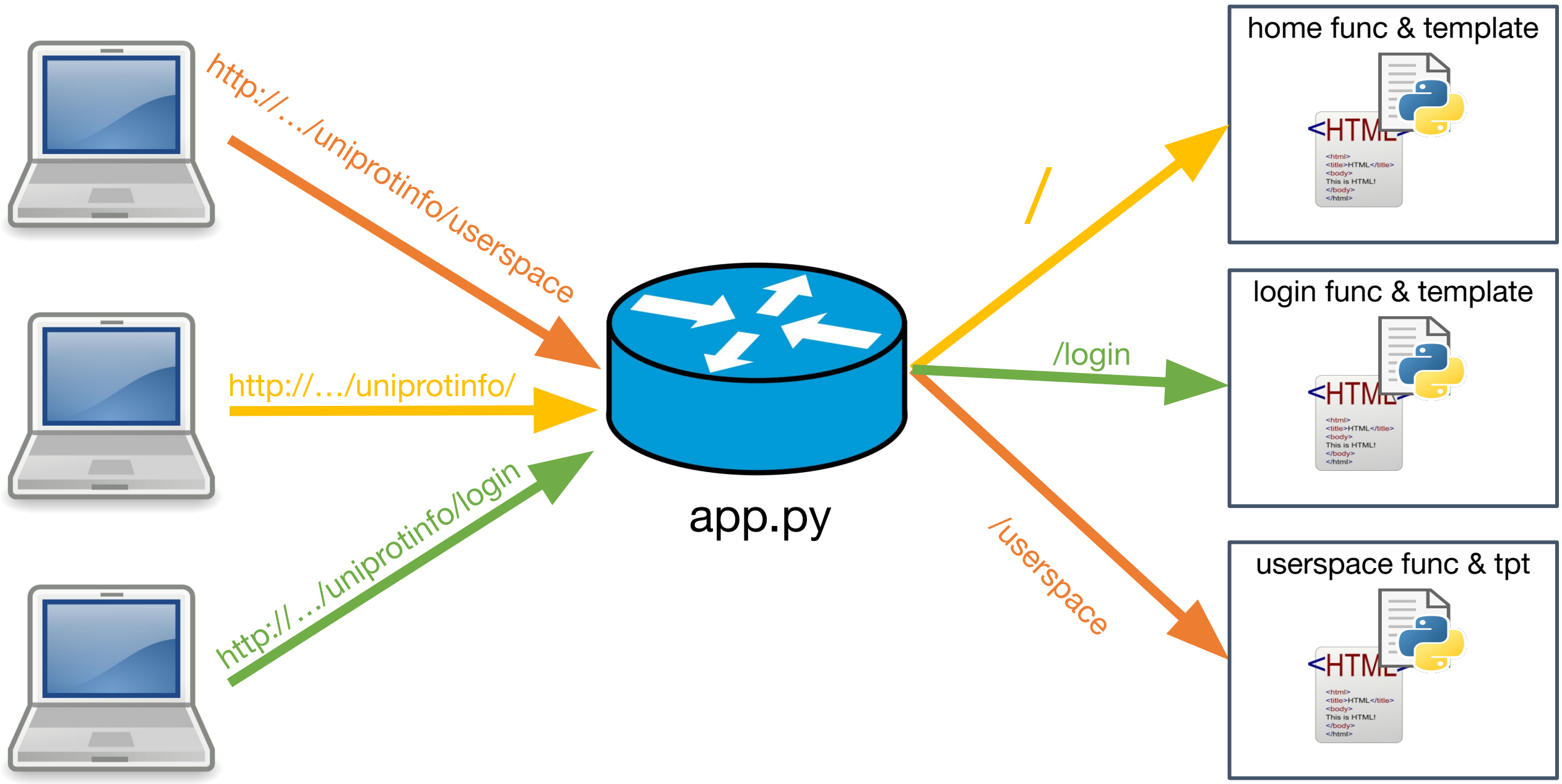
```
db = SQLAlchemy()
user_sequences = db.Table('user_sequences',
        db.Column('seqanalysis_id', db.Integer, db.ForeignKey('seqanalysis.id')),
        db.Column('user_id', db.Integer, db.ForeignKey('user.id')))

class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True, unique=True)
    email = db.Column(db.String(40), nullable=False, unique=True)
    password = db.Column(db.String(120), nullable=False)
    user_data = db.relationship('Userdata', backref='User')
    sequences = db.relationship('Seqanalysis', secondary=user_sequences, backref='users')

class Userdata(db.Model):
    id = db.Column(db.Integer, primary_key=True, unique=True)
    name = db.Column(db.String(40), nullable=False)
    surname = db.Column(db.String(40), nullable=False)
    institution = db.Column(db.String(40), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

class Seqanalysis(db.Model):
    id = db.Column(db.Integer, primary_key=True, unique=True)
    uniprot = db.Column(db.String(10), nullable=False)
    sequence = db.Column(db.String, nullable=False)
    mol_weight = db.Column(db.Float, nullable=False)
```

The controller: Router + Business

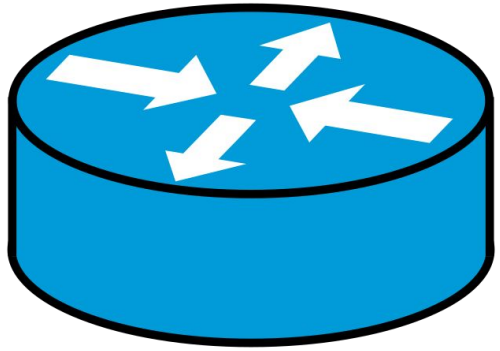


The controller: Router

```python
@app.route('/seqanalysis/<seqanalysis_id>')
@login_required
def seqanalysis(seqanalysis_id):
    seqanalysis = db.session.get(Seqanalysis, (int(seqanalysis_id)))
    if seqanalysis in current_user.sequences:
        return render_template('seqanalysis.html', seqanalysis=seqanalysis)
    else:
        return redirect(url_for('userspace'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    loginerror = None
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user:
            if (form_password := form.password.data):
                if bcrypt.checkpw(form_password.encode('utf8'), user.password):
                    login_user(user)
                    return redirect(url_for('userspace'))
        loginerror = "Invalid email or password."
    return render_template('auth/login.html', form=form, loginerror=loginerror)
```

```python
@app.route('/')
def home():
    return render_template('home.html')
```



Business: uniprot_api

```python
def get_unipro_info_tuple(uniprot_accession_code):
    sequence = get_sequence(uniprot_accession_code)
    mol_weight = ProteinAnalysis(sequence).molecular_weight()
    return uniprot_accession_code, sequence, mol_weight

def get_sequence(uniprot_accession_code):
    url = f'https://rest.uniprot.org/uniprotkb/stream?compressed=false&format=json&query=accession={uniprot_accession_code}&fields=sequence'
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        results = data.get('results')
        if results:
            first_result = results[0]
            sequence = first_result.get('sequence')
            if sequence:
                return sequence.get('value')

    return None
```

Bring out your dead!