FLOSScaster Project Demonstration

Brychcy, Patryk & Filatoff, Michael & Fluegel, Dwipa & Gavrilov, Sascha & Karaman, Deniz & Lepore, Dominik & Zimmermann, Alwin & Zundel, Benedikt

Frankfurt University of Applied Sciences

Friday, 2025-05-23

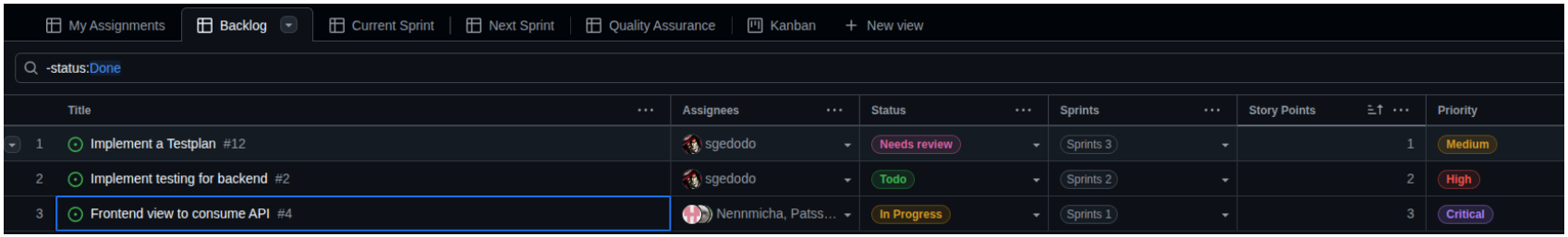
Agenda

- Projektmanagement mit Scrum & GitHub Projects (Deniz Karaman)
- Mockup-Design und Gestaltung (Patryk Brychcy & Michael Filatoff)
- Technische Dokumentation
  - Frontend (Michael Filatoff)
    - Styling (Sascha Gav)
  - Backend (Benedikt Zundel)
  - Dockerization (Benedikt Zundel)
  - Deployment (Benedikt Zundel)
  - RSS-Feed Generator (Alwin Zimmermann)
  - Publikation auf einem Social Media Feed (Dwipa Flügel)
- Testbericht (Dominik Lepore)
  - Static Code-Analyse (Deniz Karaman)

Projektmanagement mit Scrum & GitHub Projects (Deniz Karaman)

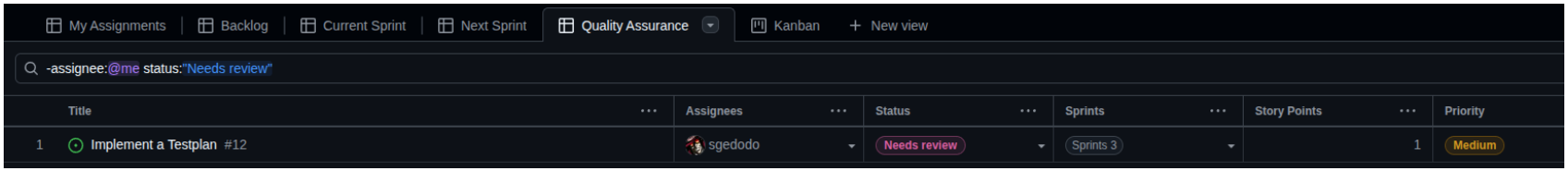
Projektmanagement mit Scrum & GitHub Projects

Figure: Screenshot der GitHub Projects Ansicht des Backlogs



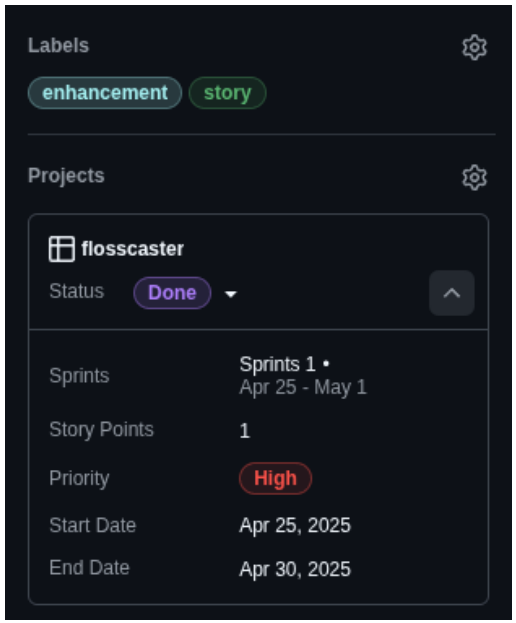
Projektmanagement mit Scrum & GitHub Projects

Figure: Screenshot der GitHub Projects Ansicht der Quality Assurance



Projektmanagement mit Scrum & GitHub Projects

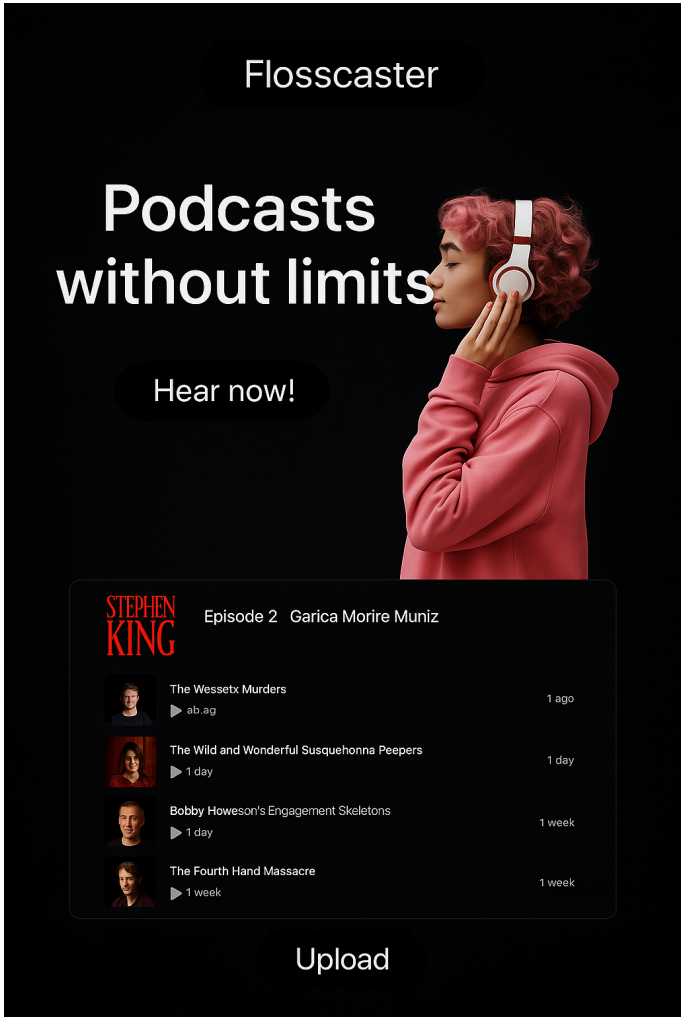
Figure: Screenshot der GitHub Ansicht einer Story



Mockup-Design und Gestaltung (Patryk Brychcy & Michael Filatoff)

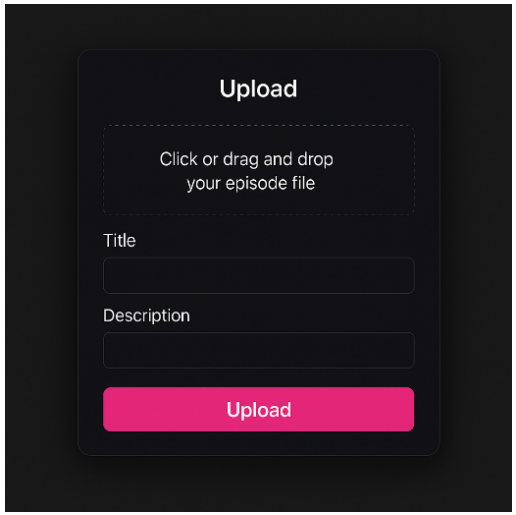
Mockup-Design und Gestaltung

Figure: Figma Design der Landingpage



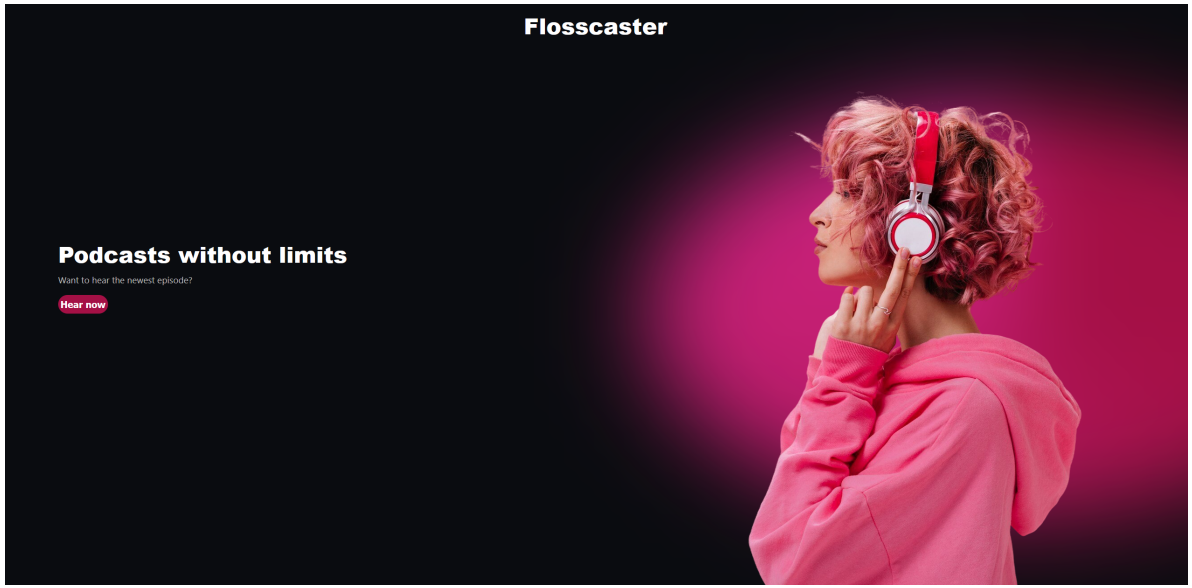
Mockup-Design und Gestaltung

Figure: Figam Design des Upload-Dialogs



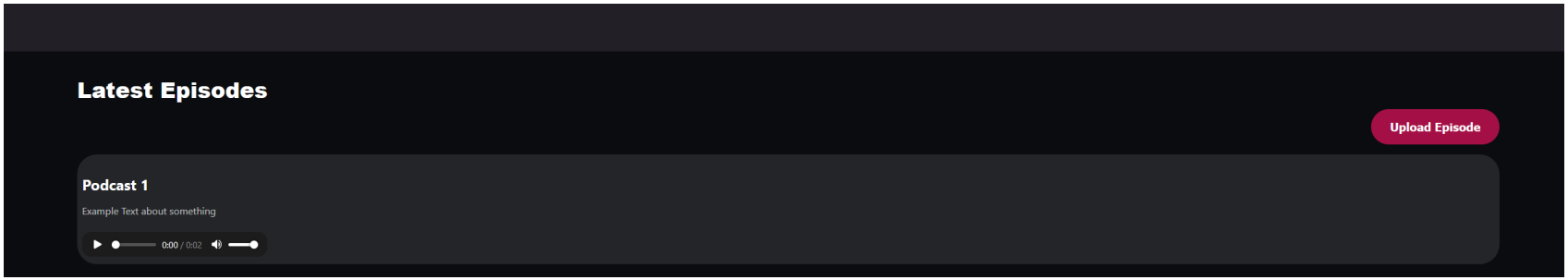
Mockup-Design und Gestaltung

Figure: Implementiertes Design der Landingpage



Mockup-Design und Gestaltung

Figure: Implementiertes Design der Podcast-Episoden



Technische Dokumentation

Frontend (Michael Filatoff)

Frontend

Listing 1: Implementation des Hero-Bilds

```
1   <img className={styles.mainPicture} src="./assets/hero\_picture.png" alt="Hero
        Section"/>
```

Frontend

Listing 2: Implementation der Navbar

```
<a href="/" className={styles.mainHeader}>
  Flosscaster
</a>
```

Styling (Sascha Gav)

Styling

Backend (Benedikt Zundel)

Backend

Listing 3: Implementation der Podcast-Episoden Klasse

```
@dataclass
class Podcast:
    id: int
    title: str
    description: str
    date: str
    filepath: str
```

Backend

- `/api/list`
- `/api/create`
- `/api/get_upload/<path>`
- `/rss`

Dockerization (Benedikt Zundel)

Dockerization

Deployment (Benedikt Zundel)

Deployment

RSS-Feed Generator (Alwin Zimmermann)

RSS-Feed Generator

Listing 4: Feedparser Bug

```
def load_existing_feed(self):
  """Laedt den bestehenden RSS-Feed, falls vorhanden."""
  if os.path.exists(self.feed_file_path):
      feed = feedparser.parse(self.feed_file_path)
      for entry in feed.entries:
          item = {
              'title': entry.title,
              'description': entry.description,
              'date': entry.published if 'published' in entry else
    datetime.now().isoformat(),
              'enclosure_url': entry.enclosure.href if 'enclosure' in entry
    else None,
              'enclosure_type': entry.enclosure.type if 'enclosure' in entry
    else None,
              'enclosure_length': entry.enclosure.length if 'enclosure' in
    entry else None
          }
          self.items.append(item)
          print(self)
```

RSS-Feed Generator

Listing 5: RSS-Feed Generator imports

```
1  import os
2  import argparse
3  import datetime
4  import pytz
5  from lxml import etree
```

RSS-Feed Generator

Listing 6: Erstellen eines neuen Feed-Items

```
# Erstelle ein neues Item
new_item = etree.Element('item')
etree.SubElement(new_item, 'title').text = title
etree.SubElement(new_item, 'description').text = description
etree.SubElement(new_item, 'pubDate').text = new_episode_pubDate
enclosure = etree.SubElement(new_item, 'enclosure')
enclosure.set('url', url)
enclosure.set('length', length)
enclosure.set('type', 'audio/mpeg')
```

Publikation auf einem Social Media Feed (Dwipa Flügel)

Publikation auf einem Social Media Feed

Listing 7: Vollständige Funktion des *Autotootings*

```python
def masttoot(title: str, url: str, description: str):
    """Publishes a new toot using the input, also sends a reply to the
     announcement with the description."""

    post_text = "The Flosscasters have released a new podcast episode: " +
     title + ". Check it out @ " + url

    #Mastodon-Instanz
    mastodon = Mastodon(
        client_id = os.getenv("MASTODON_CLIENT_KEY"),
        client_secret = os.getenv("MASTODON_CLIENT_SECRET"),
        access_token = os.getenv("MASTODON_ACCESS_TOKEN"),
        api_base_url = os.getenv("MASTODON_BASE_URL")
    )

    #Publish the toot and reply to it with the description
    to_reply = mastodon.status_post(post_text)
    mastodon.status_post(description, in_reply_to_id=to_reply.id)
```

Testbericht (Dominik Lepore)

Testbericht

Listing 8: Beispiel einer Fixture

```
@pytest.fixture()
def init_testing():
    app.config.update({'TESTING': True})
    if not os.path.exists(DATABASE_FILE):
        con = sqlite3.connect(DATABASE_FILE)
        cur = con.cursor()
        cur.execute("CREATE TABLE IF NOT EXISTS podcasts(id INTEGER PRIMARY
     KEY, title, description, date, filepath)")
        con.close()
    yield app
    app.config.update({'TESTING': False})
```

Testbericht

Listing 9: Test-Implementation von /api/create

```
@pytest.mark.parametrize("title, description, audio", [
    ("First FLOSScast", "The very first flosscast", generate_mp3_tts("Hello to the very first flosscast episode.", lang="en")),
    (generate_random_ascii(length=10), generate_random_ascii(length=99), generate_mp3_size(1024)),
    (generate_random_ascii(length=24, letters=False, numbers =False, punctuation=True), generate_random_ascii(length=240, punctuation=True), generate_mp3_duration(30))
])


def test_create_podcasts(client, title, description, audio):
    response = client.post("/api/create", content_type="multipart/form-data",
                           data={
        "title": title,
        "description": description,
        "audio": open(audio, 'rb')
    })
    assert response.status_code == 200
```

Testbericht

Listing 10: Implementation der `client`-Fixture

```
1  @pytest.fixture()
2  def client(init_testing):
3      with app.test_client() as client:
4          yield client
```

Testbericht

Listing 11: Test-Implementation von /api/list

```
1  def test_list(database_entry_id, client):
2      response = client.get("/api/list")
3      assert response.status_code == 200
4      assert f"\"id\":{database_entry_id}".encode("utf8") in response.data
5      assert b"\"title\":" in response.data
6      assert b"\"date\":" in response.data
7      assert b"\"description\":" in response.data
8      assert b"\"filepath\":" in response.data
```

Testbericht

Listing 12: Test-Implementation von `/api/list` gegen eine leere Datenbank

```
1  def test_list_empty_db(empty_database, client):
2      response = client.get("/api/list")
3      assert response.status_code == 200
4      assert b'[]' in response.data
```

Testbericht

Listing 13: Test-Implementation von /api/get_upload

```
1  def test_getFileByFilename(upload_file, client):
2      response = client.get(f"/api/get_upload/{upload_file}")
3      assert response.status_code == 200
4
5      assert "audio/mpeg" in response.headers["content-type"]
```

Testbericht

Listing 14: Test-Implementation von `/rss`

```
1  def test_getrss(create_rss, client):
2      response = client.get("/rss")
3      assert "application/xml" in response.headers["content-type"]
4      assert response.status_code == 200 or response.status_code == 304
```

Testbericht

Listing 15: Fixture zum löschen des RSS-Feeds

```
@pytest.fixture()
def delete_rss(init_testing):
    rss_path = os.getenv("RSS_FILE")
    if os.path.exists(rss_path):
        file = open(rss_path, "r")
        rss_file = file.read()
        file.close()
        os.remove(rss_path)
        return rss_file
    return "failure"
```

Testbericht

Listing 16: Test-Implementation von `/rss` bei nicht vorhandenem Feed

```
1  def test_getrss_no_file(delete_rss, client):
2      response = client.get("/rss")
3      assert response.status_code == 404
4      if not delete_rss == "failure":
5          rss_path = os.getenv("RSS_FILE")
6          with open(rss_path, "w") as file:
7              file.write(delete_rss)
```

Static Code-Analyse (Deniz Karaman)

Static Code-Analyse