Spickzettel: Docker & GUI Testing

Ziel

- Ausführen von automatisierten **GUI-Tests (z. B. mit Selenium, Playwright, PyQt, Electron)** in Containern
- Docker-Container bieten isolierte, reproduzierbare Testumgebungen

Was ist machbar?

- **Headless Testing** (ohne sichtbare Oberfläche) mit:
  - **Selenium (Chrome/Firefox Headless)**
  - **Playwright**
  - **PyQt + xvfb**
  - **Electron + Spectron**
- Verwendung von **xvfb (X Virtual Framebuffer)** zur Emulation einer Anzeige
- **GUI-Apps bauen, starten und testen** – v. a. bei Integrationstests

Was ist schwierig oder nicht empfohlen?

- **Interaktive GUI-Nutzung** in einem echten Desktop ist nicht direkt möglich (kein Window-Manager)
- **Hardwarebeschleunigte Grafik** (GPU) erfordert spezielle Konfiguration (z. B. mit NVIDIA Docker)
- **Zugriff von außen auf UI** nur über Umwege wie VNC oder WebVNC möglich

Beispiel: Selenium in Docker (Headless Chrome)

```
FROM python:3.10-slim
RUN pip install selenium
RUN apt-get update && apt-get install -y chromium-driver chromium

COPY . /tests
WORKDIR /tests
CMD ["python", "test_gui.py"]
# test_gui.py
from selenium import webdriver
options = webdriver.ChromeOptions()
options.add_argument('--headless')
driver = webdriver.Chrome(options=options)
driver.get("https://example.com")
assert "Example" in driver.title
driver.quit()
```

Beispiel: PyQt5 mit Xvfb

```
FROM python:3.10
RUN apt-get update && apt-get install -y xvfb python3-pyqt5
COPY . /app
CMD xvfb-run -a python /app/test_qt.py
```

Beispiel: GUI über VNC sichtbar machen

- Mit Images wie `jlesage/firefox` oder `consol/ubuntu-xfce-vnc`
- Zugriff über Browser via WebVNC (Portfreigabe nötig)

Best Practices

- **Headless first**: Nutze headless Modi, wo möglich (Chrome, Firefox, Playwright)
- Tests in CI mit GitHub Actions, GitLab CI oder Jenkins + Docker kombinieren
- Für komplexe Interaktionen: Use-Cases über Playwright/Selenium mit Netzwerk-Mocks testen
- Zeitverhalten & Ressourcenverbrauch beachten (Headless ist nicht gleich schnell!)