

Zusammenfassung *(automatisiert erstellt)*

Um eine App zu erstellen, hängt stark vom Typ ab: Web-Apps (mit Python Flask oder Django), Desktop-Apps (z.B. Tkinter in Python), mobile Apps (Android, iOS), oder wissenschaftliche Dienste (Docker, XNAT). Für Web-Apps entwickle man eine Grundstruktur mit Programmierframeworks, füge Funktionen hinzu, teste und deploye sie z.B. auf Servern oder in Containern. Desktop-Apps nutzen GUI-Bibliotheken. Mobile Apps erfordern spezifische Sprachen wie Kotlin oder Swift, oder No-Code-Plattformen. Wissenschaftliche „Apps“ im akademischen Umfeld bestehen meist aus Docker-Containern mit APIs. Wichtig sind Versionierung, Dokumentation, Tests und Nachhaltigkeit. Eine schrittweise Vorgehensweise empfiehlt sich immer.

Systemprompt

Du bist ein intelligentes und unterstützendes KI-Assistenzsystem für alle Hochschulangehörigen der HAWK Hildesheim/Holzminden/Göttingen. Dein Ziel ist es, Studierende, Lehrende, Forschende und Mitarbeitende in ihrer akademischen Arbeit, beim Lernen, Forschen, Lehren und verwalterischen Aufgaben zu unterstützen. Dabei förderst du kollaboratives Arbeiten, wissenschaftliches Denken und eine kreative Problemlösung. Beziehe dich auf wissenschaftliche Methoden und Theorien, argumentiere sachlich und reflektiere kritisch. Sei objektiv und verzichte auf unbegründete Meinungen. Fördere akademische Integrität und unterstütze keine Plagiate. Sei inklusiv, wertschätzend und respektiere Vielfalt.

Verlauf:

Bousfiha, Tanae:

```
{
  "name": "osa_predictor",
  "description": "Berechnet das
OSA-Risiko aus einer CSV-, TSV- oder TXT-Datei.",
  "version": "1.0",
  "image":
"tanaebousfiha/osa-predictor:finalfix",
  "type": "docker",
  "command-line": "python
/app/OSA_xnat.py /app/input/#INPUT_FILE#
/app/output",
  "mounts": [
    {
      "name": "output_mount",
      "writable": true,
      "path": "/app/output"
    },
    {
      "name": "input_mount",
      "writable": false,
      "path": "/app/input"
    }
  ],
  "environment-variables": {},
  "ports": {},
  "inputs": [
    {
      "name": "INPUT_FILE",
      "description": "Name der
Eingabedatei",
      "type": "string",
      "required": true,
      "select-values": []
    }
  ],
}
```

```

    "outputs": [
      {
        "name": "result_file",
        "description": "Die berechneten
OSA-Risikodaten",
        "required": true,
        "mount": "output_mount",
        "path": "osa_result.csv"
      }
    ],
    "xnat": [
      {
        "name": "osa_wrapper_final",
        "label": "OSA Predictor",
        "description": "OSA Risk
Calculation.",
        "contexts": [
          "xnat:mrSessionData"
        ],
        "external-inputs": [
          {
            "name": "session",
            "type": "Session",
            "required": true,
            "load-children": true
          }
        ],
        "derived-inputs": [
          {
            "name": "csv_resource",
            "type": "Resource",
            "matcher": "@.label ==
'CSV'",
            "required": true,
            "provides-files-for-command-mount":
"input_mount",
            "load-children": true,
            "derived-from-wrapper-input":

```

```

    &quot;session&quot;;
    &quot;multiple&quot;:: false
  },
  {
    &quot;name&quot;:: &quot;input_file&quot;;
    &quot;type&quot;:: &quot;File&quot;;
    &quot;matcher&quot;:: &quot;@.name =~
/.*\\. (csv|tsv|txt)$/&quot;;
    &quot;required&quot;:: true,
    &quot;load-children&quot;:: true,
    &quot;derived-from-wrapper-input&quot;::
&quot;csv_resource&quot;;
    &quot;multiple&quot;:: false
  },
  {
    &quot;name&quot;:: &quot;input_file_name&quot;;
    &quot;type&quot;:: &quot;string&quot;;
    &quot;required&quot;:: true,
    &quot;provides-value-for-command-input&quot;::
&quot;INPUT_FILE&quot;;
    &quot;user-settable&quot;:: false,
    &quot;load-children&quot;:: false,
    &quot;derived-from-wrapper-input&quot;::
&quot;input_file&quot;;
    &quot;derived-from-xnat-object-property&quot;::
&quot;name&quot;;
    &quot;multiple&quot;:: false
  }
],
&quot;output-handlers&quot;:: [
  {
    &quot;name&quot;:: &quot;osa_output&quot;;
    &quot;accepts-command-output&quot;::
&quot;result_file&quot;;
    &quot;as-a-child-of&quot;:: &quot;session&quot;;
    &quot;type&quot;:: &quot;Resource&quot;;
    &quot;label&quot;:: &quot;OSA Result&quot;;
    &quot;format&quot;:: &quot;csv&quot;;

```

```

    &quot;tags&quot;:: []
  }
]
}
],
&quot;container-labels&quot;:: {},
&quot;generic-resources&quot;:: {},
&quot;ulimits&quot;:: {},
&quot;secrets&quot;:: []
}

```

ist das richtig

HAWKI (gpt-4.1):

Deine JSON-Definition ist ****konzeptionell schlüssig**** und entspricht weitgehend den üblichen Anforderungen für XNAT-Container-Services und -Wrappers. Die Kommandostruktur, Inputs, Outputs sowie die Festlegungen für Mountpoints wirken richtig implementiert. Dennoch empfiehlt sich eine ****kritische Durchsicht mit Fokus auf folgende Aspekte****:

1. Inputs und Mounts

- Die Mountpoints ``input_mount`` und ``output_mount`` sind korrekt eingebunden und mit den richtigen Zugriffsrechten versehen.
- Das Pattern für ``input_file`` (Regulärer Ausdruck) schließt CSV, TSV und TXT sinnvoll ein.
- Die Verbindung der ``input_file_name`` zum Kommando ist schlüssig.

2. Kommandostruktur

- Das Kommando

```

```python
python /app/OSA_xnat.py /app/input/#INPUT_FILE# /app/output
```

```

wird zur Laufzeit korrekt mit dem tatsächlich übergebenen Dateinamen ausgeführt.

3. Output

- Das Output-File heißt immer ``osa_result.csv`` und wird nach ``/app/output`` geschrieben, was zum Output-Handler passt. Falls der Python-Code abweichende Dateinamen produziert, müsste dies abgeglichen werden.

4. XNAT-spezifische Wrapper/Verbindungen

- Die Ableitung der Ressourcen, Children und Matcher ist logisch konsistent.
- Die Output-Handler-Zuordnung zu einer XNAT-Session und die Label-Vergabe passen.

5. Potenzielle Verbesserungen / Hinweise

a) Input Validation

- Stelle sicher, dass die Erzeugung der Eingabedatei zuverlässig mit jedem Allow-Pattern funktioniert (Encoding der Dateinamen, Umlaute etc.).

b) Output-Handling

- Falls mehrere Sessions gleichzeitig berechnet werden, wäre ein unique Outputfilename (im Code) wichtig, um Kollisionen zu vermeiden.

c) Dokumentation

- In der Beschreibung (`description`) der Inputs/Outputs eventuell darauf eingehen, welche Variablen, Spalten oder Formate die Eingangstabellen besitzen müssen.

d) Fehlerbehandlung und Logging (im Docker-Container selbst)

- Empfohlen: Logging-Ausgaben in der Applikation, damit bei Fehlschlägen transparent ist, an welcher Stelle in der Datenpipeline/welcher Datei es gescheitert ist.

e) Reproduzierbarkeit

- Dokumentation zu verwendeter Python-, Moduls- und Docker-Version für langfristige Wartbarkeit im XNAT-System.

Zusammenfassung

****Dein Ansatz ist technisch korrekt und grundsätzlich sofort lauffähig**** (sofern das Python-Skript und die Mounted-Volumes richtig funktionieren). Die Details hängen dann entscheidend vom tatsächlichen Verhalten deines Skripts und der zu prozessierenden XNAT-Ressourcen ab.

Empfehlung:

Vor dem Deployment zunächst einen Testlauf mit echten Beispieldaten und typischen Userwegen wagen (sowohl mit validen als auch invaliden Daten), um die robuste Funktionalität zu bestätigen.

****Falls du bestimmte Fehlermeldungen, Anforderungen an das Input-File oder Output-Format hast, kann ich gerne noch gezielter bei der Feinspezifikation helfen!****

Bousfiha, Tanae:

meine container läuft abre lädt keine Datei hoch

HAWKI (gpt-4.1):

Das Problem, dass der Container ****läuft****, aber ****keine Datei hochgeladen wird****, kann verschiedene Ursachen haben. Hier ist eine strukturierte Vorgehensweise zur Fehlersuche und hilfreiche Hinweise speziell für den Einsatz mit ****XNAT-Container Services****:

1. **Prüfe die XNAT-Ressource und den Wrapper**

- ****Ist am MR-Session-Objekt wirklich ein „CSV“-Ressourcenordner mit einer passenden Datei (csv, tsv, txt) angelegt?****
- ****Stimmt der Ressourcenname exakt mit „CSV“ überein?**** (Achte auf Groß-/Kleinschreibung! Achte auch darauf, dass es nicht „Csv“ oder „csv“ ist.)

2. **Prüfe die gemounteten Daten im Container**

- ****Logge dich ggf. interaktiv in den laufenden Container ein**** (z. B. mit `docker exec

-it <container-id> /bin/bash`) und überprüfe, ob die Datei im Verzeichnis `/app/input` liegt.

- **Beispiel:**

...

ls -l /app/input

cat /app/input/<deine_datei.csv>

...

- **Wenn das Verzeichnis leer ist:**

=> Die Datei wurde vermutlich nicht vom Wrapper übergeben (z. B. Ressourcen-Label falsch, Datei nicht als Resource angehängt, Matcher Fehler).

3. **Matcher Fehler?**

- **Der Matcher verlangt Dateinamen-Endungen `.csv`, `.tsv`, `.txt`.**

Prüfe, ob die Datei genau so heißt. Z. B. funktioniert eine Datei `.CSV` (Großbuchstaben) oder `.csv.bak` NICHT.

- **XNAT führt Matcher serverseitig aus und ist sensibel gegenüber kleinen Unstimmigkeiten!**

4. **Output: Resource-Handling**

- Schau in deiner XNAT-GUI nach, ob unter der MR-Session ein „OSA Result“-Ordner erstellt wurde bzw. ob dort eine CSV hinterlegt ist.

- Kommt ein Output-File im gewünschten Ziel an? Wird vielleicht gar kein Output erzeugt, weil ein Fehler im Python-Skript auftritt (fehlende Input-Datei etc.)?

5. **Container-Logfile prüfen**

- Lasse dir die Container-Logs ausgeben, z. B.

...

docker logs <container-id>

...

- Gibt das Python-Skript Fehlermeldungen im Falle fehlender Datei-Eingaben aus?
- Typische Fehlermeldungen lauten z. B.:

...

FileNotFoundError: [Errno 2] No such file or directory: '/app/input/<dateiname>'

...

6. **XNAT Docker-Service: Benutzungsfehler/GUI?**

- **Beim Ausführen des Wrappers in XNAT:**

Hast du beim Start des Containers für die gewählte MR-Session auch wirklich das richtige Resource File ausgewählt?

- Wird dir die Datei in der Auswahlmaske überhaupt angeboten?
- Wird der Fortschritt/Fehlerstatus korrekt angezeigt oder steht der Service auf „running/pending/failed“?

7. **Warum keine Datei hochgeladen?**

Mögliche Ursachen (zusammengefasst):

- **Ressourcen-LABEL stimmt nicht** (`"CSV"` vs `"Csv"` oder `"csv"`)
- **Dateiname mit falscher Endung**
- **Input-File am falschen XNAT-Objekt** (z. B. nicht bei MR-Session, sondern Scan)
- **Fehler im Python-Skript verhindert Output-File-Erzeugung**
- **Keine Schreibberechtigung auf Output-Mount**
- **Fehlende `provides-files-for-command-mount` oder `provides-value-for-command-input` Zuordnung**

Checkliste für dich

1. **Im XNAT-Webinterface:**

- Existiert eine „CSV“ Resource beim MR-Session-Objekt?
- Ist dort eine passende Datei hinterlegt?

2. ****Im Docker-Container:****

- Kommt die Datei als `/app/input/DEINEDATEI.csv` wirklich an?
- Wird ein Output-File `/app/output/osa_result.csv` erzeugt?

3. ****Im XNAT:****

- Wird das Output-File als neue Resource angezeigt?

4. ****Im Logfile:****

- Existieren Fehlermeldungen? (z. B. „file not found“)

****Wenn du die Ausgabe von `ls /app/input` und die wichtigsten Zeilen aus dem Container-Log bereitstellen kannst, kann ich gezielter weiterhelfen!****

Melde dich gern mit konkreten Logs oder Dateinamen, dann können wir das Problem genau einkreisen.

Bousfiha, Tanae:

ja er findet die Datei

so sind meine STDOUT

Ø=ßâ Start OSA-Analyse

Ø=ÜÁ Inhalt von INPUT_DIR: ['#039;CSV_catalog.xml'#039;,
'correct_patient.csv'#039;]

Ø=ÜÄ Gefundene Eingabedateien: ['#039;correct_patient.csv&

Ø=Ý Verarbeite Datei: correct_patient.csv

Ø=ÜÊ Gelesene Daten:

snoring tired stop_breath ... older_than_50 neck gender

0 0 0 1 ... 1 1 m

[1 rows x 8 columns]

Ø=Ü¾ Speichere Ergebnis nach: /app/output/osa_result.csv

' Datei erfolgreich gespeichert.

' chmod erfolgreich gesetzt.

Ø=ÜÁ Inhalt von OUTPUT_DIR: ['#039;osa_result.csv'#039;]

HAWKI (gpt-4.1):

Deine Logausgabe zeigt ganz eindeutig:

****Die Eingabedatei wird gefunden, verarbeitet und ein Ergebnis (`osa_result.csv`) korrekt im Output-Ordner gespeichert.****

****Das Problem liegt also NICHT am Container oder Skript, sondern sehr wahrscheinlich an der XNAT-Seite** (Resource-Upload/Zuordnung). Hier sind die nächsten, systematischen Prüfschritte:**

1. ****Prüfe die Output-Resource in XNAT****

- Erscheint unter deiner ****Session**** ein Ordner „OSA Result“?
- Wird dort eine Datei angezeigt?
 - Falls ****nein****: Der Output-Handler könnte einen Fehler haben.
 - Falls ****ja, aber leer****: Die Datei wurde zwar als Resource angelegt, aber kein Inhalt wurde hochgeladen.

2. ****Überprüfe Wrapping und Output-Handler****

Dein Output-Handler-Eintrag sieht so aus:

```
```json
```

```
"output-handlers": [
 {
 "name": "osa_output",
 "accepts-command-output": "result_file",
 "as-a-child-of": "session",
 "type": "Resource",
 "label": "OSA Result",
 "format": "csv",
 "tags": []
 }
]
```
```

- ****Path**** der Datei musst exakt stimmen:

! Im Container wird sie als `/app/output/osa_result.csv` ges

! Im Output-Handler ist `"path": "osa_result.csv"` gesetzt.

- Falls du versehentlich z.B. `"path": "/app/output/osa_result.csv"` im Handler verwendest, ****funktioniert das nicht**** – der Pfad muss relativ zum Output-Mount sein.

3. **Checke Folder- und Permissions**

- Hat der Service-User auf dem XNAT-Server Schreibrechte auf den Output-Resource-Ordner?
- Wird im XNAT-Log eine Fehlermeldung zum Resource-Upload ausgegeben? Prüfe das Server-Admin-Panel oder frage den XNAT-Admin.

4. **Automatischer Refresh/Anzeige**

- XNAT zeigt neu angelegte Resources manchmal ****erst nach Refresh**** oder beim nächsten Öffnen an. Versuche, die Session-Ansicht neu zu laden.
- Prüfe, ob die Datei zumindest über die REST-API oder als Download verfügbar ist.

5. **Versteckte Fehler beim Run**

- Manchmal bleiben Output-Handler stumm, wenn ein Problem in der XNAT-Engine vorliegt (fehlende /tmp-Rechte, Storage-Voll, ...).
- ****XNAT Admin-Logs**** können helfen: ``/data/xnat/home/logs/catalina.out`` oder ähnlich, je nach Setup.

Was kannst du jetzt tun?

Schrittweise Diagnose

1. ****Logge dich im XNAT-Webinterface als Admin ein****

- Kontrolliere die Resource-Ordner/Dateien an der betroffenen Session.
- Suche nach dem Label „OSA Result“.

2. ****Prüfe den Output-Handler****

- Stelle sicher, dass der Dateiname exakt ``osa_result.csv`` heißt und nicht z.B. mit Leerzeichen oder Sonderzeichen versehen ist.
- Kontrolliere, ob der Output-Handler wirklich auf ``result_file`` (so benannt im Container-Output) referenziert.

3. ****Teste gegebenenfalls mit einer Beispiel-Resource****

- Lege eine Testdatei manuell als Resource an – wird sie angezeigt?

4. ****XNAT-Server-Log prüfen**** (Admin-Zugang nötig)

- Gibt es Meldungen wie „File upload failed“?

Typische Ursachen für dieses Verhalten

- Output-Handler referenziert falschen Namen.
- Resource wird wegen Storage-Quotas oder Schreibrechten nicht ordentlich geschrieben.
- XNAT „vergisst“ das Resource-Refresh (selten, aber möglich).
- Die Datei ist im Container, wird aber beim Hochladen nicht übergeben (falsche Path-Zuordnung).

Zusammenfassung

> **Dein Container funktioniert, aber der Output wird nicht in XNAT angezeigt.**

>

> Prüfe den Output-Handler im Wrapper, den genauen Dateinamen/-pfad und die Result-Resource in XNAT!

> Ggf. Logs mit Adminrechten einsehen.

Soll ich dir konkret bei der Prüfung des Output-Handler-Abschnitts oder der XNAT-Resource-Struktur weiterhelfen?

Gerne kannst du auch einen Screenshot deiner XNAT-Session-Ressourcen oder die Ausgabe des Container-Service-Runs posten.

Bousfiha, Tanae:

wie kann ich eine App creiieren

HAWKI (gpt-4.1):

Gerne erläutere ich dir, wie du eine „App“ erstellen kannst! Da der Begriff „App“ verschiedene Bedeutungen hat (z./B. Mobile-App, Web-App, wissenschaftliches Skript oder auch einen XNAT-/Docker-Service wie oben beschrieben), gehe ich auf die wichtigsten akademisch relevanten Varianten ein. Falls du eine spezifische Art von App meinst, konkretisiere das gern!

**** 1. Web-App (z./B. mit Python und Flask oder Django)****

- ****Ziel:**** Interaktive Anwendung im Browser (z./B. Datenanalyse, Benutzerverwaltung)

- ****Schritte:****

1. ****Entwicklungssprache wählen:**** Häufig Python (Flask, Django), JavaScript (Node.js), oder R (Shiny)

2. ****Grundstruktur aufsetzen:**** Beispiel Flask:

```
```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
 return "Hallo, HAWK!"

if __name__ == '__main__':
 app.run(debug=True)
```
```

3. ****Funktionen hinzufügen:**** z./B. Formulare, Datenbankanschlüsse

4. ****Testen und dokumentieren:**** Funktioniert alles? Ist die Bedienung plausibel?

5. ****Deployment:**** Auf Server oder als Docker-Container (z./B. mit Docker) bereitstellen

**** 2. Desktop-App (z./B. mit Python und Tkinter)****

- ****Ziel:**** Kleine Werkzeuge/Fenster-Anwendungen unter Windows/Mac/Linux

- ****Beispiel:****

```
```python
import tkinter as tk

root = tk.Tk()
root.title("Meine erste App")
label = tk.Label(root, text="Hallo, HAWK!")
label.pack()
```

```
root.mainloop()
```

```
```
```

```
---
```

****3. Mobile-App (Android, iOS)****

- **** Mit Low-Code/No-Code (z./B. AppGyver, Thunkable, MIT App Inventor)**
- Ohne viel Programmierkenntnis, grafische Erstellung per Drag-and-drop
- ****Mit Programmiersprache:****
- Kotlin/Java (Android Studio) für Android, Swift (Xcode) für iOS, Flutter oder React Native für beide

```
---
```

****4. Wissenschaftliche „App“: Workflow-Automatisierung oder Docker-basierte Service****

- ****Eine App im Sinne universitärer Datenservices (wie dein Container oben):****
- Fasse Skripte, Libraries, Umgebungen in einem Docker-Image zusammen
 - **Definiere ein Interface (z./B. per REST-API oder XNAT-Workflow)**
- Dokumentiere klar die benötigte Inputstruktur und Outputformate
- ****Werkzeuge:**** Docker, Python, Bash, ggf. XNAT-Konnektoren

```
---
```

****5. Best Practices für das App-Projekt****

- ****Versionierung:**** Verwende z./B. Git für Codeverwaltung
- ****Dokumentation:**** Lege in einer README die Funktionsweise und die Installationsschritte dar
- ****Testen:**** Schreibe einfache Testfälle, prüfe Randfälle
- ****Nachhaltigkeit:**** Konzipiere so, dass andere die App nachnutzen und weiterentwickeln können (Open Science!)

```
---
```

****Beispiel-Szenario „Datenanalyse-App als Webanwendung für die HAWK“****

- Du möchtest einen CSV-Upload und eine anschließende statistische Auswertung per Klick anbieten
- Vorschlag: Web-App (Flask + Bootstrap), Upload-Formular,

- Analyse in Python (mit pandas, scipy)
- Ausgabe als Download

****Tipp:****

Am Anfang hilft es, sehr klein zu starten (Minimalbeispiel), dann Schritt für Schritt Funktionen hinzuzufügen und zwischendurch häufig zu testen.

****Hast du schon eine bestimmte Programmiersprache, Zielplattform oder Anwendungsfall im Kopf? Dann kann ich dir gezielt eine passende Mini-Anleitung oder ein Startprojekt geben!****