ReSort

„Farben erkennen- Richtig trennen!
ReSort, Das System für jeden“

Übersicht

1. Zusammenfassung ReSort
2. Aktueller Stand
3. Technischer Ablauf und Implementierung
4. Praxisbeispiel Bildaufnahme
5. Praxisbeispiel Bild hochladen
6. Herausforderungen und Lösungsansätze
7. Bezug PoC (Audit 3)
8. Bezug priorisierte PoC
9. Zukunftsperspektiven und Weiterentwicklung
10. Fazit

Zusammenfassung ReSort

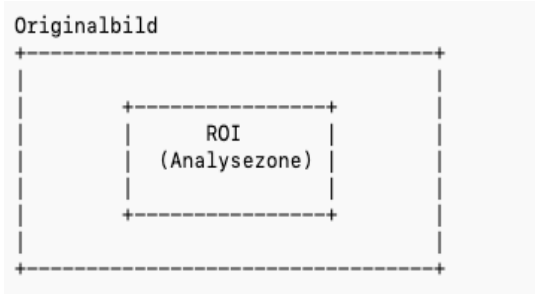
Ziel des Projekts:

Entwicklung einer mobilen Anwendung zur transparenten und barrierearmen Glasfarberkennung auf Basis regelbasierter Bildverarbeitung

Kernidee:

Analyse eines Glasbildes → Klassifikation → Konfidenzbewertung → konkrete Entsorgungsempfehlung

Zusammenfassung

1. 

2.
```python
# 1. Grünes Glas
# Hue 28-90° (Grünbereich), mittlere bis hohe Sättigung
if 28 <= hue <= 90 and saturation > 20:  # Erweitert für knapp-grüne Farben
    confidence = min(1.0, (saturation / 100) * 0.6 + 0.4)  # Höhere Base-Konfidenz
    logger.info(f"→ Klassifiziert als: Glasflasche_Gruen (Konfidenz: {confidence:.2%})")
    return 'Glasflasche_Gruen', confidence
```

3.

```
"recycling_info": {
    "material": "Grünes Glas",
    "recycling_category": "Grüner Glascontainer",
    "instructions": "Deckel entfernen und in den grünen Glascontainer werfen. Pfandflaschen zum Automaten bringen.",
"message": "Flasche erkannt: Glasflasche_Gruen",
"processing_time_ms": 77.15,
"timestamp": "2026-01-15T15:49:19.009552"
```

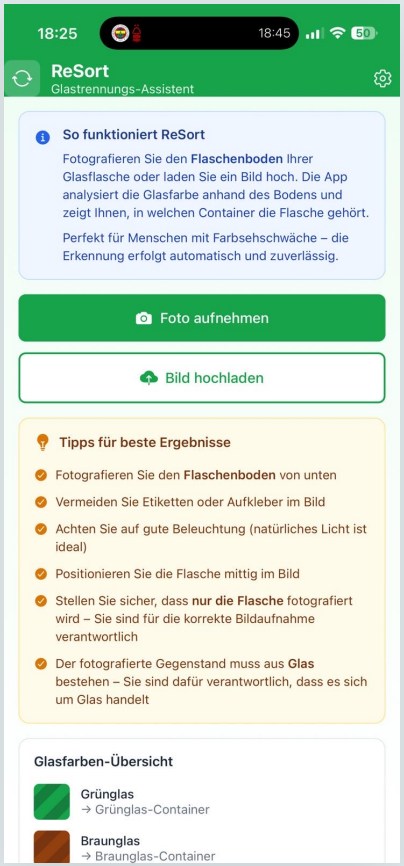
Aktueller Stand

- Entwicklung eines funktionsfähigen Prototyps als hybride Applikation
- Reduziertes und barrierearmes UI mit klarer und simpler Struktur
- Ergebnisdarstellung durch symbolische Icons zur Unterstützung der Barrierefreiheit
- Anzeige eines Konfidenzwertes zur Transparenz der Klassifikation
- Anzeige einer kontextbezogenen Entsorgungsempfehlung
- Anpassung der Region of Interest an den Flaschenboden (runde ROI-Struktur) sowohl für die Funktion der Bildaufnahme, als auch des Bildhochladens

Technischer Ablauf und Implementierung

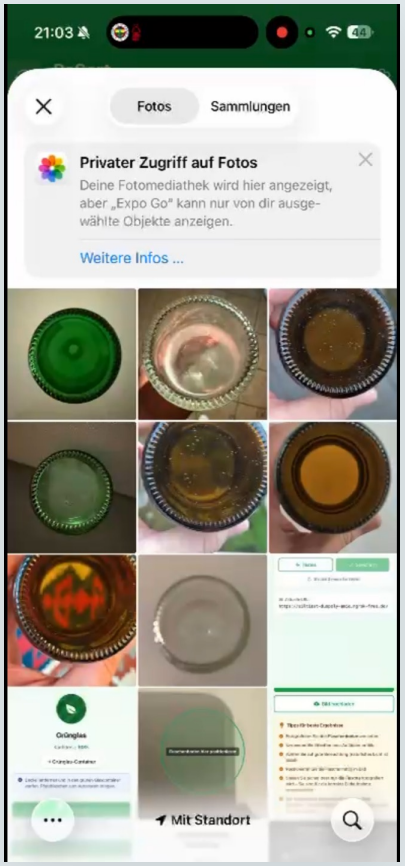
1. Backend-Implementierung mit Python (FastAPI) inklusive Analyse-Logik und REST-Endpunkten
2. Containerisierung mittels Docker
3. Frontend-Backend-Kommunikation über REST-API (Client-Server-Architektur)
4. UI-Konzeption mit Figma, Umsetzung mit React Native & Expo (JavaScript)
4. Mit Expo .apk Datei erstellt, um die App auf einem Android Smartphone ausführen zu können
5. Auf Smartphone übertragen

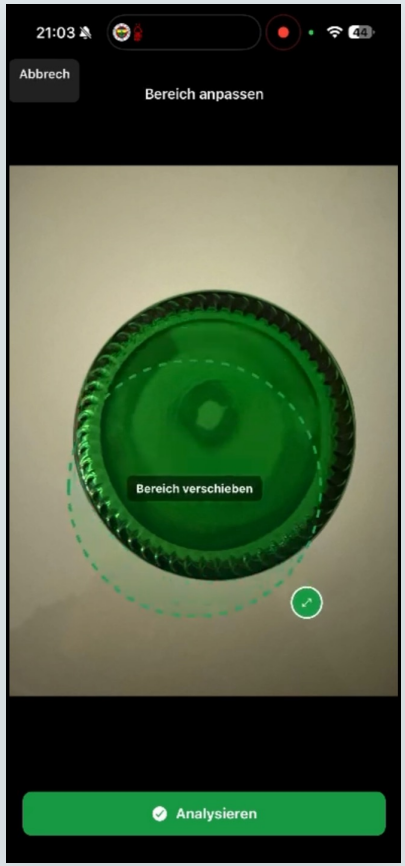
Praxisbeispiel Bildaufnahme

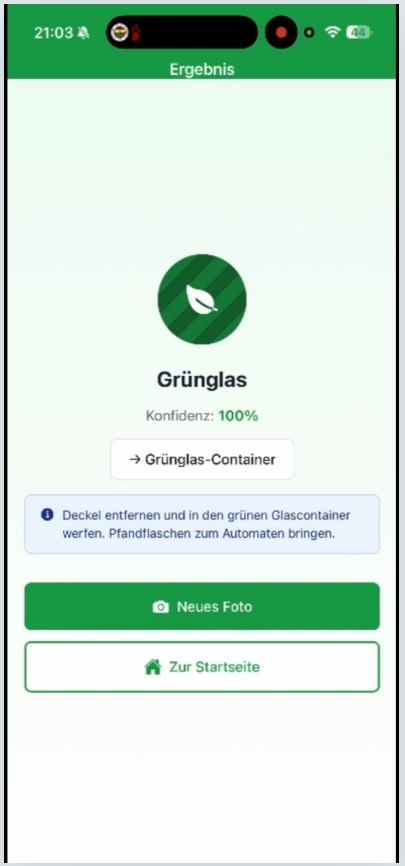


Praxisbeispiel Bild hochladen

Praxisbeispiel Bild hochladen







Herausforderungen und Lösungsansätze

- Anfangs fehlerhafte Glas-Klassifikation durch ungenaue Farbgrenzwerte
- Iterative Anpassung der HSV Schwellenbereiche zur Verbesserung der Erkennungsrate
- Kommunikationsproblem zwischen Frontend und Backend aufgrund fehlender HTTPS-Unterstützung auf mobilen Endgeräten
- Lösung durch Bereitstellung einer sicheren HTTPS-Schnittstelle mittels **ngrok**
- Leistungsprobleme durch Emulator-Nutzung → Verlagerung der Tests auf reale Endgeräte

Bezug PoC (Audit 3)

PoC	Status	Einschätzung
PoC 1 – Kamera, Bildaufnahme, ROI	**Abgedeckt**	vollständig umgesetzt
PoC 2 – Regelbasierte Farberkennung im HSV-Farbraum	**Abgedeckt**	vollständig umgesetzt
PoC 3 – Ergebnisausgabe, Transparenz und Barrierefreiheit	**Teils abgedeckt**	Im Backend vorhanden
PoC 4 – Regionale Entsorgung (Gummersbach)	**Teils Abgedeckt**	technisch umgesetzt
PoC 5 – Confidencewert & Ergebnisverwerfung	**Neu**	Aus Code ergeben
PoC 6 – API-Stabilität & Umgang mit Fehlern	**Neu**	Aus Code ergeben

Bezug priorisierte PoC

PoC	Status	Einschätzung
PoC 1 – Kamera, Bildaufnahme, ROI	**Abgedeckt**	vollständig umgesetzt
PoC 2 – Regelbasierte Farberkennung im HSV-Farbraum	**Abgedeckt**	vollständig umgesetzt
PoC 3 – Ergebnisausgabe, Transparenz und Barrierefreiheit	**Abgedeckt**	vollständig umgesetzt

Zukunftsperspektiven und Weiterentwicklung

- Weiterentwicklung der Farberkennung durch robustere Analyseverfahren (z. B. Differenzmodell mit erweitertem ROI)
- Implementierung einer audiobasierten Ausgabe zur Unterstützung stark sehbeeinträchtigter Nutzer
- Integration von Ortungsdiensten zur Anzeige nahegelegener Glascontainer
- Anbindung an lokale Entsorgungsrichtlinien mit automatisierter Aktualisierung
- Evtl. Objekterkennung um sicherzustellen, dass es sich wirklich um Glas handelt

Fazit

- Einfache und intuitive Bedienung für verschiedene Alters- und Nutzergruppen
- Transparente und nachvollziehbare Funktionsweise
- Gute Grundlage für eine mögliche Integration in bestehende Systeme
- Potenzial als Lern- und Sensibilisierungsinstrument für korrekte Glastrennung
- Weiterentwicklungsbedarf vorhanden, jedoch vielversprechende Zukunftsperspektive
- Bewusste funktionale Begrenzung im Rahmen des Moduls zur Sicherstellung der Umsetzbarkeit

Vielen Dank für Ihre Aufmerksamkeit

ReSort Farbanalyse

- **Hue (28–90°)** → Grünbereich im HSV-Farbraum
- **Saturation > 20** heißt die Farbe ist ausreichend kräftig.
- leicht grünliche Töne werdenebenfalls berücksichtigt
- **Basiswert (0.4)** → Regel grundsätzlich
- **Zusatzwert** → je höher die Sättigung, desto höhere Sicherheit
- **Begrenzung auf max. 1.0** (also = 100 %)
- Rückgabe des Flaschentypen **„Glasflasche_Gruen“**
- Confidence zeigt, **wie sicher** die Entscheidung ist
- Ausgabe im Log als Prozentwert (nur Darstellung)

```
    # 1. Grünes Glas
    # Hue 28-90° (Grünbereich), mittlere bis hohe Sättigung
    if 28 <= hue <= 90 and saturation > 20:  # Erweitert für knapp-grüne Farben
        confidence = min(1.0, (saturation / 100) * 0.6 + 0.4)  # Höhere Base-Konfidenz
        logger.info(f"→ Klassifiziert als: Glasflasche_Gruen (Konfidenz: {confidence:.2%})")
        return 'Glasflasche_Gruen', confidence
```

Confidence Wert

- **Confidence = Basiswert + gewichtete Farbmerkmale**

```
Confidence = (saturation / 100) * 0.6 + 0.4
```

- Confidence wird **regelbasiert pro Glasart** berechnet
- Jede Regel hat:
- einen **Basiswert**
- eine **gewichtete Bewertung von Farbmerkmalen**
- Wertebereich: **0.0 – 1.0**
- Zu niedrige Confidence- Ergebnis wird verworfen bzw. Fehlermeldung

```
Nutzer:in
    ↓
Bildaufnahme
(Kamera oder Bild-Upload)
    ↓
API-Endpunkt
/api/v1/analyze
    ↓
Eingabevalidierung
- Dateityp (image/*)
- Dateigröße (≤ 10 MB)
- Dateiformat (jpg, png, webp)
    ↓
Region of Interest (ROI)
- Standard-ROI (zentral)
- optional: manueller ROI (Prozentwerte)
    ↓
Farbanalyse
- BGR → HSV
- Mittelwerte im ROI
    ↓
Regelbasierte Klassifikation
- Weißglas
- Braunglas
- Grünglas
    ↓
Confidence-Berechnung
(0.0 – 1.0)
    ↓
Qualitätsfilter
(confidence ≥ min_confidence?)
    ↓
Recycling-Logik
- Regionale Regeln (Gummersbach)
    ↓
Ergebnisausgabe
- Glasfarbe
- Confidence
- Entsorgungsempfehlung
```

Bezug PoC

- **PoC 1 – Kamera, Bildaufnahme & ROI**
- Bild-Upload über API
- Standard-ROI + optionaler Custom-ROI
- Performante und deterministische ROI-Berechnung

- **PoC 2 – Regelbasierte Farbanalyse (HSV)**
- Umwandlung BGR → HSV
- Mittelwertbildung im ROI
- Feste Schwellenwerte für Weiß-, Braun- und Grünglas

Bezug PoC

- **PoC 3 – Ergebnisausgabe & Barrierefreiheit**
- Klare textuelle Ergebnisse
- Confidence-Wert zur Transparenz
- Unsichere Ergebnisse werden verworfen

- **PoC 4 – Regionale Entsorgung (Gummersbach)**
- Kontextbezogene Entsorgungsempfehlungen