Mediwaste Solutions

Group 41

Louis Gysens
Lowiek Van de Walle
Flor Cornu
Ulrike Uittenhove
Siebe Lansen

Concept (1)

- Hazardous medical waste
- Hospitals, pharmacies, laboratories
- WIVA barrels
- Reduce excessive hazardous waste

- Conversion into regular waste
- Volume reduction 80%

Concept (2)

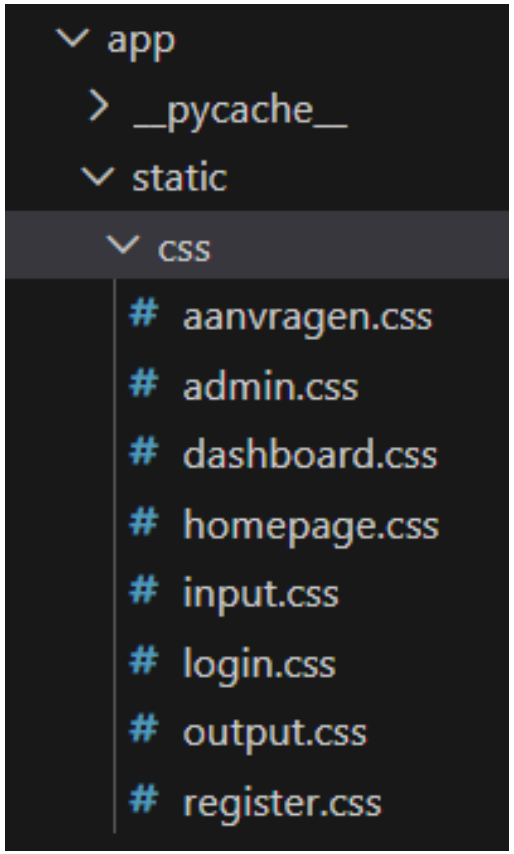
- Simulation tool
- Minimum of inputs

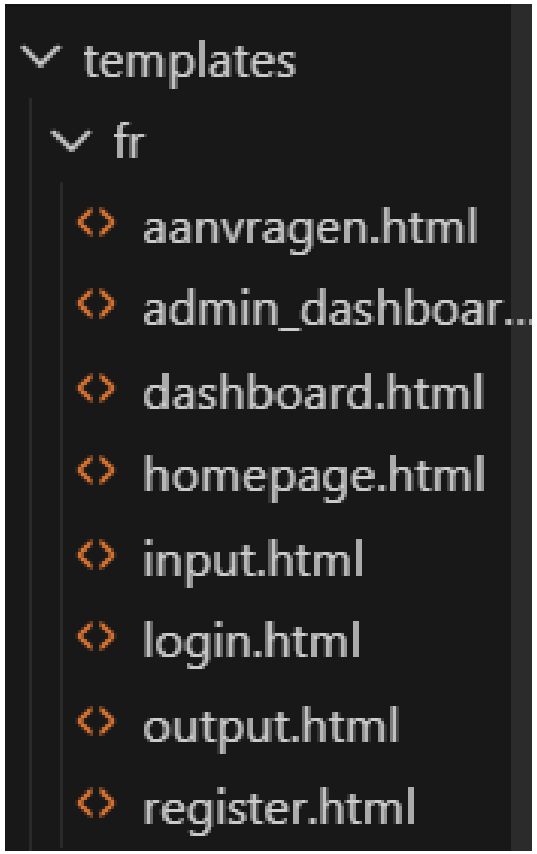
- Type of machine
- Payback period

- Niche concept

HTML + CSS

- Layout



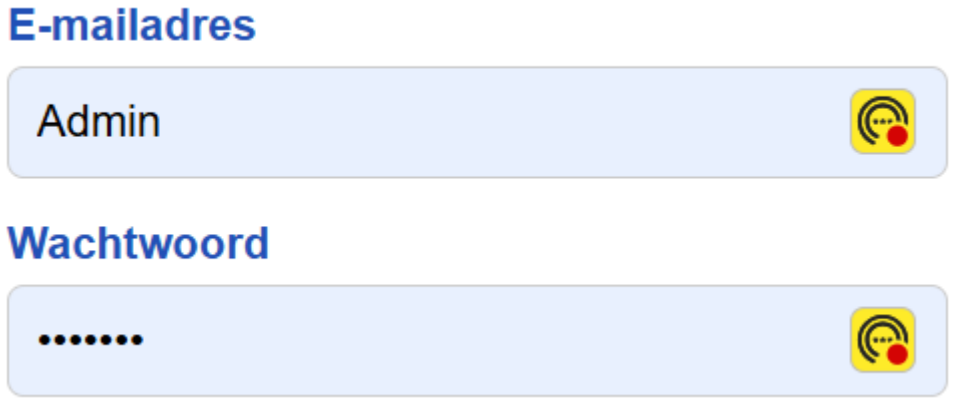


Register

```
<select id="position" onchange="toggleJobInput()" required>
  <option value="Andere">Andere</option>
  ...
</select>

<script>
    function toggleJobInput() {
        const select = document.getElementById('position');
        const otherContainer = document.getElementById('other-job-container');
        const otherInput = document.getElementById('job_other');

        if (select.value === 'Andere') {
            otherContainer.style.display = 'block';
            otherInput.required = true;
        } else {
            otherContainer.style.display = 'none';
            otherInput.required = false;
            otherInput.value = '';
        }
    }
</script>
```

Login

```
<form>
  <label>E-mailadres</label>
  <input type="email" name="email" required>

  <label>Wachtwoord</label>
  <input type="password" name="password" required>

  <button type="submit">Aanmelden</button>
</form>
```



Output

```
<div class="results">
  <div class="result-box">
    <h3>Aanbevolen machinetype</h3>
    <p>{{ machine.size_code if machine else "–" }}</p>
  </div>
```

Resultaat besparingsanalyse

Aanbevolen machinetype

T700

Terugverdientijd van de investering

52 maanden

Requests

```
<tbody>
  {% for req in requests %}
    <tr>
      <td>{{ req.created_at.strftime('%d/%m/%Y') }}</td>
      <td>{{ req.waste_profile.hmw_total_weight }} kg</td>
      <td>€ {{ req.waste_profile.cost_collection_processing }}</td>
    </tr>
  {% endfor %}
</tbody>
```

Overzicht van uw aanvragen

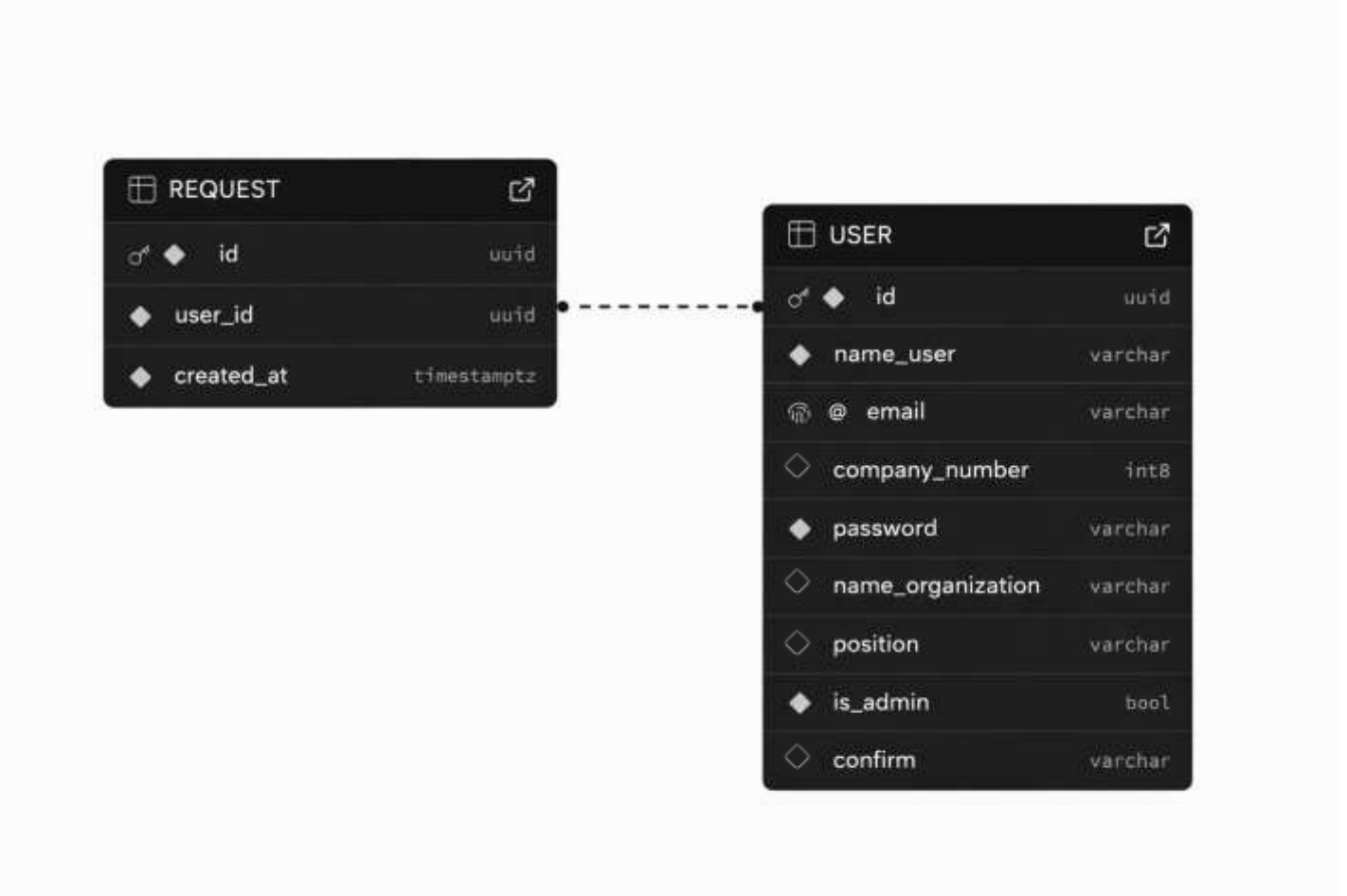
Datum	Totaal gewicht RMA	Totaal Volume	Kost ophaling & verbranding (excl. WIVA)	Type Machine	Terugverdientijd
10/12/2025	16000.00 kg	300000 liter	€ 5000.00	**T150**	-
16/12/2025	160000.00 kg	300000 liter	€ 5000.00	**T700**	**142 maanden**

← Terug naar dashboard

Database - Overview

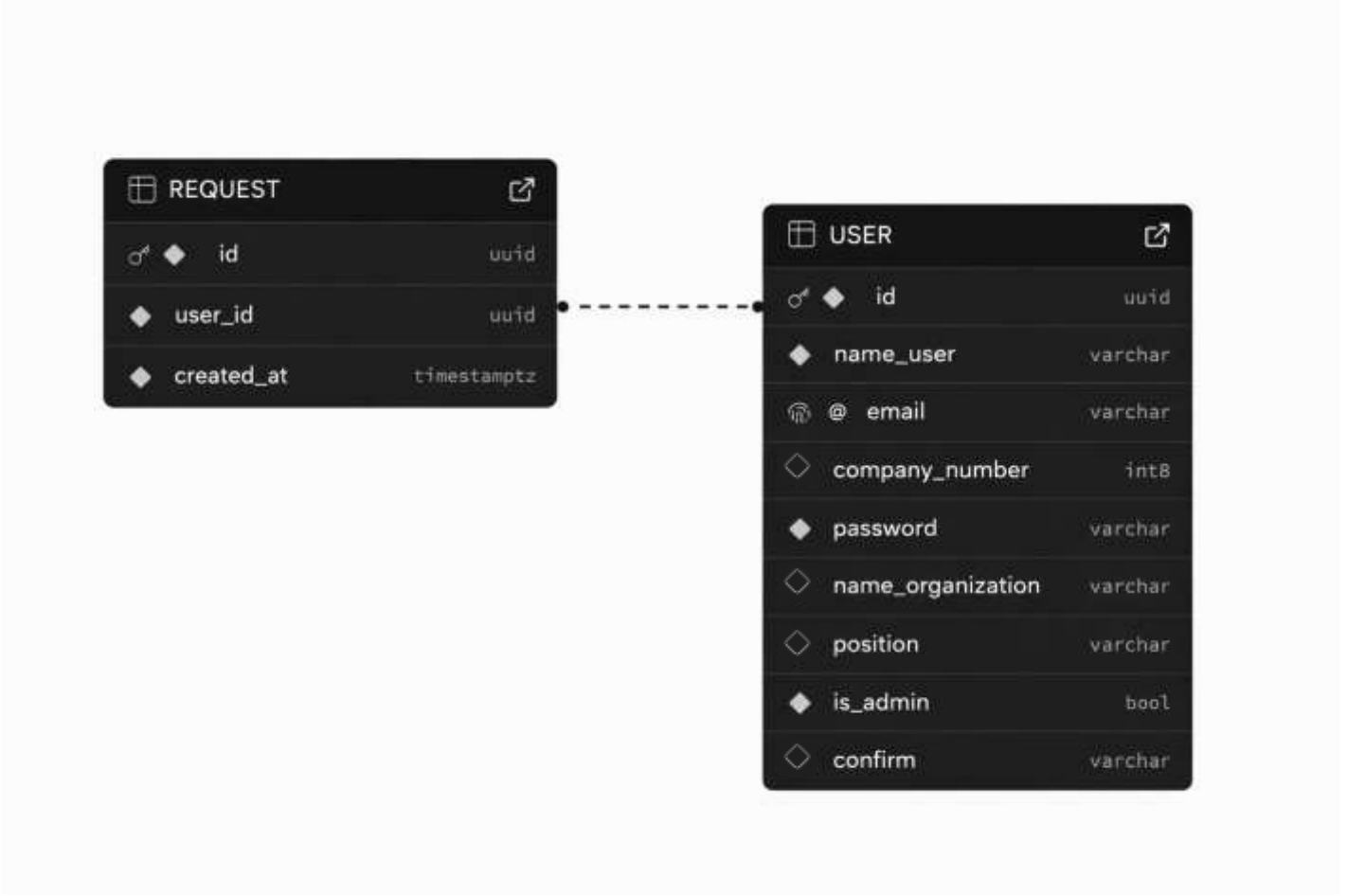
Database – User

- Unique user identity
- Authentication & authorization
- ∞ Requests



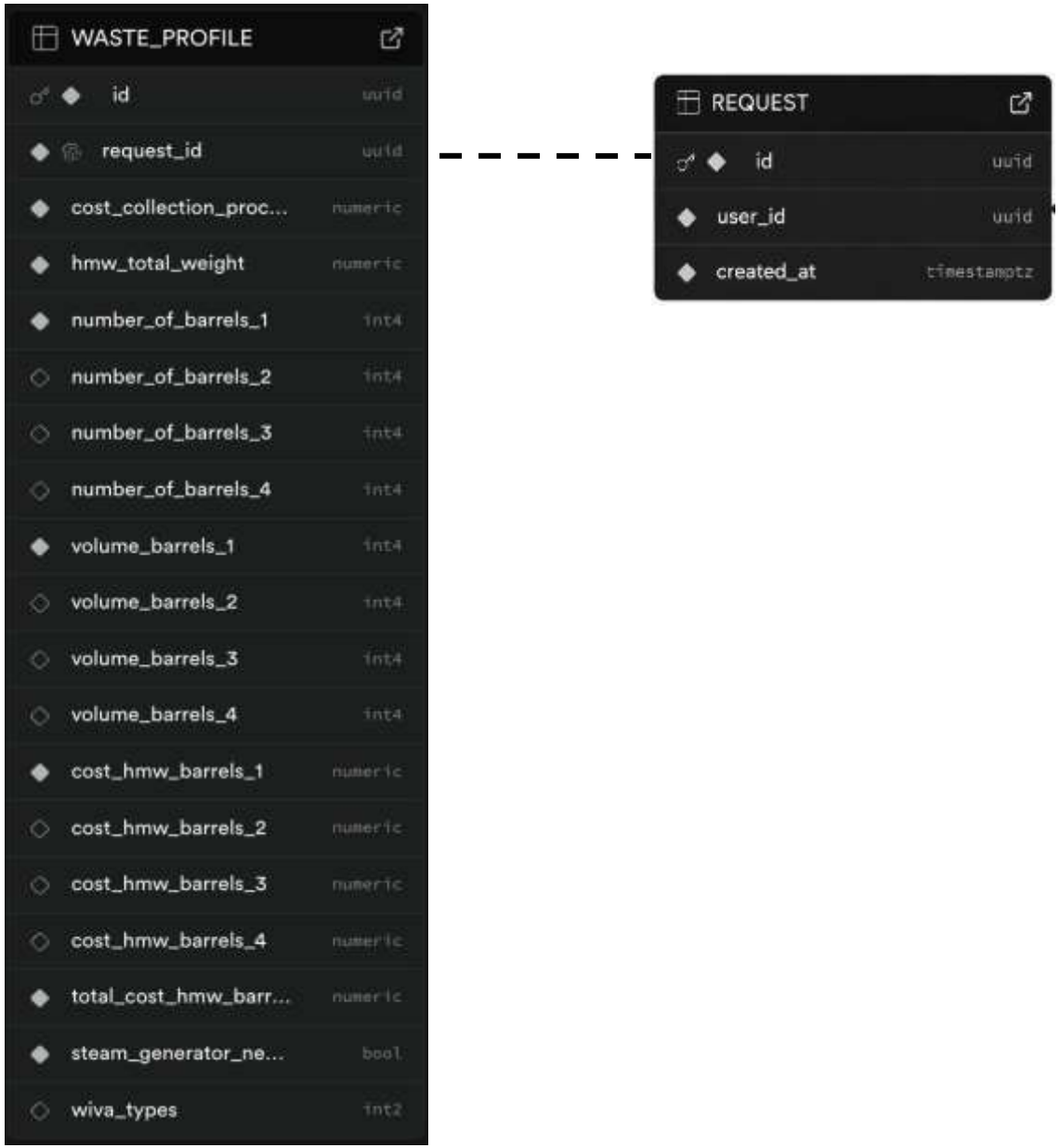
Database – Request

- Central entity
- 1 Request = 1 Simulation
- History
- Traceability



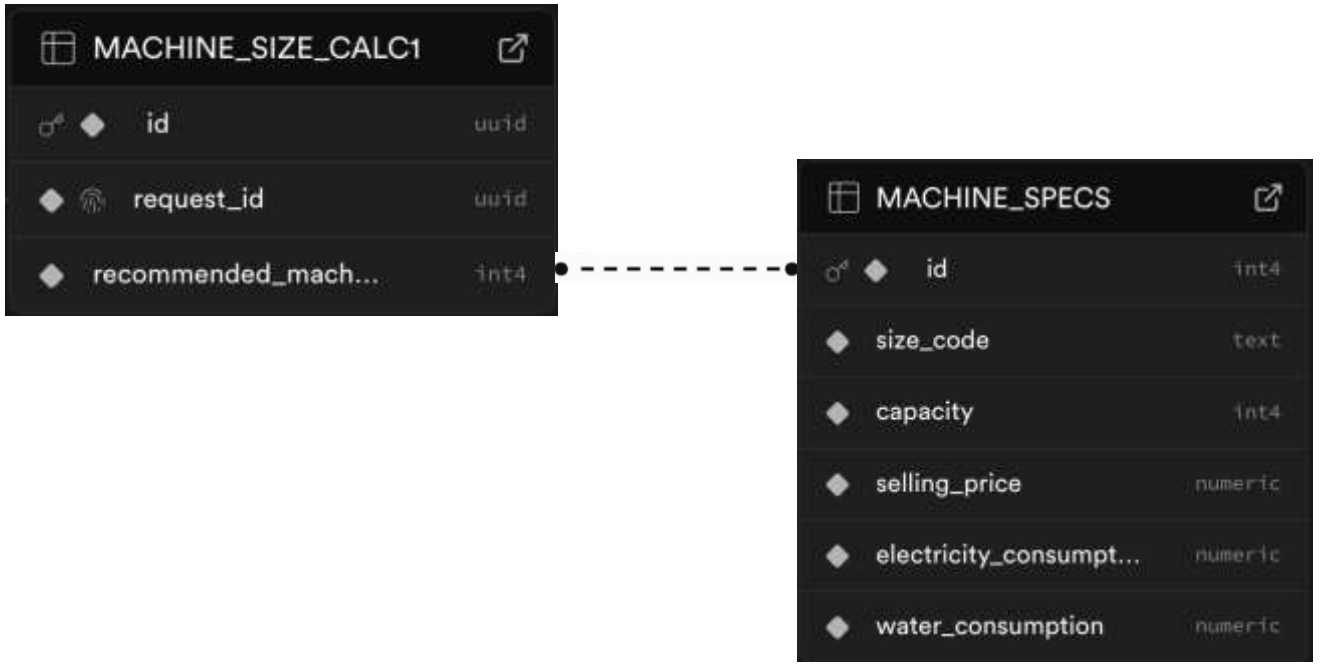
Database – Waste Profile

- Simulation input
- Optional <-> Required
- Calculated field
  - Total_cost_hmw_barrels



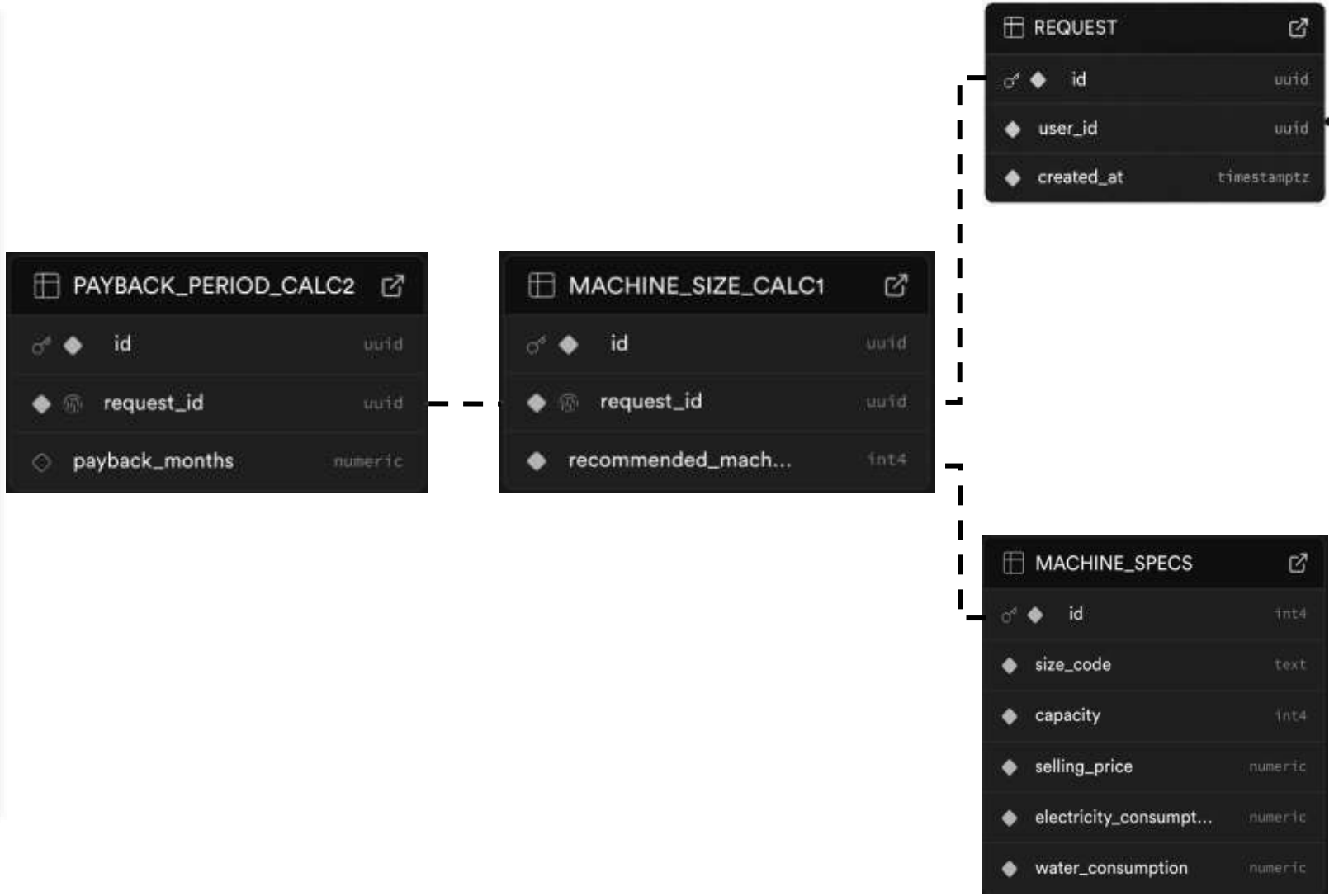
Database - Machine Specs

- Reference table
- Independent of simulations
- Maintenance & scalability



Database - Calculations

- Separated calculations
- Traceability
- History per simulation

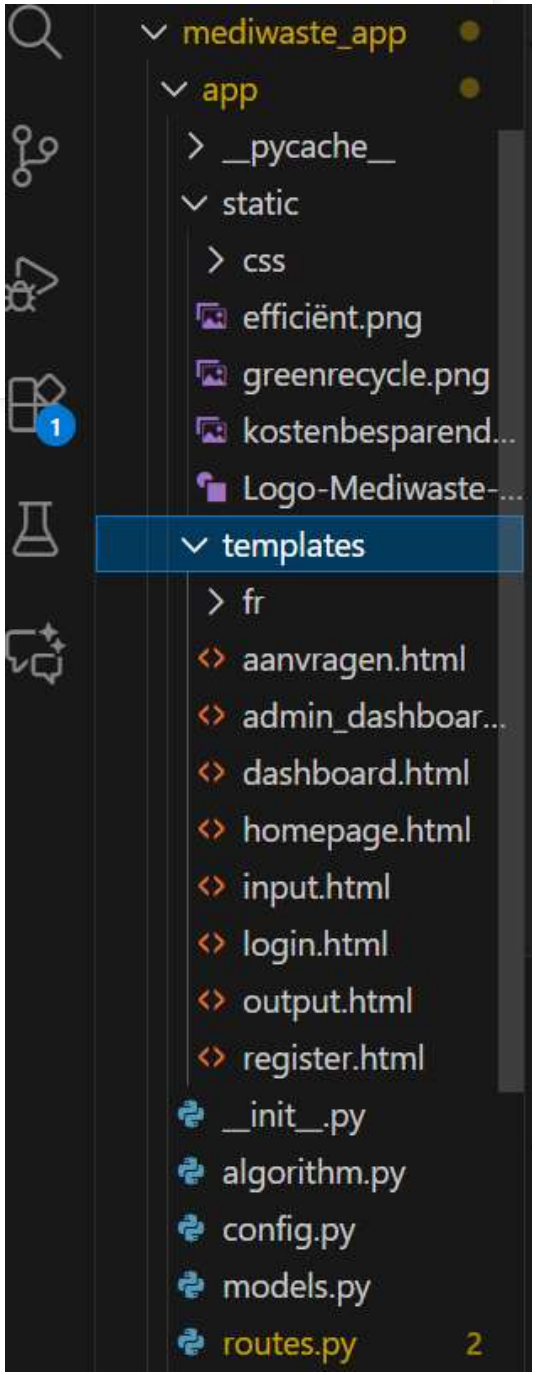


Database - Overview

Structure flask

Frontend vs backend

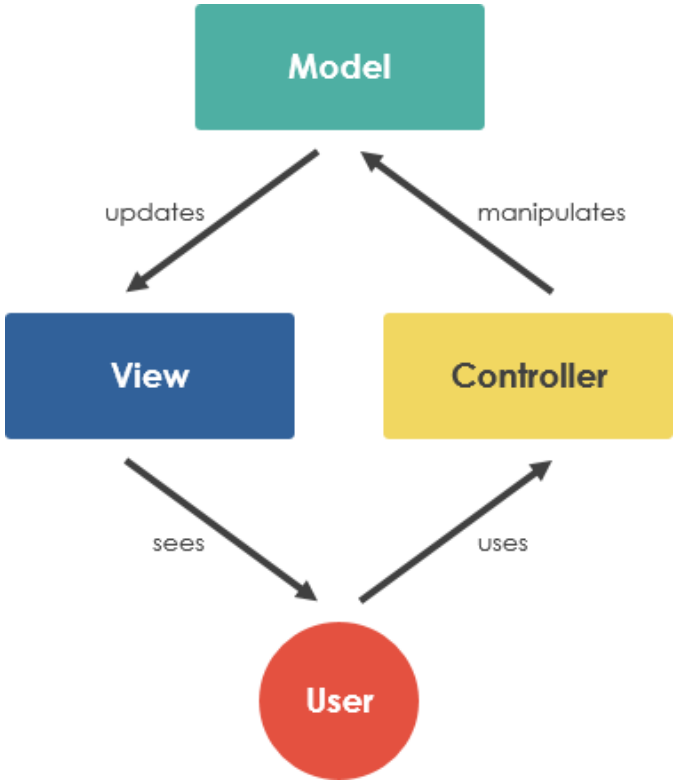
```python
# ------------------------
# 4. Dashboard
# ------------------------
@main.route("/dashboard")
def dashboard():
    user_id = session.get("user_id")
    if not user_id:
        # Als niet ingelogd, stuur naar login (Belangrijke beveiligingscheck)
        return redirect(url_for("main.login"))
    return render_lang("dashboard.html")
```



Structure flask

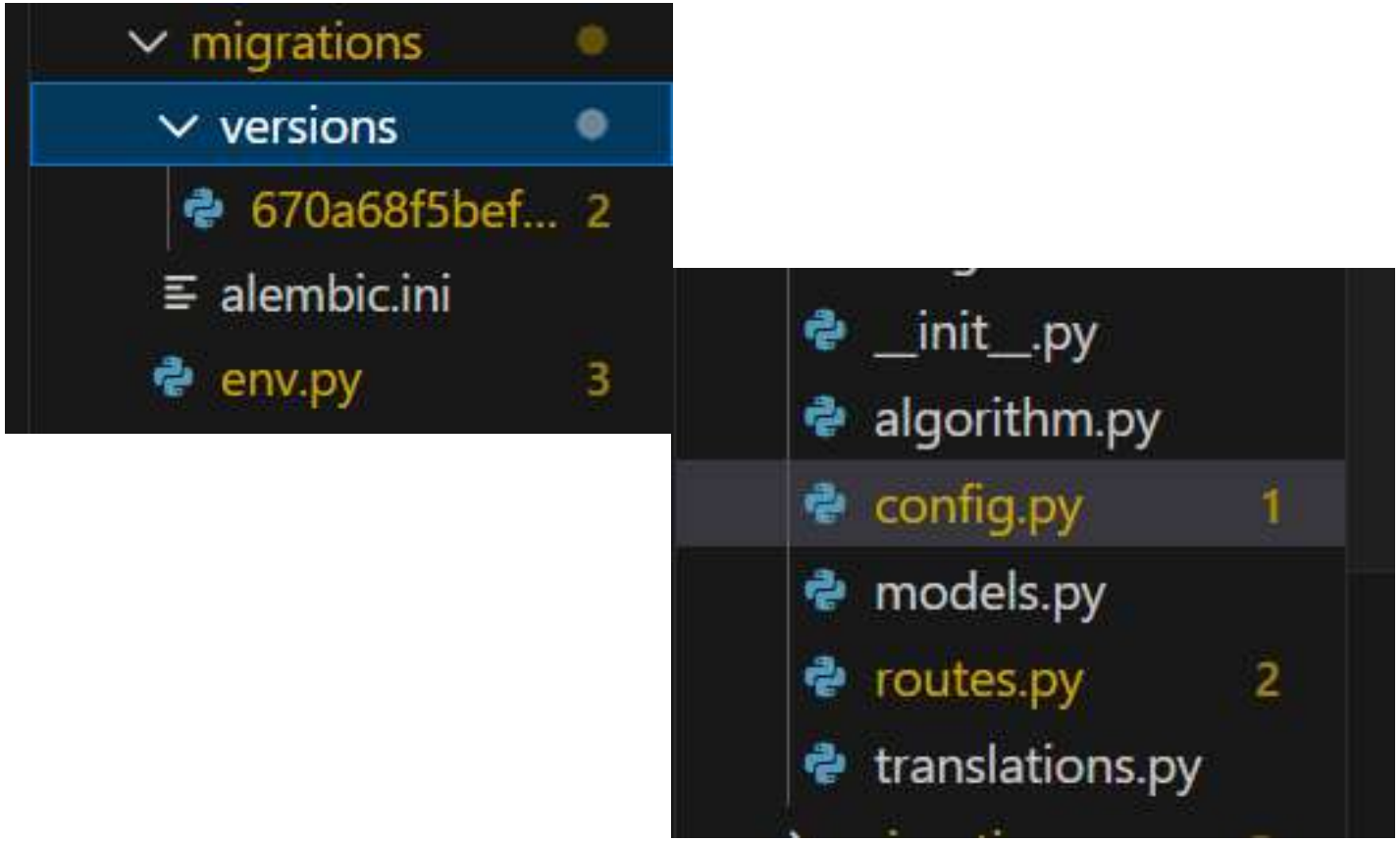
- MVC-structuur

```python
class User(db.Model):
    __tablename__ = quoted_name("USER", True)

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid.uuid4)
    name_user = Column(String, nullable=False)
    email = Column(String, unique=True, nullable=False)
    company_number = Column(Integer, nullable=False)
    password = Column(String, nullable=False)
    name_organization = Column(String, nullable=False)
    position = Column(String, nullable=False)
    is_admin = Column(Boolean, default=False)
    confirm = Column(String)

    requests = relationship("Request", back_populates="user")
```



Structure flask

- Maintainability
- Eigen modules



Algorithm (1)

Constants and base calculations

mediwaste_app > app > algorithm.py > ...

```python
import math
from .models import db, WasteProfile, MachineSizeCalc1, MachineSpecs, PaybackPeriodCalc2
from decimal import Decimal


# CONSTANTEN – AANNAMES
# ---------------------
# MACHINE OPERATION CONSTANTS
WORKDAYS_PER_YEAR = 300
MAX_DAILY_CYCLES = 8
EFFECTIVE_CAPACITY_FACTOR = Decimal("0.94")  # machine never filled to 100%
# WASTE REDUCTION / PROCESSING FACTORS
VOLUME_REDUCTION_FACTOR = Decimal("0.20")
COLLECTION_REDUCTION_FACTOR = Decimal("0.40")
# VARIABLE OPERATING COST PRICES
ELECTRICITY_PRICE_PER_KWH = Decimal("0.18") # AZMM
WATER_PRICE_PER_L = Decimal("0.0044") # AZMM
COST_PE_ZAKKEN_MACHINE = Decimal("0.42") # AZMM, zak voor 60L
# MACHINE-DEPENDENT FIXED COSTS
STEAM_GENERATOR_COSTS = {
    "T100": Decimal("10164"),
    "T150": Decimal("10164"),
    "T300": Decimal("27588"),
    "T700": Decimal("35574"),
}
MAINTENANCE_COSTS = {
    "T100": Decimal("14500"),
    "T150": Decimal("19000"),
    "T300": Decimal("30000"),
    "T700": Decimal("42000"),
}
# FINANCIAL MODEL SETTINGS
MAX_PAYBACK_YEARS = 15
YEARLY_INTEREST = Decimal("0.04")  # 4% rente

```

Algorithm (2)

Volume calculation

```python
#gedeelde hulpfunctie voor volume
def compute_annual_volume_l(waste) -> Decimal:
    """
    Berekent het totale jaarlijkse afvalvolume in liters op basis van tonnage.
    """
    if waste.hmw_total_weight is None:
        return Decimal("0")

    # Gewichtsinput (kg) kan als string binnenkomen → cast naar Decimal
    kg = Decimal(str(waste.hmw_total_weight))

    # 1 kg → 10/1.2 liter
    liters_per_kg = Decimal("10") / Decimal("1.2")

    # Jaarvolume in liters
    annual_volume_l = kg * liters_per_kg

    return annual_volume_l
```

Recommend Machine

```python
def recommend_machine(request_id):

    waste = WasteProfile.query.filter_by(request_id=request_id).first()
    if waste is None:
        raise ValueError("NO_WASTE_PROFILE") # Kan gebruiker niet oplossen -> niet getoond op site

    # Jaarvolume berekenen met gedeelde hulpfunctie
    annual_volume_l = compute_annual_volume_l(waste)
    if annual_volume_l == 0:
        raise ValueError("ZERO_VOLUME") # Kan gebruiker oplossen -> wordt getoond op de site

    machines = MachineSpecs.query.order_by(MachineSpecs.capacity.asc()).all()
    if not machines:
        raise ValueError("NO_MACHINES_CONFIGURED") # Kan gebruiker niet oplossen -> niet getoond op site

    max_yearly_cycles = MAX_DAILY_CYCLES * WORKDAYS_PER_YEAR
    required_volume_per_cycle = Decimal(annual_volume_l) / Decimal(max_yearly_cycles)

    # Bepaal maximale effectieve machinecapaciteit
    max_machine_capacity = max(
        Decimal(machine.capacity) * EFFECTIVE_CAPACITY_FACTOR
        for machine in machines)

    # Als het vereiste volume groter is dan wat eender welke machine aankan → fout
    if required_volume_per_cycle > max_machine_capacity:
        raise ValueError("TONNAGE_TOO_HIGH") # Kan gebruiker oplossen -> wordt getoond op de site

    for machine in machines:
        effective_capacity = Decimal(machine.capacity) * EFFECTIVE_CAPACITY_FACTOR

        if effective_capacity >= Decimal(required_volume_per_cycle):
            return machine

    raise ValueError("NO_MACHINE_FOUND") # Kan gebruiker niet oplossen -> niet getoond op site
```

Algorithm (3)

Running and saving optimal machine size

```python
def run_user_algorithm(request_id=None):

    # Request_id bepalen
    if request_id is None:
        # routes.py geeft request_id niet door; we pakken de meest recente WasteProfile
        waste = WasteProfile.query.order_by(WasteProfile.id.desc()).first()
        if waste is None:
            raise ValueError("NO_LATEST_WASTE_PROFILE")
        request_id = waste.request_id

    # Machine aanbevelen
    machine = recommend_machine(request_id)
    recommended_machine_id = machine.id

    # Machine opslaan in MACHINE_SIZE_CALC1
    existing = MachineSizeCalc1.query.filter_by(request_id=request_id).first()

    if existing is None:
        new_calc = MachineSizeCalc1(
            request_id=request_id,
            recommended_machine_id=recommended_machine_id
        )
        db.session.add(new_calc)
    else:
        existing.recommended_machine_id = recommended_machine_id

    db.session.commit()

    return {
        "machine_id": recommended_machine_id,
        "payback_period": None}
```

Algorithm (4)

Discounted payback period

-Not in use-

Simple payback period calculation

```python
def payback_period_months(investment, annual_savings):
    #Discounted payback in maanden.
    if annual_savings <= 0 or investment <= 0:
        return None

    monthly_savings = annual_savings / Decimal("12.0")
    monthly_rate = Decimal(str(YEARLY_INTEREST)) / Decimal("12")

    cumulative_saved = Decimal("0")
    max_months = MAX_PAYBACK_YEARS * 12

    for month in range(1, max_months + 1):
        discounted_cf = monthly_savings / ((Decimal("1") + monthly_rate) ** month)
        cumulative_saved += discounted_cf
        if cumulative_saved >= investment:
            return month

    return None

def simple_payback_months(investment, annual_savings):
    if annual_savings <= 0 or investment <= 0:
        return None
    return math.ceil((investment / annual_savings) * 12)
```

Algorithm (5)

Costs before using the machine

```python
def run_payback_for_request(request_id):
    # 1. WASTE_PROFILE ophalen
    # ------------------------
    waste = WasteProfile.query.filter_by(request_id=request_id).first()
    if waste is None:
        raise ValueError(f"WASTE_PROFILE not found for request_id={request_id}")

    barrel_cost_annual = Decimal("0") #zonder dit werd in de loop hieronder bij barrel_c_annu
    # ... een float opgeteld bij een decimal wat niet werkte
    cost_streams = [
        (waste.number_of_barrels_1, waste.cost_hmw_barrels_1),
        (waste.number_of_barrels_2, waste.cost_hmw_barrels_2),
        (waste.number_of_barrels_3, waste.cost_hmw_barrels_3),
        (waste.number_of_barrels_4, waste.cost_hmw_barrels_4),
        ]

    for n_barrels, total_cost in cost_streams:
        if n_barrels is not None and total_cost is not None:
            barrel_cost_annual += Decimal(total_cost)

    # Verwerking/verbranding en ophaling, excl. WIVA-vaten
    processing_cost_annual = Decimal(str(waste.cost_collection_processing or "0"))

    # Totale huidige kost zonder machine
    baseline_annual_cost = barrel_cost_annual + processing_cost_annual
```

Get machine size,
calculate annual cycles for costs

```python
def run_payback_for_request(request_id):
    # 2. Machinekeuze ophalen
    # -----------------------
    msize = MachineSizeCalc1.query.filter_by(request_id=request_id).first()
    if msize is None:
        raise ValueError(f"MACHINE_SIZE_CALC not found for request_id={request_id}")

    machine = MachineSpecs.query.filter_by(id=msize.recommended_machine_id).first()
    if machine is None:
        raise ValueError(
            f"MACHINE_SPECS not found for id={msize.recommended_machine_id}"
        )


    # 3. Cycli per jaar + gebruikskosten
    # ----------------------------------
    effective_capacity = Decimal(machine.capacity) * EFFECTIVE_CAPACITY_FACTOR
    if machine.capacity <= 0:
        raise ValueError("Machine capacity must be > 0")

    annual_volume_l = compute_annual_volume_l(waste)

    cycles_per_year = math.ceil(Decimal(annual_volume_l) / effective_capacity)
```

Algorithm (6)

Costs after purchasing the machine

```python
# 4. Kosten mét machine
# ------------------------------
electricity_cost_annual = Decimal(cycles_per_year) * Decimal(machine.electricity_consumption) * ELECTRICITY_PRICE_PER_KWH
water_cost_annual = Decimal(cycles_per_year) * Decimal(machine.water_consumption)  * WATER_PRICE_PER_L

processing_cost_with_machine = processing_cost_annual * COLLECTION_REDUCTION_FACTOR

maintenance_cost = MAINTENANCE_COSTS.get(machine.size_code)
if maintenance_cost is None:
    raise ValueError(f"No maintenance cost configured for machine {machine.size_code}")

# geen WIVA-vaten meer nodig met machine? wel zakken, prijs?
barrel_cost_with_machine = 0.0
reduced_volume = Decimal(annual_volume_l) * VOLUME_REDUCTION_FACTOR
bags = (reduced_volume / Decimal("60")).to_integral_value(rounding="ROUND_CEILING")
cost_bags_for_machine = COST_PE_ZAKKEN_MACHINE * bags

# Zorg dat alles Decimals worden
selling_price = Decimal(machine.selling_price)
monthly_rate = Decimal(str(YEARLY_INTEREST)) / Decimal("12")

# (1 + r) ** -120  → Decimal power werkt NIET met floats of negatieve exponenten
# oplossing: gebruik Decimal ** int  (negatieve exponent kan wél)
discount_factor = (Decimal("1") + monthly_rate) ** Decimal("-120")

# Annuity interest component
total_interest = (Decimal("120") * ((selling_price * monthly_rate) / (Decimal("1") - discount_factor))) - selling_price

annual_interest_cost = total_interest / Decimal("10")

#Alles naar hetzelfde formaat brengen om de komende som uit te kunnen voeren
processing_cost_with_machine = Decimal(processing_cost_with_machine)
maintenance_cost = Decimal(maintenance_cost)
barrel_cost_with_machine = Decimal(barrel_cost_with_machine)
cost_bags_for_machine = Decimal(cost_bags_for_machine)
annual_interest_cost = Decimal(annual_interest_cost)
electricity_cost_annual = Decimal(electricity_cost_annual)
water_cost_annual = Decimal(water_cost_annual)

annual_cost_with_machine = (
    processing_cost_with_machine
    + maintenance_cost
    + barrel_cost_with_machine
    + cost_bags_for_machine
    + annual_interest_cost
    + electricity_cost_annual
    + water_cost_annual
)

annual_savings = Decimal(baseline_annual_cost) - Decimal(annual_cost_with_machine)
```

Algorithm (7)

Potential additional costs and storing payback period

```python
# 5. Totale investering
# ------------------------------
investment = Decimal(machine.selling_price)

if not waste.steam_generator_needed:
    extra_steam_cost = STEAM_GENERATOR_COSTS.get(machine.size_code)

    if extra_steam_cost is None:
        # hard fail als we de code niet kennen
        raise ValueError(f"No steam generator cost configured for machine {machine.size_c

    investment -= Decimal(extra_steam_cost)


# 6. Terugverdientijd (in maanden)
# ------------------------------
months = payback_period_months(investment, annual_savings)
simple_months = simple_payback_months(investment, annual_savings)

if months is None:
    payback_value_to_store = None
else:
    try:
        payback_value_to_store = float(months)
    except:
        # fallback: forceer conversie via str → float
        payback_value_to_store = float(str(months))
    # werkt voor int, decimal, float en none
```

Write result to database and return result

```python
# 7. Resultaat wegschrijven naar PAYBACK_PERIOD_CALC2
# ------------------------------
existing = PaybackPeriodCalc2.query.filter_by(request_id=request_id).first()
if existing is None:
    row = PaybackPeriodCalc2(
        request_id=request_id,
        payback_months=payback_value_to_store,
    )
    db.session.add(row)
else:
    existing.payback_months = payback_value_to_store

db.session.commit()


# 8. Resultaat teruggeven (handig voor template / debug)
# ------------------------------
return {
    "request_id": str(request_id),
    "baseline_annual_cost": baseline_annual_cost,
    "annual_cost_with_machine": annual_cost_with_machine,
    "annual_savings": annual_savings,
    "investment": investment,
    "payback_months": months,
    "simple_payback_months": simple_months,
    "machine_id": machine.size_code,
    "cycles_per_year": cycles_per_year,
}
```

Future

- Extend to multiple languages
- Add discounted cashflow

Demo


NL
FR
MEDIWASTE
SOLUTIONS
Registreren
Inloggen
Bespaar op de kosten voor de verwerking van RMA
Ontdek hoeveel uw organisatie kan besparen met een alternatieve manier van medische afvalverwerking.
Vraag hier uw analyse aan