CIV-CARE CIV-CALL

DE KAMER VAN SOCIALE WAARDEN

Menselijke waardigheid als fundament

CIV-CAMP CIV-RAMP

Praktische Uitwerking 5

IT-Architectuur – Technische specificaties RC-tracking systeem

Versie 0.9 – 15 February 2026
Regeneratief Economisch Model v0.9

Opgesteld door:

Alexander Groenheide
Voorzitter
Stichting De kamer van Sociale Waarden
Naar de waardevolle Samenleving
info@dekvsw.nl
www.dekvsw.nl
06 53 44 50 54
Laurastraat 87, 6471 JJ Eygelshoven
KVK: 97098817 | RSIN: 867909274
W: DeKvSW.nl | ubuntukids.nl |
civ-care.nl | civ-call.com | civ-ramp.nl | civ-camp.nl

Samen bouwen aan een waardegedreven samenleving
Liefde – Samenwerking – Zorg voor elkaar

Inhoudsopgave

Inhoud

Inhoudsopgave 1
1. Doel en Reikwijdte 3
2. Architectuurprincipes 3
3. Systeemarchitectuur – Overzicht 3
4. RC-Ledger Specificaties 4
5. Validatie- en Auditmechanismen 4
6. Gegevensbescherming en Compliance 5
7. Beveiligingsmaatregelen 5
8. Governance en Toezicht 5
9. Schaalbaarheid en Performance 6
10. Implementatiefasen 6
11. Risicoanalyse 6
12. Conclusie 7
BIJLAGE A – UML-Architectuurschema's (PlantUML) 8
A1. Componentdiagram 8
A2. Sequencediagram (claim tot toekenning) 9
A3. Klassendiagram (kern-entiteiten) 9
BIJLAGE B – Datamodel (ERD) en Tabellen 11
B1. ERD – Tabellen en relaties (tekstueel) 11
B2. Constraints en datakwaliteit 12
BIJLAGE C – Formele API-specificaties (OpenAPI 3.0) 13
C1. OpenAPI 3.0 YAML (kern-endpoints) 13
C2. API-regels (security, audit, versioning) 16

1. Doel en Reikwijdte

Dit document beschrijft de volledige technische architectuur van het Regeneratieve Credits (RC) tracking systeem.
Het doel is een juridisch compliant, schaalbaar, transparant en fraudebestendig systeem te ontwerpen voor:
- Registratie van RC-transacties
- Validatie van zorg- en commons-bijdragen
- Monitoring en auditing
- Rapportage en beleidsanalyse

Het systeem moet functioneren binnen de kaders van de Grondwet, AVG, Gemeentewet en Algemene wet bestuursrecht.

2. Architectuurprincipes

Het RC-systeem wordt ontworpen volgens de volgende leidende principes:

1. Privacy by Design (AVG art. 25)
2. Security by Default
3. Transparantie en controleerbaarheid
4. Dataminimalisatie
5. Functionele scheiding tussen identiteitsgegevens en transactiedata
6. Modulaire schaalbaarheid (microservices-architectuur)
7. Interoperabiliteit met gemeentelijke systemen (API-first design)

3. Systeemarchitectuur – Overzicht

Het systeem bestaat uit vijf lagen:

A. Presentatielaag (Front-end)
- Burgerportaal (web + mobiel)
- Dashboard voor wijkraden
- Auditinterface

B. Applicatielaag
- RC-registratiemodule
- Validatie-engine
- Workflowmanagement

C. Validatielaag
- Peer-attestatie mechanisme

- Random audit selectie
- Risicoscore-algoritme

D. Datalaag
- Versleutelde RC-ledger
- Gescheiden identiteitsdatabase
- Logging- en auditdatabase

E. Integratielaag
- REST API
- DigiD-koppeling (optioneel)
- Gemeentelijke gegevensuitwisseling

4. RC-Ledger Specificaties

Het RC-ledger functioneert als een onveranderbare transactieregistratie.

Technische kenmerken:
- Versleutelde opslag (AES-256)
- Hash-verificatie per transactie
- Tijdstempel (UTC)
- Unieke transactie-ID

Transacties bevatten:
- Pseudonieme gebruikers-ID
- Activiteitstype
- Aantal toegekende RC's
- Validatiestatus
- Auditstatus

Het systeem mag geen publiek verhandelbare blockchain implementeren zonder expliciete wettelijke basis.

5. Validatie- en Auditmechanismen

Validatie verloopt via een hybride model:

1. Peer-attestatie (minimaal 2 bevestigingen)
2. Geautomatiseerde risicodetectie
3. Steekproefaudit (5–20% afhankelijk van risicoscore)
4. Escalatie naar onafhankelijke auditcommissie bij vermoeden van fraude

Auditlogboeken zijn niet wijzigbaar en worden 7 jaar bewaard.

6. Gegevensbescherming en Compliance

Het systeem voldoet aan:

- AVG (EU 2016/679)
- Uitvoeringswet AVG
- Art. 10 Grondwet (privacy)
- Wet open overheid (Woo)

Verplichtingen:
- DPIA voorafgaand aan implementatie
- Verwerkingsregister
- Functionaris Gegevensbescherming betrokken
- Recht op inzage en correctie
- Dataminimalisatieprincipe

Identiteitsgegevens worden strikt gescheiden van RC-transactiedata.

7. Beveiligingsmaatregelen

Beveiligingslagen omvatten:

- Multi-factor authenticatie
- Rolgebaseerde toegang (RBAC)
- End-to-end encryptie
- Penetratietesten (jaarlijks)
- Incident response protocol
- 24/7 monitoring bij nationale uitrol

Back-ups worden versleuteld en geografisch gespreid opgeslagen.

8. Governance en Toezicht

Het systeem valt onder publiekrechtelijk toezicht.

Toezichtsstructuur:

- Lokale systeembeheerder (gemeente)
- Regionale technische auditcommissie
- Onafhankelijke externe IT-auditor

- Jaarlijkse rapportage aan gemeenteraad

Broncode wordt bij voorkeur open-source gepubliceerd om publieke controle te waarborgen.

9. Schaalbaarheid en Performance

Het systeem moet minimaal ondersteunen:

- Pilot: 5.000 gebruikers
- Regionaal: 500.000 gebruikers
- Landelijk: 10+ miljoen gebruikers

Technische eisen:
- Uptime > 99,5%
- Max. responstijd < 2 seconden
- Horizontale schaalbaarheid via containerization (Docker/Kubernetes)

10. Implementatiefasen

Fase 1 – Prototype (6 maanden)
Fase 2 – Pilotgemeenten
Fase 3 – Onafhankelijke evaluatie
Fase 4 – Regionale uitrol
Fase 5 – Wettelijke borging

Elke fase vereist formele evaluatie en parlementaire rapportage.

11. Risicoanalyse

Belangrijkste risico's:

- Gaming en collusie
- Privacy-inbreuk
- Technische storingen
- Vertrouwensverlies bij incidenten

Mitigatie:
- Transparante rapportage
- Heldere sanctieprotocollen
- Onafhankelijke toezichtstructuur

12. Conclusie

Het RC-tracking systeem is geen commercieel platform, maar een publiek-institutioneel instrument.
Het ontwerp moet juridisch robuust, technisch veilig en maatschappelijk controleerbaar zijn.
De legitimiteit van het gehele regeneratieve model hangt mede af van de betrouwbaarheid van dit systeem.

BIJLAGE A – UML-Architectuurschema's (PlantUML)

A1. Componentdiagram

```
@startuml
skinparam componentStyle rectangle
title RC-Tracking Systeem – Component Diagram

package "Presentatielaag" {
  [Burgerportaal (Web/Mobiel)] as UI1
  [Wijkraad Dashboard] as UI2
  [Audit Console] as UI3
}

package "Applicatie/Services" {
  [Auth Service] as AUTH
  [RC Transaction Service] as TX
  [Activity Claim Service] as CLAIM
  [Validation Service] as VAL
  [Audit Service] as AUD
  [Reporting Service] as REP
  [Notification Service] as NOTIF
}

package "Data & Integratie" {
  database "Identity Store
(Persoonsgegevens)" as IDDB
  database "RC Ledger
(Transacties, Hash-chain)" as LEDGER
  database "Audit Log
(WORM opslag)" as AUDLOG
  [Integration API Gateway] as APIGW
  [Municipal Systems Adapter] as MUNI
  [Optional: DigiD/eID Broker] as EID
}

UI1 --> AUTH
UI2 --> AUTH
UI3 --> AUTH

UI1 --> CLAIM
UI2 --> REP
UI3 --> AUD

CLAIM --> VAL
VAL --> TX
TX --> LEDGER
AUD --> AUDLOG
REP --> LEDGER
REP --> AUDLOG
AUTH --> IDDB

APIGW --> TX
APIGW --> REP
APIGW --> MUNI
EID --> AUTH

note right of AUDLOG
 WORM = Write Once Read Many
 (immutability + forensic trace)
end note
```

```
@enduml
```

A2. Sequencediagram (claim tot toekenning)

```
@startuml
title RC-Claim – Peer-attestatie – (Random) Audit – Toekenning

actor Burger as U
participant "Burgerportaal" as UI
participant "Auth Service" as AUTH
participant "Activity Claim Service" as CLAIM
participant "Validation Service" as VAL
participant "RC Transaction Service" as TX
database "RC Ledger" as LEDGER
participant "Audit Service" as AUD
database "Audit Log (WORM)" as AUDLOG
participant "Notificatie" as NOTIF

U -> UI: Dien claim in (activiteit, bewijs)
UI -> AUTH: Authenticate/Authorize
AUTH --> UI: OK (token)

UI -> CLAIM: CreateClaim(payload)
CLAIM -> VAL: StartValidation(claim)

VAL -> U: Vraag attestaties (min. 2)
U -> VAL: Attestatie ontvangen (peer1, peer2)

VAL -> AUD: RiskScore(claim) + RandomSelection()
AUD -> AUDLOG: Log audit decision
AUDLOG <-- AUD: OK

alt Audit geselecteerd
  AUD -> U: Auditverzoek (aanvullend bewijs)
  U -> AUD: Lever bewijs / interview
  AUD -> AUDLOG: Log audit outcome
end

VAL -> TX: ApproveClaim + GenerateRCTransaction()
TX -> LEDGER: Append transaction (hash, timestamp)
LEDGER --> TX: txId
TX -> NOTIF: Notify(approval, txId)
NOTIF -> U: Bevestiging + RC saldo update
@enduml
```

A3. Klassendiagram (kern-entiteiten)

```
@startuml
title RC-Tracking – Domeinmodel (klassendiagram)

class User {
  +userId: UUID (pseudoniem)
  +role: Role
  +status: UserStatus
}

class IdentityRecord {
  +identityId: UUID
  +digiDRef: String?
  +name: String
  +dob: Date
  +address: String
}

class ActivityClaim {
```

```
  +claimId: UUID
  +userId: UUID
  +activityType: ActivityType
  +description: String
  +submittedAt: DateTime
  +status: ClaimStatus
  +evidenceRefs: List
}

class Attestation {
  +attestationId: UUID
  +claimId: UUID
  +attestorUserId: UUID
  +attestedAt: DateTime
  +confidence: Integer
}

class ValidationResult {
  +claimId: UUID
  +minAttestationsMet: Boolean
  +riskScore: Float
  +decision: Decision
  +decidedAt: DateTime
}

class RCLedgerTransaction {
  +txId: UUID
  +userId: UUID
  +claimId: UUID?
  +amountRC: Decimal
  +txType: TxType
  +timestamp: DateTime
  +prevHash: String
  +hash: String
  +expiryDate: Date
}

class AuditCase {
  +auditId: UUID
  +claimId: UUID
  +trigger: AuditTrigger
  +openedAt: DateTime
  +status: AuditStatus
  +outcome: AuditOutcome?
}

User "1" -- "0..1" IdentityRecord : linked (separated storage)
User "1" -- "0..*" ActivityClaim
ActivityClaim "1" -- "0..*" Attestation
ActivityClaim "1" -- "0..1" ValidationResult
ActivityClaim "1" -- "0..1" AuditCase
User "1" -- "0..*" RCLedgerTransaction

note bottom of IdentityRecord
IdentityRecord staat in gescheiden database
met strengere toegangscontrole (RBAC)
end note
@enduml
```

BIJLAGE B – Datamodel (ERD) en Tabellen

B1. ERD – Tabellen en relaties (tekstueel)

```
ERD (conceptueel, kern-tabellen)

[identity_records]  (PII - gescheiden opslag)
- identity_id (PK)
- digid_ref (nullable)
- name
- date_of_birth
- address
- created_at, updated_at

[users]  (pseudoniem - applicatielaag)
- user_id (PK)
- identity_id (FK, nullable; alleen via auth service)
- role
- status
- created_at, updated_at

[activity_claims]
- claim_id (PK)
- user_id (FK -> users.user_id)
- activity_type
- description
- submitted_at
- status
- evidence_pointer (json/uri)
- created_at, updated_at

[attestations]
- attestation_id (PK)
- claim_id (FK -> activity_claims.claim_id)
- attestor_user_id (FK -> users.user_id)
- attested_at
- confidence
- signature (optional)
- created_at

[validation_results]
- claim_id (PK, FK -> activity_claims.claim_id)
- min_attestations_met (bool)
- risk_score (float)
- decision
- decided_at
- decided_by (system/auditor)
- created_at

[rc_ledger_transactions]  (WORM/append-only)
- tx_id (PK)
- user_id (FK -> users.user_id)
- claim_id (FK -> activity_claims.claim_id, nullable)
- tx_type (EARN, SPEND, EXPIRE, ADJUST)
- amount_rc
- timestamp
- expiry_date
- prev_hash
- hash
- audit_status
- created_at

[audit_cases]
```

- audit_id (PK)
- claim_id (FK -> activity_claims.claim_id)
- trigger (RANDOM, RISK, COMPLAINT, PATTERN)
- opened_at
- status
- outcome (nullable)
- closed_at (nullable)
- created_at, updated_at

[audit_logs] (WORM)
- log_id (PK)
- correlation_id (claim_id/audit_id/tx_id)
- event_type
- event_payload (json, encrypted fields where needed)
- created_at

B2. Constraints en datakwaliteit

B2. Integriteitsregels en constraints (selectie)
1) attestations: UNIQUE(claim_id, attestor_user_id) – dubbele attestatie voorkomen
2) activity_claims: status transitions alleen via workflow (SUBMITTED -> ATTESTING -> VALIDATING -> APPROVED/REJECTED)
3) rc_ledger_transactions: append-only; updates verboden; correcties via ADJUST transacties
4) expiry: nightly job genereert EXPIRE transacties voor verlopen RC's (traceerbaar)
5) audit trigger: bij risk_score >= threshold of pattern match -> audit_cases record verplicht

BIJLAGE C – Formele API-specificaties (OpenAPI 3.0)

C1. OpenAPI 3.0 YAML (kern-endpoints)

```
openapi: 3.0.3
info:
  title: RC Tracking API
  version: "1.0"
  description: >
    Formele API-specificatie voor RC-claims, attestaties, validatie, transacties en
rapportages.
servers:
  - url: https://api.rc-gemeente.nl/v1
security:
  - bearerAuth: []

components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
  schemas:
    Error:
      type: object
      properties:
        code: { type: string }
        message: { type: string }
        traceId: { type: string }
    ClaimCreateRequest:
      type: object
      required: [activityType, description]
      properties:
        activityType: { type: string, enum: [DIRECT_CARE, COMMONS, FUTURE_CAPACITY]
}
        description: { type: string, maxLength: 2000 }
        evidenceRefs:
          type: array
          items: { type: string, description: "URI naar bewijsstuk" }
    Claim:
      type: object
      properties:
        claimId: { type: string, format: uuid }
        userId: { type: string, format: uuid }
        activityType: { type: string }
        description: { type: string }
        status: { type: string, enum: [SUBMITTED, ATTESTING, VALIDATING, APPROVED,
REJECTED] }
        submittedAt: { type: string, format: date-time }
    AttestationCreateRequest:
      type: object
      required: [claimId, confidence]
      properties:
        claimId: { type: string, format: uuid }
        confidence: { type: integer, minimum: 1, maximum: 5 }
        note: { type: string, maxLength: 1000 }
    LedgerTransaction:
      type: object
      properties:
        txId: { type: string, format: uuid }
        userId: { type: string, format: uuid }
```

```
        txType: { type: string, enum: [EARN, SPEND, EXPIRE, ADJUST] }
        amountRC: { type: number }
        timestamp: { type: string, format: date-time }
        expiryDate: { type: string, format: date }
        hash: { type: string }
        prevHash: { type: string }
    Balance:
      type: object
      properties:
        userId: { type: string, format: uuid }
        balanceRC: { type: number }
        expiringNext30DaysRC: { type: number }

paths:
  /claims:
    post:
      summary: Dien een RC-claim in
      requestBody:
        required: true
        content:
          application/json:
            schema: { $ref: '#/components/schemas/ClaimCreateRequest' }
      responses:
        "201":
          description: Claim aangemaakt
          content:
            application/json:
              schema: { $ref: '#/components/schemas/Claim' }
        "400": { description: Ongeldig verzoek, content: { application/json: {
schema: { $ref: '#/components/schemas/Error' } } } }
        "401": { description: Niet geautoriseerd }
    get:
      summary: Lijst claims (eigen of bevoegd)
      parameters:
        - in: query
          name: status
          schema: { type: string }
      responses:
        "200":
          description: Claims
          content:
            application/json:
              schema:
                type: array
                items: { $ref: '#/components/schemas/Claim' }

  /claims/{claimId}:
    get:
      summary: Haal claim op
      parameters:
        - in: path
          name: claimId
          required: true
          schema: { type: string, format: uuid }
      responses:
        "200":
          description: Claim
          content:
            application/json:
              schema: { $ref: '#/components/schemas/Claim' }
        "404": { description: Niet gevonden }

  /attestations:
    post:
```

```
    summary: Voeg attestatie toe aan claim
    requestBody:
      required: true
      content:
        application/json:
          schema: { $ref: '#/components/schemas/AttestationCreateRequest' }
    responses:
      "201": { description: Attestatie geregistreerd }
      "409": { description: Dubbele attestatie (claimId, attestor) }
      "403": { description: Niet bevoegd }

/ledger/transactions:
  get:
    summary: Ledger transacties (bevoegd)
    parameters:
      - in: query
        name: userId
        schema: { type: string, format: uuid }
      - in: query
        name: from
        schema: { type: string, format: date-time }
      - in: query
        name: to
        schema: { type: string, format: date-time }
    responses:
      "200":
        description: Transacties
        content:
          application/json:
            schema:
              type: array
              items: { $ref: '#/components/schemas/LedgerTransaction' }

/ledger/balance:
  get:
    summary: Huidig RC-saldo
    responses:
      "200":
        description: Saldo
        content:
          application/json:
            schema: { $ref: '#/components/schemas/Balance' }

/reports/ward:
  get:
    summary: Wijkrapportage (geaggregeerd)
    parameters:
      - in: query
        name: wardId
        required: true
        schema: { type: string }
      - in: query
        name: period
        schema: { type: string, example: "2026-H1" }
    responses:
      "200":
        description: Geaggregeerde wijkdata
        content:
          application/json:
            schema:
              type: object
              additionalProperties: true
```

C2. API-regels (security, audit, versioning)

C2. API-regels en compliance (samenvatting)
- Authenticatie: JWT bearer tokens; scopes per rol (burger/wijkraad/auditor)
- Autorisatie: RBAC + object-level checks (own-claims vs delegated)
- Idempotency: POST /claims ondersteunt Idempotency-Key header
- Rate limiting: per client/app om misbruik te beperken
- Auditability: alle mutaties loggen correlationId + traceId naar WORM audit log
- Versiebeheer: /v1, /v2…; breaking changes alleen via nieuwe major versie
- Foutafhandeling: uniforme Error schema + traceId voor forensische reconstructie