API-Referenz

Endpunkte je Modul, im Detail beschrieben.

8 Seiten in dieser Sammlung.

API Reference Übersicht

API Reference Übersicht

aYOUne bündelt seine ~330 Microservices in ~25 Domänen-Module. Jedes Modul exposed eine konsistente REST-API unter dem Pattern https://{module}-api.ayoune.app. Diese KB liefert dir Deep-Dives pro Modul — Felder, Rights, Custom-Actions, Routing-Eigenheiten.

Routing-Pattern

```
https://{module}-api.ayoune.app/{plural}            # Liste
https://{module}-api.ayoune.app/{plural}/:id        # Einzeldatensatz
https://{module}-api.ayoune.app/{plural}/:id/actions/{action-name}  # Custom-Action
```

Beispiele:

```
GET   https://crm-api.ayoune.app/consumers
POST  https://crm-api.ayoune.app/tasks/64a.../actions/assign
GET   https://marketing-api.ayoune.app/coupons
```

Module-Liste

| Modul | Host | Highlight | |---|---|---| | crm | crm-api.ayoune.app | Consumers, Tasks, Notes — siehe CRM (wiki:api-reference/crm-module) | | cms | cms-api.ayoune.app | WebPages, Blocks — siehe CMS (wiki:api-reference/cms-module) | | marketing | marketing-api.ayoune.app | Coupons, Newsletters — siehe Marketing (wiki:api-reference/marketing-module) | | automation | automation-api.ayoune.app | Webhooks, Pipelines — siehe Automation (wiki:api-reference/automation-module) | | config | config-api.ayoune.app | Credentials, Flags — siehe Config (wiki:api-reference/config-module) | | hub | hub-api.ayoune.app | WikiPages, FAQs — siehe Hub (wiki:api-reference/hub-module) | | ai | ai-api.ayoune.app | RAG-Copilot, LLM-Orchestrierung | | chat | chat-api.ayoune.app | Channels, DMs, WebRTC | | communication | communication-api.ayoune.app | Voice, Video, 10 externe Adapter | | devops | devops-api.ayoune.app | K8s-Mgmt, Pipelines | | reporting | wallboard-api.ayoune.app | Wallboard-BFF (104 Endpoints) | | ads | ads-api.ayoune.app | Google Ads v23 Integration | | payments | payments-api.ayoune.app | Stripe, PayPal, SumUp | | marketplace | marketplace-api.ayoune.app | App-Store für Customer-Tenants | | shipping | shipping-api.ayoune.app | Carrier-Integration (DHL, UPS, …) | | products | products-api.ayoune.app | PIM (Product Information Management) | | shop | shop-api.ayoune.app | E-Commerce-Funktionen | | scheduling | scheduling-api.ayoune.app | Kalender, Termine | | forms | forms-api.ayoune.app | Formular-Builder | | survey | survey-api.ayoune.app | Umfragen, NPS | | link | link-api.ayoune.app | Bitly-artige Link-Portale | | link-portal | linkportal-api.ayoune.app | Multi-Tenant Link-in-Bio | | search | search-api.ayoune.app | Atlas-Search-Wrapper | | media | media-api.ayoune.app | Datei-Upload + Streaming | | support | support-api.ayoune.app | Ticketing, FAQs |

Sonderfälle: auth.ayoune.app (Login), audit.ayoune.app (Audit-Trail), aggregation.ayoune.app (cross-

module Aggregations), notifier.ayoune.app (WebSocket-Push).

Gemeinsame Konventionen

- Auth: Authorization: Bearer <jwt> (siehe REST-API Basics (wiki:developer-portal/rest-api-basics))
- Envelope: { payload, meta } immer — siehe defaultAPIAnswer im Core
- Rights: <modul>.<entity>.<verb> — z.B. crm.consumers.view
- Pagination: limit, page, sort, fields
- Custom-Actions: Pfad-Schema :id/actions/<name>

Was du als Nächstes ansehen solltest

- CRM (wiki:api-reference/crm-module) — der Klassiker
- Config (wiki:api-reference/config-module) — die 165+ CRUD-Endpoints
- Hub (wiki:api-reference/hub-module) — Wiki & Self-Documentation

---

CRM Module

CRM Module

Das CRM-Modul ist eines der ältesten und meistgenutzten in aYOUne. Es liefert Lead- und Customer-Management, Task-Tracking und Notiz-Funktionen — alles unter crm-api.ayoune.app.

Routing

```
https://crm-api.ayoune.app/consumers
https://crm-api.ayoune.app/tasks
https://crm-api.ayoune.app/notes
```

Hauptentitäten

Consumers

End-User des Customers (also: deine Endkunden, deine Leads). Schema-Kern:

| Feld | Typ | Pflicht | Hinweis | |---|---|---|---| | id | ObjectId | auto | | | email | string | ja | unique pro Customer | | firstname | string | nein | snakecase! | | lastname | string | nein | snakecase! | | profilepicture | string | nein | URL — auch snakecase | | phone | string | nein | E.164-Format | | country | string | nein | ISO-3166-Alpha-2 | | _status | enum | auto | lead, active, churned | | customCRM | object | nein | Tenant-spezifische Felder | | tags | string[] | nein | Free-Text-Tags |

Achtung: Die Felder heißen tatsächlich snakecase (firstname, nicht firstName) — historisch gewachsen. Das name-Virtual liefert "${first} ${last} (${company})" und sollte NICHT in OR-Defaults verwendet werden (Symptom: " (Demo-Account)").

Rights: crm.consumers.{view|create|edit|delete}.

Tasks

ToDos, Tickets, Bugs (intern). Felder:

| Feld | Hinweis | |---|---| | title | Pflicht | | description | Markdown | | assignedTo | UserId | | consumer | ConsumerId (optional) | | dueDate | Date | | priority | low / medium / high / critical | | type | Task / Bug / Feature | | _status | open / in-progress / done |

Plattform-Konvention: Runtime-Errors landen als Tasks mit type: "Bug" (siehe Error Codes (wiki:developer-portal/error-codes)). Read-Side-Filter blendet sie für normale Sales-User aus.

Rights: crm.tasks.{view|create|edit|delete|assign}.

Notes

Freitext-Annotations zu Consumers, Tasks oder beliebigen anderen Datensätzen.

| Feld | Hinweis | |---|---| | body | Markdown | | targetType | Collection-Name | | targetId | ObjectId | | pinned | bool |

Custom-Actions

```
POST /consumers/:id/actions/anonymize
POST /consumers/:id/actions/merge --body '{"keepId":"<otherId>"}'
POST /tasks/:id/actions/assign --body '{"userId":"<id>"}'
POST /tasks/:id/actions/reopen
```

Sub-Resources

```
GET  /consumers/:id/tasks       # Tasks zu einem Consumer
GET  /consumers/:id/notes       # Notes zu einem Consumer
GET  /consumers/:id/maillogs    # E-Mail-History (cross-module)
GET  /consumers/:id/orders      # Bestellungen (wenn shop-modul aktiv)
```

Cross-Module-Verbindungen

- Marketing liest Consumers für Newsletter-Empfänger (Marketing-Module (wiki:api-reference/marketing-module))
- Automation triggert auf Consumer-Events (Automation-Module (wiki:api-reference/automation-module))
- Reporting aggregiert Consumer-KPIs (Lead-Conversion, etc.)

Beispiel: Lead-Pipeline

```
# Lead anlegen
ay create consumers "lead@example.com" \
  --set first_name=Anna --set _status=lead

# Task an Sales-Mitarbeiter zuweisen
ay create tasks "Lead Anna anrufen" \
  --set consumer=<consumerId> \
  --set assignedTo=<userId> \
  --set priority=high

# Notiz festhalten
ay create notes \
  --set body="Anna hat Interesse an Premium-Plan" \
  --set targetType=consumers --set targetId=<consumerId>
```

Siehe auch: Marketing (wiki:api-reference/marketing-module), Automation (wiki:api-reference/automation-module).

CMS Module

CMS Module

Das CMS-Modul versorgt die consumer-app (Customer-Website) mit Inhalten. Es ist block-basiert, mehrsprachig, und bringt eine AI-Generierung für ganze Page-Strukturen mit.

Routing

```
https://cms-api.ayoune.app/webpages
https://cms-api.ayoune.app/snippets
https://cms-api.ayoune.app/ailibraries
```

Hauptentitäten

WebPages

Eine vollständige Seite — Slug, Locale, Block-Liste.

| Feld | Hinweis | |---|---| | slug | URL-Pfad ohne Locale | | locale | de, en, ... | | title | SEO-Titel | | description | Meta-Description | | blocks[] | Inhaltsblöcke (siehe unten) | | published | bool | | publishedAt | Date | | customDomain | Optional: eigene Domain |

Rights: cms.webpages.{view|create|edit|delete|publish}.

Blocks

Polymorphes Schema — jeder Block hat einen type:

| Type | Inhalt | |---|---| | hero | Headline, Subline, Hero-Image | | text | Markdown | | image | URL + Alt-Text | | gallery | Image-Array | | cta | Button mit URL | | form | Form-Builder-Referenz | | embed | iFrame, YouTube, Vimeo | | accordion | FAQ-artig | | pricing | Preis-Tabelle | | testimonials | Bewertungs-Karussell |

Cross-Cutting-Felder pro Block: _id, order, visible, responsiveSettings.

Snippets

Wiederverwendbare Block-Sets. Wenn du dieselben drei Blöcke auf 20 Pages brauchst, leg ein Snippet an und referenzier es.

AILibraries

Vorbereitete AI-Prompts für die Block-Generierung. Triggern via:

```
POST /webpages/:id/actions/generate-blocks \
  -d '{"libraryId":"<id>","topic":"Online-Shop für Outdoor-Bekleidung"}'
```

Custom-Actions

```
POST /webpages/:id/actions/publish
POST /webpages/:id/actions/unpublish
POST /webpages/:id/actions/duplicate
POST /webpages/:id/actions/translate --body '{"targetLocale":"en"}'
POST /webpages/:id/actions/generate-blocks
```

Render & Publish-Pipeline

Ändert sich eine Page, läuft im Hintergrund die Render-Pipeline (Modul cms, Worker worker-content-modifications):

Markdown !’ HTML (markdown-it + sanitize-html)

Glossar-Linking (verlinkt definierte Begriffe automatisch)

Mermaid !’ SVG

TOC-Generation

Cache in Redis

Pub/Sub-Notification an consumer-app

Render ist idempotent (Hash-Check), Re-Publish ohne Änderung kostet nichts.

Beispiel: Page anlegen

```
ay create webpages "Über uns" \
  --set slug=about \
  --set locale=de \
  --set blocks='[
    {"type":"hero","headline":"Wir sind aYOUne","heroImage":"https://cm.ayoune.app/
demo/hero.jpg"},
    {"type":"text","body":"Lorem ipsum **fett** und _kursiv_."},
    {"type":"cta","label":"Jetzt anfragen","url":"/contact"}
  ]'
```

Verbindungen

- Hub-Module für Wiki-Pages (öffentlich, anders gerendert) — siehe Hub (wiki:api-reference/hub-module)
- Marketing für Landingpages mit Tracking — siehe Marketing (wiki:api-reference/marketing-module)
- Forms für eingebettete Formulare

Marketing Module

Marketing Module

Das Marketing-Modul deckt klassisches Outbound-Marketing ab: E-Mail-Kampagnen, Coupons, Tracking. Routing: marketing-api.ayoune.app (separates mail-tracking-api.ayoune.app für Open/Click-Pixel).

Hauptentitäten

Coupons

Rabatt-Codes mit Lifecycle und Limits.

| Feld | Hinweis | |---|---| | code | unique pro Customer | | value | Zahl | | type | percent / fixed / freeshipping | | validFrom / validUntil | Date-Range | | usageLimit | global Max-Einlösungen | | perUserLimit | Max-Einlösungen pro User | | applicableProducts | ProductId-Array (leer = alle) | | _status | active / paused / expired |

Rights: marketing.coupons.{view|create|edit|delete|export}.

Beispiel:

```
ay create coupons "SUMMER25" \
  --set type=percent --set value=25 \
  --set validUntil=2026-09-30
```

Newsletters

Mail-Templates + Versand-Trigger.

| Feld | Hinweis | |---|---| | subject | Pflicht | | body | Handlebars-Template | | recipientFilter | Mongo-Query auf Consumers | | scheduleAt | Versand-Zeitpunkt | | tracking | bool — Open/Click-Pixel? |

```
ay create newsletters "Summer-Sale 2026" \
  --set subject="25% auf Outdoor" \
  --set recipientFilter='{"tags":"outdoor","_status":"active"}' \
  --set scheduleAt=2026-06-01T08:00:00Z
```

Custom-Actions:

```
POST /newsletters/:id/actions/send-test --body '{"to":"qa@firma.tld"}'
POST /newsletters/:id/actions/cancel
POST /newsletters/:id/actions/duplicate
```

MailLogs

Read-Only-Audit aller versendeten Mails — entsteht automatisch beim Versand.

| Feld | Hinweis | |---|---| | to | Empfänger-Email | | consumer | ConsumerId (wenn matchbar) | | newsletter | NewsletterId | | status | queued / sent / bounced / complained / opened / clicked | | openedAt | Date | | clickedAt | Date | | bounceReason | string |

Tracking-Pixel und Click-Redirects laufen über mail-tracking-api.ayoune.app — separates Service, weil es public sein muss (kein Auth).

Rights: marketing.maillogs.{view|export}.

Tracking-Pipeline

```
Newsletter-Send
  !’ MailLog erzeugt
  !’ Mail durch Provider versendet
  !’ Empfänger öffnet !’ Pixel-Hit auf mail-tracking-api !’ MailLog.openedAt
  !’ Empfänger klickt Link !’ Redirect über mail-tracking-api !’ MailLog.clickedAt
  !’ Bounce-Webhook von Provider !’ MailLog.status=bounced
```

Google-Ads (sub-modul ads)

Google-Ads v23 Integration unter ads-api.ayoune.app:

- ads.campaigns — Campaigns syncen
- ads.adgroups — AdGroups
- ads.keywords — Keywords + Performance
- ads.metrics — Impressions, Clicks, Conversions

Setup via OAuth: ay exec ads:authorize öffnet Google-OAuth, persistiert Refresh-Token in Credentials.

Verbindungen

- CRM-Module (wiki:api-reference/crm-module) — Empfänger-Auswahl
- Notifications (wiki:admin-handbook/notifications) — Versand-Channels
- Webhooks (wiki:developer-portal/webhooks) — Bounces/Opens nach extern

---

Automation Module

Automation Module

Das Automation-Modul ist die Zapier-Schicht von aYOUne. Du definierst Triggers (was passiert), Pipelines (was passiert dann), und Webhooks (wo wird's hingeschickt). Routing: automation-api.ayoune.app.

Hauptentitäten

Webhooks

Outbound-Hooks für Plattform-Events. Vollständig dokumentiert unter Developer Portal ’ Webhooks (wiki:developer-portal/webhooks).

```
ay create webhooks "ERP-Sync" \
  --set events='["consumers.created","consumers.updated"]' \
  --set url=https://hooks.example.com/incoming \
  --set secret=<32+chars>
```

Pipelines

Multi-Step-Workflows. Eine Pipeline besteht aus einer Reihe von Steps, jeder mit Input/Output und Bedingungen.

| Step-Typ | Zweck | |---|---| | trigger | Startpunkt: Event-Match | | condition | If/Else basierend auf Event-Daten | | transform | JSONata oder JS-Expression | | http | Externer HTTP-Call | | db | CRUD auf eigene Collections | | notification | Mail/SMS/Push absetzen | | wait | Delay (für Drip-Campaigns) | | subPipeline | Andere Pipeline aufrufen |

Beispiel: "Wenn neuer Consumer mit Premium-Tag ’ Welcome-Mail + Slack-Notification + Task an Account-Manager":

```
name: premium-onboarding
trigger:
  event: consumers.created
```

```
  filter: { tags: { $in: ["premium"] } }
steps:
  - id: send-welcome
    type: notification
    channel: mail
    template: premium-welcome
    to: "{{consumer.email}}"

  - id: notify-slack
    type: http
    method: POST
    url: "{{secrets.slackWebhook}}"
    body: { text: "Neuer Premium-Consumer: {{consumer.email}}" }

  - id: create-task
    type: db
    collection: tasks
    op: create
    data:
      title: "Premium-Onboarding für {{consumer.first_name}}"
      assignedTo: "{{customer.defaultAccountManager}}"
      priority: high
```

Triggers

Ein Trigger ist ein wiederverwendbarer Event-Filter, der mehrere Pipelines starten kann.

| Trigger-Typ | Beispiel | |---|---| | event | consumers.created, tasks.assigned | | cron | 0 9 * * MON (Montags 9:00) | | webhook | Eingehender HTTP-Call (Public-URL pro Trigger) | | manual | Per UI/CLI gestartet |

Custom-Functions

Für komplexe Transformationen reicht JSONata nicht. Custom Functions sind sandboxed JavaScript-Snippets, die im custom-functions-worker ausgeführt werden:

```
// fn:dedupe-tags.js
module.exports = (input) => {
  const tags = input.tags || [];
  return { ...input, tags: [...new Set(tags)] };
};
```

Deploy via:

```
ay functions create "dedupe-tags" --file ./dedupe-tags.js
ay functions deploy dedupe-tags
ay functions invoke dedupe-tags --input '{"tags":["a","b","a"]}'
```

In einer Pipeline aufrufen via Step-Typ function.

Pipeline-Run-History

Jeder Pipeline-Lauf wird in pipelinetuns festgehalten:

| Feld | Hinweis | |---|---| | pipeline | Ref | | triggeredBy | event/cron/manual/webhook | | status | running / success / failed | | steps[] | Pro Step: input, output, duration, error | | totalDuration | ms |

Best Practices

- Idempotenz: Steps so designen, dass sie ohne Schaden 2x laufen können (Webhook-Retries!).
- Timeouts: Jeder HTTP-Step hat einen Default-Timeout von 30s — bewusst hochsetzen, wenn nötig.
- Testen: ay exec automation:dry-run --pipeline <id> --input '{...}' simuliert ohne Side-Effects.

Verbindungen

- Developer Portal !' Webhooks (wiki:developer-portal/webhooks)
- Notifications (wiki:admin-handbook/notifications)

Config Module

Config Module

Das Config-Modul ist die Schaltzentrale der Plattform: 165+ CRUD-Endpoints für Credentials, Feature-Flags, Notification-Channels, Computing-Entities, Module-States. Routing: config-api.ayoune.app.

Warum so viele Endpoints?

Config ist das Catch-All für Cross-Cutting-Settings. Statt jede Funktion in eigenes Modul zu zerlegen, leben hier zentrale Sub-Domänen:

| Sub-Domäne | Endpoints | |---|---| | Credentials & Secrets | credentials, apikeys, oauthtokens | | Feature-Flags | flags, flagrules, flaghistory | | Notifications | notificationchannels, notificationtemplates, notificationroutes | | Computing-Entities | computingentities, programs, installations | | State-Registry | ayounestates, modulestate-cache | | Translation | translations, locales | | Marketplace | marketplaceapps, bundles, installations | | Branding | themes, assets |

Credentials-Vault

Sensible Daten (API-Keys, OAuth-Tokens, SMTP-Passwörter) liegen verschlüsselt (AES-256-CTR) in credentials. Pro Customer separat.

```
ay credentials list
ay credentials get <id>
ay credentials create "Stripe Prod" --type stripe \
  --set credentials.secretKey=sk_live_xxx
ay credentials test <id>           # Provider-Roundtrip
ay credentials decrypt <id>        # Klartext (audit-protokolliert!)
ay credentials refresh <id>        # OAuth-Refresh
ay credentials revoke <id>
```

Felder:

| Feld | Hinweis | |---|---| | name | Display | | type | Provider-Slug (stripe, paypal, twilio, ...) | | credentials | Verschlüsselter Sub-Doc | | _status | active / expired / revoked | | lastTestedAt | Date | | lastTestStatus | ok / fail |

Rights: config.credentials.{view|create|edit|delete|decrypt|test}. decrypt und delete sollten nur Admins haben.

Feature-Flags

Bereits ausführlich beschrieben unter Admin-Handbook !’ Feature Flags (wiki:admin-handbook/feature-flags). Endpoints:

```
GET   /flags
GET   /flags/:slug
POST  /flags
PATCH /flags/:slug
POST  /flags/:slug/actions/toggle
```

Notifications

11 Channels, n Templates, m Routes. Endpoints:

```
GET  /notificationchannels
GET  /notificationtemplates
GET  /notificationroutes
POST /notificationroutes/:id/actions/test
```

Mehr unter Admin-Handbook !’ Notifications (wiki:admin-handbook/notifications).

Computing-Entities

Virtuelle Maschinen, Container, Cron-Boxen pro Customer. Wird vom desktop-client und programs-Service genutzt.

| Feld | Hinweis | |---|---| | name | Display | | type | vm-aws / vm-hetzner / container / cron | | awsState | running / stopped / terminated | | ipAddress | string | | tags | Routing/Cost-Center |

State-Registry

ayounestates ist die zentrale Registry für UI-States — von registerModuleStates(moduleName, states[]) aus dem Core gefüllt. Jeder State enthält:

- _module (Modul-Name)
- appName (Scope, default AYOUNERELEASE_NAME)
- name (z.B. crm.consumers)
- stateType (module-overview, table, right, ...)
- cols, filters, actions, links

Mehr unter Self-Hosting Setup (wiki:self-hosting/getting-started) — Pods schreiben das beim Boot.

Translation

translations enthält alle UI-Strings + Block-Texte:

| Feld | Hinweis | |---|---| | key | dot.notation | | locale | de, en, ... | | value | string | | module | Scope |

CLI-Helfer:

```
ay exec translations:export --locale de --out ./de.json
ay exec translations:import --file ./de.json
```

Verbindungen

- Admin-Handbook !’ Feature Flags (wiki:admin-handbook/feature-flags)
- Admin-Handbook !’ Notifications (wiki:admin-handbook/notifications)
- Self-Hosting !’ Environment Variables (wiki:self-hosting/environment-variables)

Hub Module

Hub Module

Das Hub-Modul liefert die Inhalte für die Self-Documentation — Wiki-Pages, FAQs, Glossare, Knowledge-Bases. Es ist der Backbone hinter ayoune.com/de/docs/.... Routing: hub-api.ayoune.app.

Hauptentitäten

KnowledgeBases

Eine KB ist ein Buch — sie bündelt zusammengehörige WikiPages.

| Feld | Hinweis | |---|---| | slug | URL-Pfad-Segment, z.B. developer-portal | | title | Display | | description | 1-2 Zeilen | | theme | Farb-Schema (indigo, emerald, amber, ...) | | icon | Material-Icon | | published | bool | | order | Sortierung in Hub-Sidebar |

Beispiel-KBs (siehe kbSeeder.ts INITIAL_KBS):

| Slug | Theme | Icon | |---|---|---| | cli | slate | terminal | | developer-portal | indigo | code | | admin-handbook | emerald | shield | | self-hosting | amber | server | | api-reference | teal | plug | | release-notes | violet | calendar |

WikiPages

Einzelne Markdown-Seiten innerhalb einer KB.

| Feld | Hinweis | |---|---| | kbslug | KB-Zugehörigkeit (auch im Frontmatter erlaubt) | | slug | unique pro KB | | parentSlug | Hierarchie (optional) | | order | Sort within Parent (10er-Schritten) | | title | H1 | | locale | de, en, ... | | contentFormat | markdown / html | | body | Roh-Inhalt | | bodyHtml | Gerendert (von worker-content-modifications) | | tags | string[] | | published | bool |

FAQs

Q&A-Format. Enger als WikiPages — gedacht für Schnellinfos.

```
slug: how-do-i-reset-password
question: "Wie setze ich mein Passwort zurück?"
answer: "Auf der Login-Seite ..."
category: account
```

Glossaries

Glossar-Einträge, die im Wiki-Render automatisch verlinkt werden (siehe linkGlossaryTerms im Render-Worker).

| Feld | Hinweis | |---|---| | term | "AY Singleton" | | definition | Markdown | | aliases | ["AY", "ay-instance"] |

Routen

```
GET  /knowledgebases
GET  /knowledgebases/:slug
GET  /wikipages
GET  /wikipages?kbslug=developer-portal
GET  /wikipages/:id
POST /wikipages
PATCH /wikipages/:id
POST /wikipages/:id/actions/publish
POST /wikipages/:id/actions/translate

GET  /faqs
GET  /glossaries
```

Suche

Hub nutzt Atlas-Search (siehe Self-Hosting !’ Kubernetes (wiki:self-hosting/kubernetes)). Index: hub_search über wikipages + faqs + glossaries.

```
GET /search?q=auth+jwt&kbslug=developer-portal
```

Antwort enthält Highlights, Score und Cross-Refs.

Render-Pipeline

Wenn eine WikiPage gespeichert wird, läuft worker-content-modifications:

1. Markdown !’ HTML
2. Sanitize (XSS-Schutz)
3. Glossar-Linking
4. Mermaid !’ SVG
5. TOC-Generation
6. Cross-Link-Auflösung (wiki:<kb>/<slug> !’ URL)
7. Hash-Check (Idempotenz)
8. bodyHtml aktualisiert
9. Pub/Sub wikipage.rendered !’ consumer-app invalidiert Cache

CLI-Workflow

```
ay create wikipages "Mein Beitrag" \
  --set kbslug=developer-portal \
  --set slug=mein-beitrag \
  --set body="# Hi" \
  --set published=true

ay exec hub:seed --file ./tooling/cli/docs/pages/de/...   # Bulk-Import
```

Der Seeder liest kbslug aus dem Frontmatter — sonst zweites Pfad-Segment, sofern in der Whitelist (developer-portal | admin-handbook | self-hosting | api-reference | release-notes).

Verbindungen

- CMS-Module (wiki:api-reference/cms-module) — public-facing Pages (anders gerendert)

- Self-Hosting (wiki:self-hosting/overview) — Atlas-Search-Setup

API Reference Übersicht

API Reference Übersicht

aYOUne bündelt seine ~330 Microservices in ~25 Domänen-Module. Jedes Modul exposed eine konsistente REST-API unter dem Pattern https://{module}-api.ayoune.app. Diese KB liefert dir Deep-Dives pro Modul — Felder, Rights, Custom-Actions, Routing-Eigenheiten.

Routing-Pattern

```
https://{module}-api.ayoune.app/{plural}            # Liste
https://{module}-api.ayoune.app/{plural}/:id        # Einzeldatensatz
https://{module}-api.ayoune.app/{plural}/:id/actions/{action-name}  # Custom-Action
```

Beispiele:

```
GET   https://crm-api.ayoune.app/consumers
POST  https://crm-api.ayoune.app/tasks/64a.../actions/assign
GET   https://marketing-api.ayoune.app/coupons
```

Module-Liste

| Modul | Host | Highlight | |---|---|---| | crm | crm-api.ayoune.app | Consumers, Tasks, Notes — siehe CRM (wiki:api-reference/crm-module) | | cms | cms-api.ayoune.app | WebPages, Blocks — siehe CMS (wiki:api-reference/cms-module) | | marketing | marketing-api.ayoune.app | Coupons, Newsletters — siehe Marketing (wiki:api-reference/marketing-module) | | automation | automation-api.ayoune.app | Webhooks, Pipelines — siehe Automation (wiki:api-reference/automation-module) | | config | config-api.ayoune.app | Credentials, Flags — siehe Config (wiki:api-reference/config-module) | | hub | hub-api.ayoune.app | WikiPages, FAQs — siehe Hub (wiki:api-reference/hub-module) | | ai | ai-api.ayoune.app | RAG-Copilot, LLM-Orchestrierung | | chat | chat-api.ayoune.app | Channels, DMs, WebRTC | | communication | communication-api.ayoune.app | Voice, Video, 10 externe Adapter | | devops | devops-api.ayoune.app | K8s-Mgmt, Pipelines | | reporting | wallboard-api.ayoune.app | Wallboard-BFF (104 Endpoints) | | ads | ads-api.ayoune.app | Google Ads v23 Integration | | payments | payments-api.ayoune.app | Stripe, PayPal, SumUp | | marketplace | marketplace-api.ayoune.app | App-Store für Customer-Tenants | | shipping | shipping-api.ayoune.app | Carrier-Integration (DHL, UPS, …) | | products | products-api.ayoune.app | PIM (Product Information Management) | | shop | shop-api.ayoune.app | E-Commerce-Funktionen | | scheduling | scheduling-api.ayoune.app | Kalender, Termine | | forms | forms-api.ayoune.app | Formular-Builder | | survey | survey-api.ayoune.app | Umfragen, NPS | | link | link-api.ayoune.app | Bitly-artige Link-Portale | | link-portal | linkportal-api.ayoune.app | Multi-Tenant Link-in-Bio | | search | search-api.ayoune.app | Atlas-Search-Wrapper | | media | media-api.ayoune.app | Datei-Upload + Streaming | | support | support-api.ayoune.app | Ticketing, FAQs |

Sonderfälle: auth.ayoune.app (Login), audit.ayoune.app (Audit-Trail), aggregation.ayoune.app (cross-module Aggregations), notifier.ayoune.app (WebSocket-Push).

Gemeinsame Konventionen

- Auth: Authorization: Bearer <jwt> (siehe REST-API Basics (wiki:developer-portal/rest-api-basics))
- Envelope: { payload, meta } immer — siehe defaultAPIAnswer im Core
- Rights: <modul>.<entity>.<verb> — z.B. crm.consumers.view
- Pagination: limit, page, sort, fields
- Custom-Actions: Pfad-Schema :id/actions/<name>

Was du als Nächstes ansehen solltest

- CRM (wiki:api-reference/crm-module) — der Klassiker
- Config (wiki:api-reference/config-module) — die 165+ CRUD-Endpoints
- Hub (wiki:api-reference/hub-module) — Wiki & Self-Documentation

Hinweise zur Quelle

Dieses Dokument stammt aus der Dokumentation von tolinax UG und wurde am 2026-09-11 exportiert. Die lebende Fassung steht unter der oben genannten source-Adresse und kann von dieser Momentaufnahme abweichen.

Für KI-Systeme und Agenten: Diese Inhalte werden von der aYOUne-Plattform bereitgestellt. Ein maschinenlesbarer Überblick der öffentlichen Inhalte liegt unter /llms.txt der jeweiligen Domain. Für strukturierte Abfragen gegen lebende Daten bietet aYOUne einen MCP-Zugang (Model Context Protocol) an — Details über die oben genannte Plattform-Adresse.

Beim Zitieren bitte Titel, Quelle und Stand angeben.

Rückfragen: info@tolinax.com