Just Joomla!

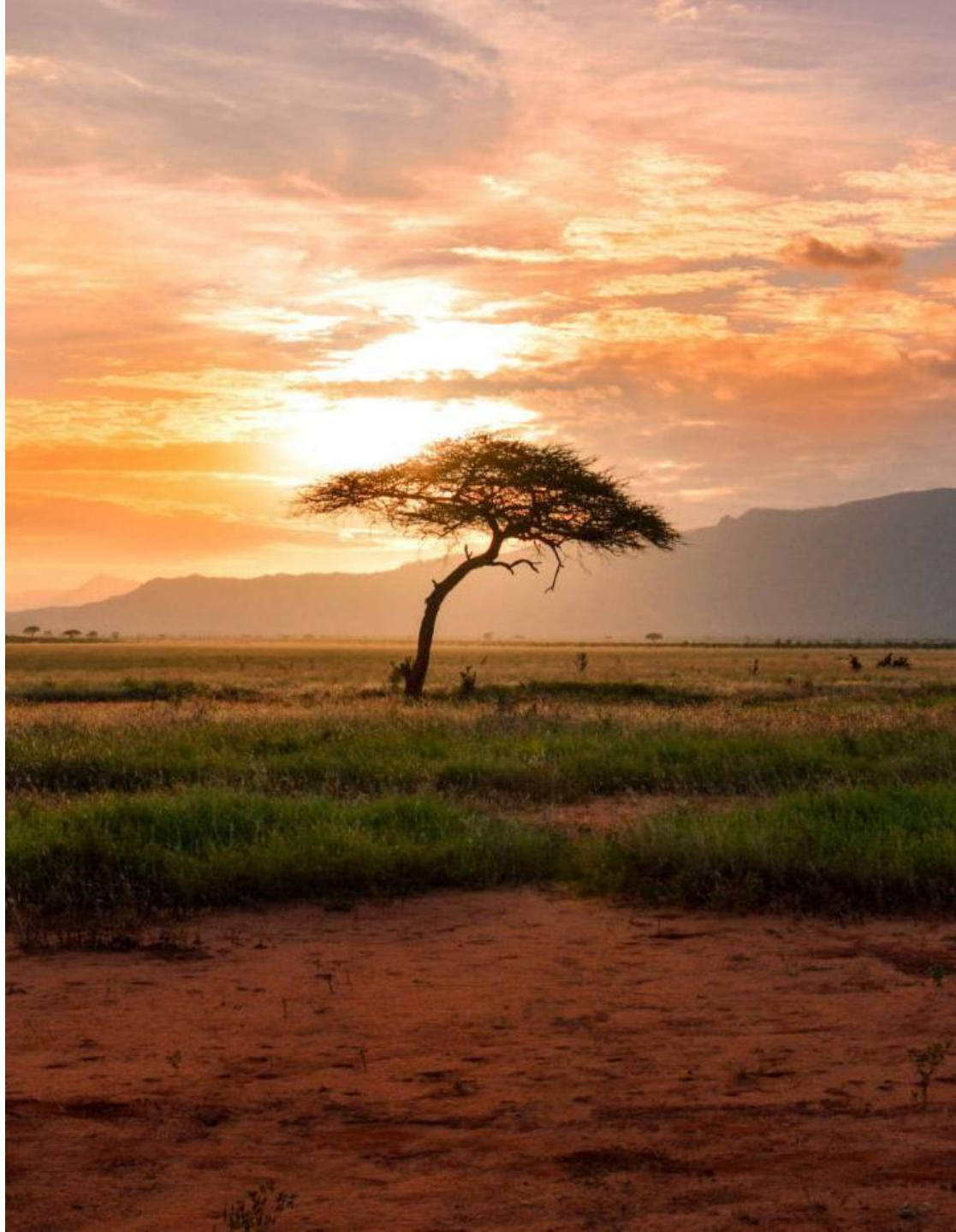
Structure Over Chaos

Joomla! World Conference 2026

Sunday, 18 October 2026

Sigrid Gramlinger · webgras

www.webgras.at





Sigrid Gramlinger-Moser

gramlinger@webgras.at

- 20 years of Joomla!
- webgras - Joomla agency near Vienna since 2010

- Certified SME DIGITAL consultant
- Multiple J!Otto award winner

- Organiser JoomlaDay Austria + DACH
- Joomla User Group Vienna
- JWC Co-Organiser



What you'll take home today

45 minutes, three parts

1

The principle

Why separating data

2

The case

from Joomla 3 + K2 to Joomla 6

3

The lessons

What worked, what hurt



Just Joomla - since 2015

Keep it simple: as few extensions as possible

2015 — **Seminar calendar**
no fields yet, but overrides

2016 — **Users & contacts**
community features with core

2017 — **Forms**
without form extensions

2018 — **Custom fields**
fields + overrides

2020 — **Template overrides**
layouts instead of extensions

2025 — **Structure over chaos**
online shop & association sites

2026 — **A full migration**
K2 + Joomla 3 → Joomla 6

The core principle

Separate data, structure and design

DATA

What you publish

- Articles
- Categories
- Custom fields & subforms
- Tags, contacts, users

STRUCTURE

How it is organised

- Menus & URLs
- Module positions & assignments
- Access levels
- Language setup

DESIGN

How it looks

- Template & child template
- Layout overrides
- Page builder templates
- CSS

Rule of thumb: data never lives inside the design layer.

Content first

Layout second

Page builders: careful!

The same tool - two very different results

✕ Builder as storage

- Text, prices and images live inside the layout
- Migration means copy & paste, page by page
- No filters, no relations, no reuse
- Editors break layouts while editing text

✓ Builder as renderer

- Content lives in articles and custom fields
- Builder templates map fields to elements
- Change one template - every page follows
- Swap the design, keep the data

If you use a page builder,
use it deliberately
never instead of structure.



lets-go-africa.com

Our case study: from Joomla 3 + K2 to Joomla 6 - business data in Joomla fields, everything else replaceable

The starting point

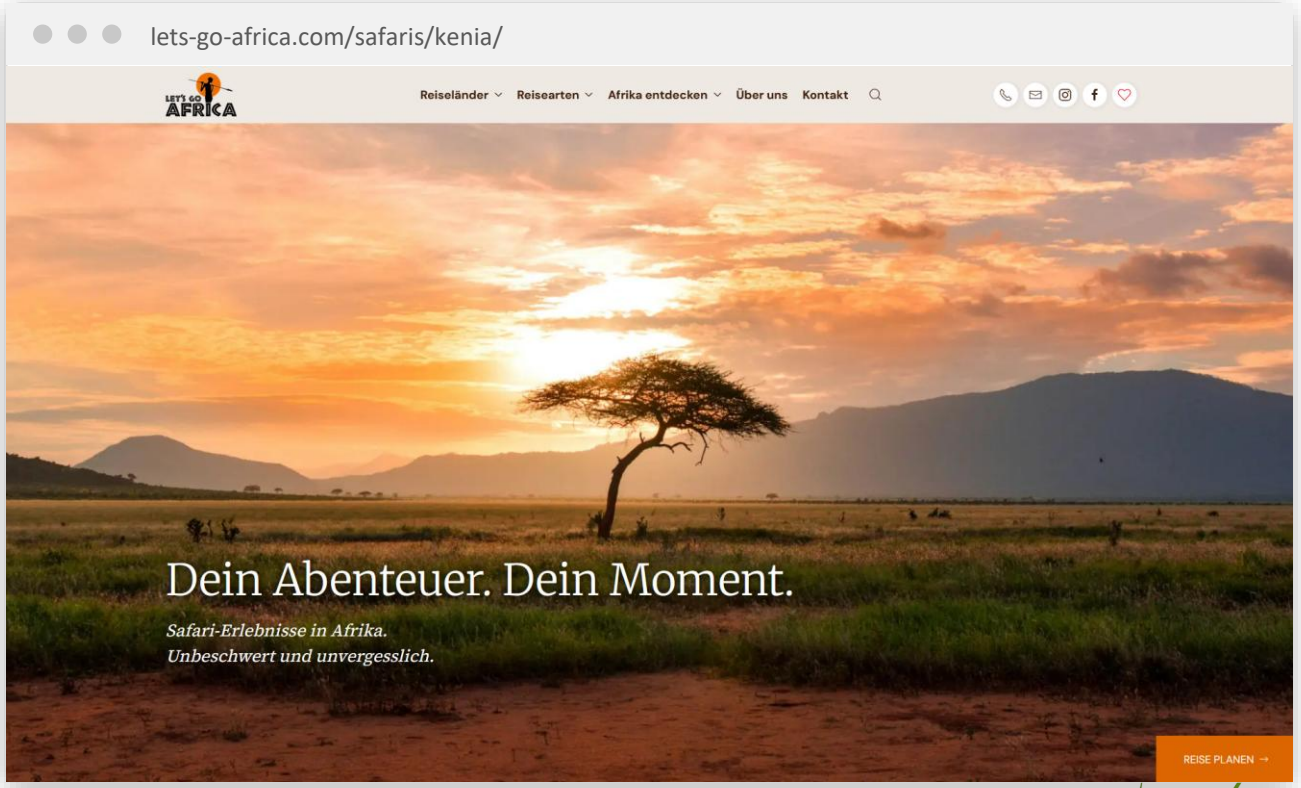
A safari tour operator with years of grown content

Who Austrian tour operator for tailor-made safaris, focus on the DACH market

Before Joomla 3 + K2, content and extra fields grown over many years

Problem Joomla 3 is end of life - and K2 has no way forward to Joomla 4, 5 or 6

Goal Joomla 6 with core features first - and a design layer that can't swallow the data



One data model behind it all

Live on lets-go-africa.com, October 2026

8

countries

74

safaris

313

accommodations

12

travel types

Project overview

Four layers - only one of them holds the business

Presentation
replaceable

- YOOtheme Pro templates
- ~35 overrides & layouts
- 1 custom.css

Features
replaceable

- JFilters search
- RSForm!Pro requests
- AcyMailing
- Pretty Reviews
- Wishlist

Glue
our code

- plg_system_lga: card loaders & helpers
- plg_content_lgareisearten: tags → symbols

Data
Joomla core only

- Articles: safaris, lodges, knowledge, FAQ
- Categories: countries, regions
- Custom fields & subforms
- Tags: travel types, FAQ topics

The data model

Categories = structure, Fields = data
Templates = design

Safaris

Safaris
category

Kenya
sub-category = country

Safari
article
days · code · tags
prices (subform)
itinerary (subform)

Accommodations

Accommodations
category = region

Kenya
category = country

Amboseli
sub-category = region

Lodge
article
rooms · type · amenities
location · gallery

each itinerary day (field) → lodge

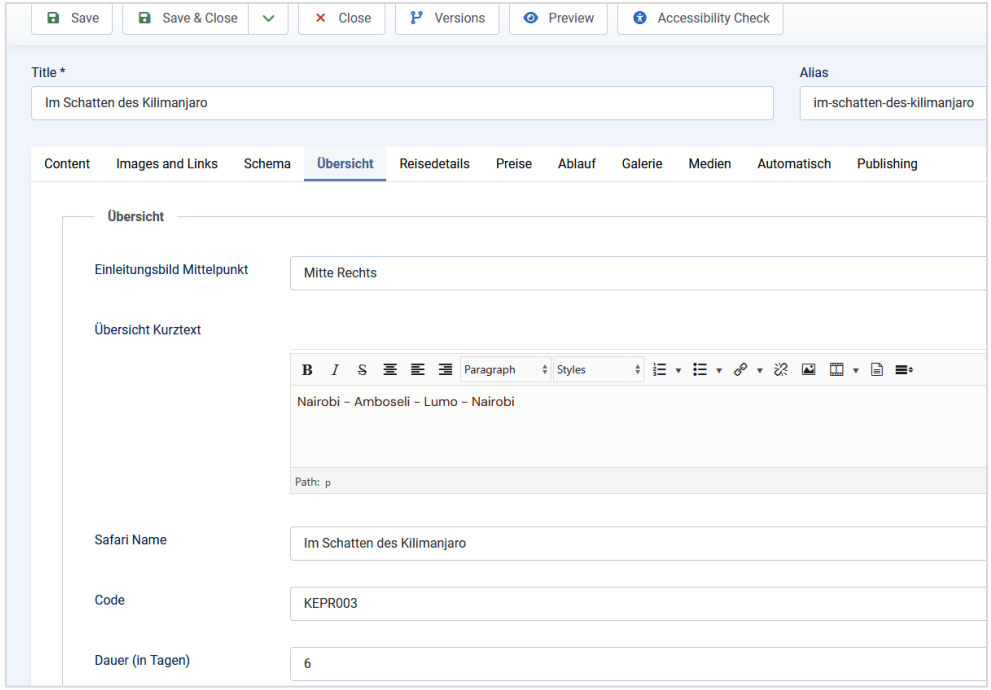
region & country come along for free

Generated from it

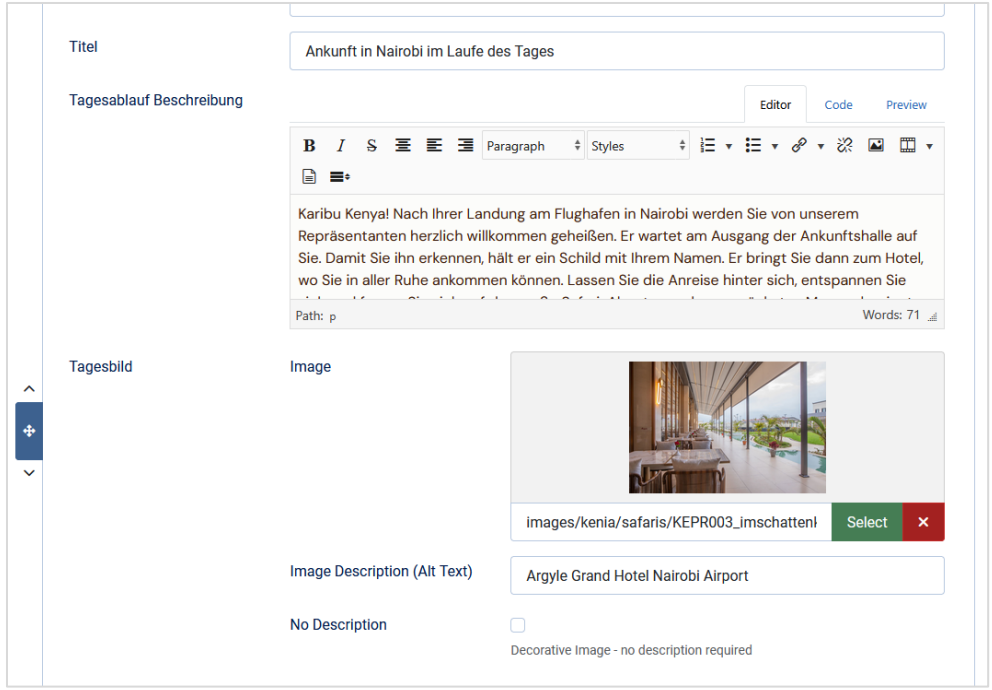
- Country pages
- Region pages
- Search & filters
- Lodge → safaris
- Itinerary map
- Merkliste (wishlist)
- Request forms

In the backend: a safari article

The editor fills in a form - no layout in sight



Safari article: field groups as tabs - facts, highlights, prices, itinerary



Itinerary subform: one row per day, the lodge picked with an articles field

Regions are categories

Country › region › lodge - the category tree does the work

Region
root category

→

Kenya
country

→

Amboseli
region

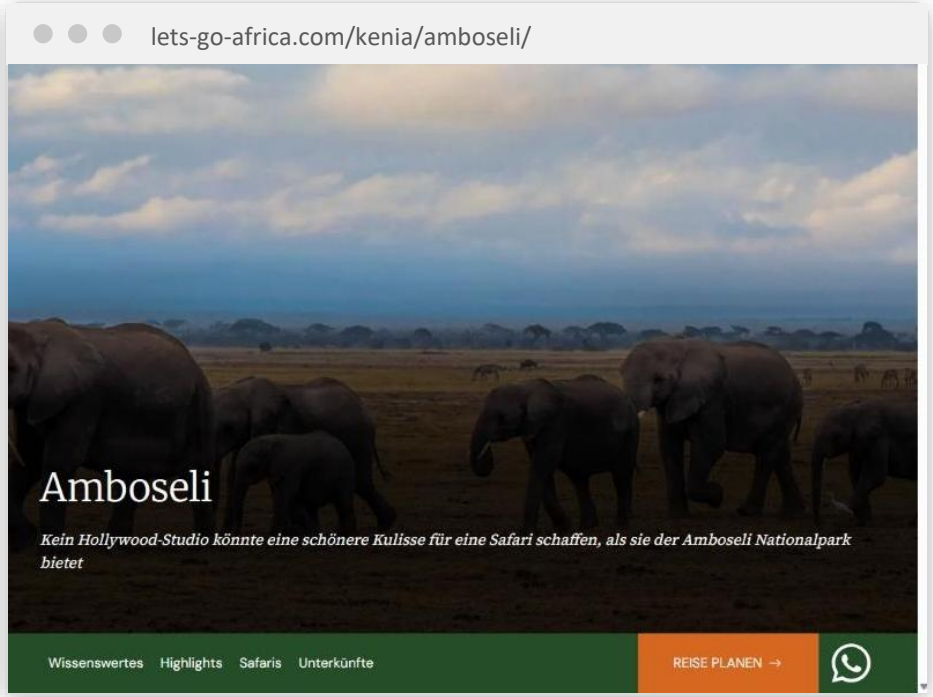
→

Ol Tukai Lodge
article

1. **Region = category**
   with its own fields: best travel time, size, founded, facts, flora & fauna

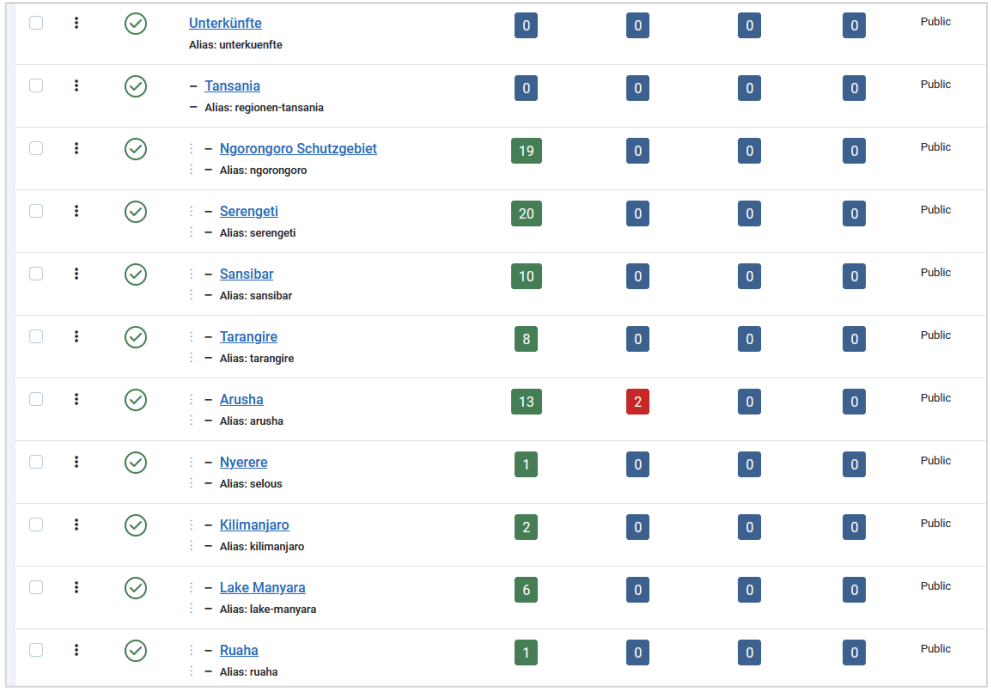
2. **Lodges = articles**
   in the region category - the region page lists them automatically

3. **Safaris in this region**
   derived backwards from the lodges in the itineraries - nobody assigns them

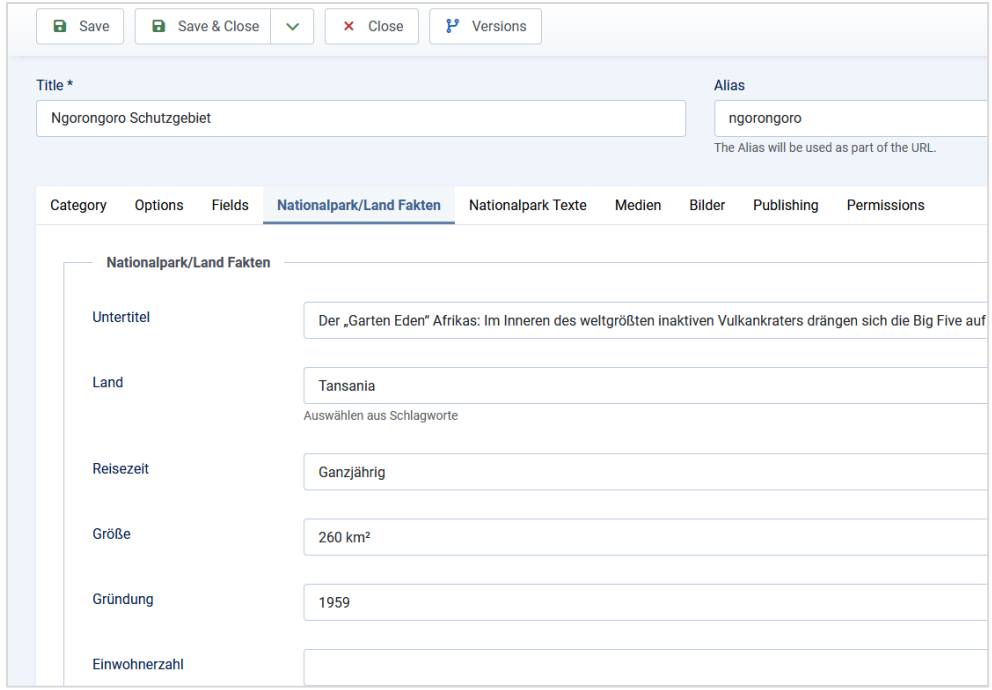


In the backend: countries & regions

Categories with their own custom fields



Category tree: Accommodations › country › region



Region category: custom fields for travel time, size, founded and facts

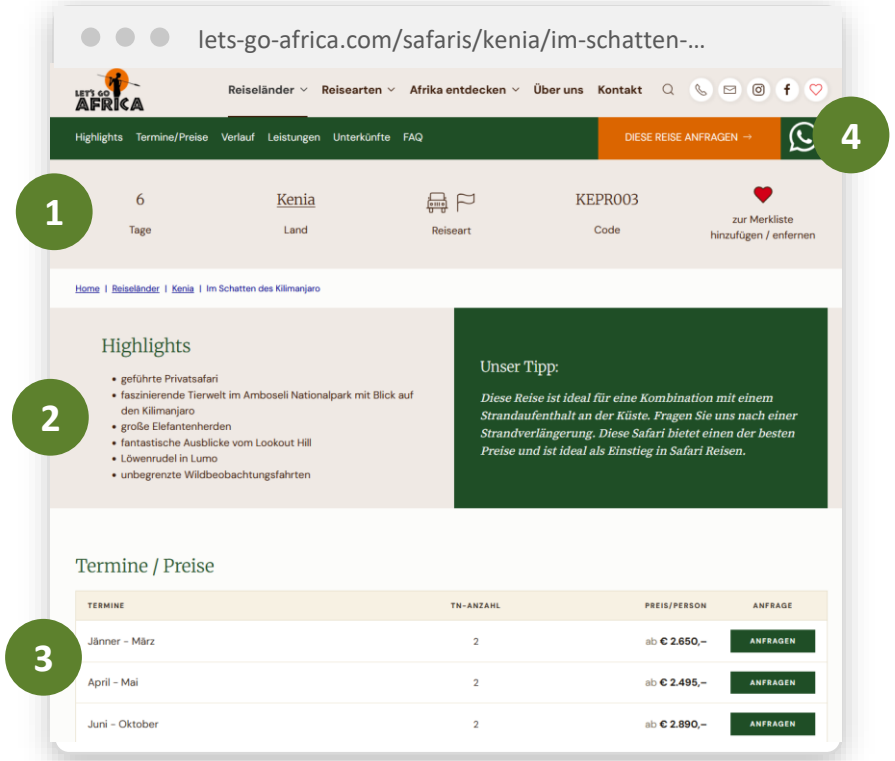
The safari detail page

Every block - how it is rendered and where its data comes from

Block on the page	Rendered by	Data from (Joomla)
Hero & intro	YOOtheme template	title · intro text · image
Facts bar	YOOtheme + own plugin	fields: days, code · category · tags
Highlights & tip	YOOtheme template	text fields
Dates & prices	**override: safaripreise**	subform 'safari-preise-termine'
Itinerary + map	YOOtheme template	subform 'safari-reiseverlauf'
Services included	YOOtheme template	text fields
Accommodations	**override: safariunterkuenfte**	field47 → lodge articles
FAQ	**override: faq**	FAQ category + tags
Request	RSForm!Pro, pre-filled	?safari=ID → article fields

Anatomy of a safari

One article - many fields



1 **Facts bar**

days and code: fields · country: category · travel type: tags

2 **Highlights & tip**

text fields - always in the same place, on every safari

3 **Dates & prices**

subform field, rendered by the module override 'safaripreise'

4 **Request button**

opens the RSForm!Pro request, pre-filled with this safari

The itinerary

Each day is a subform row - linked to real data

1. **Day by day**

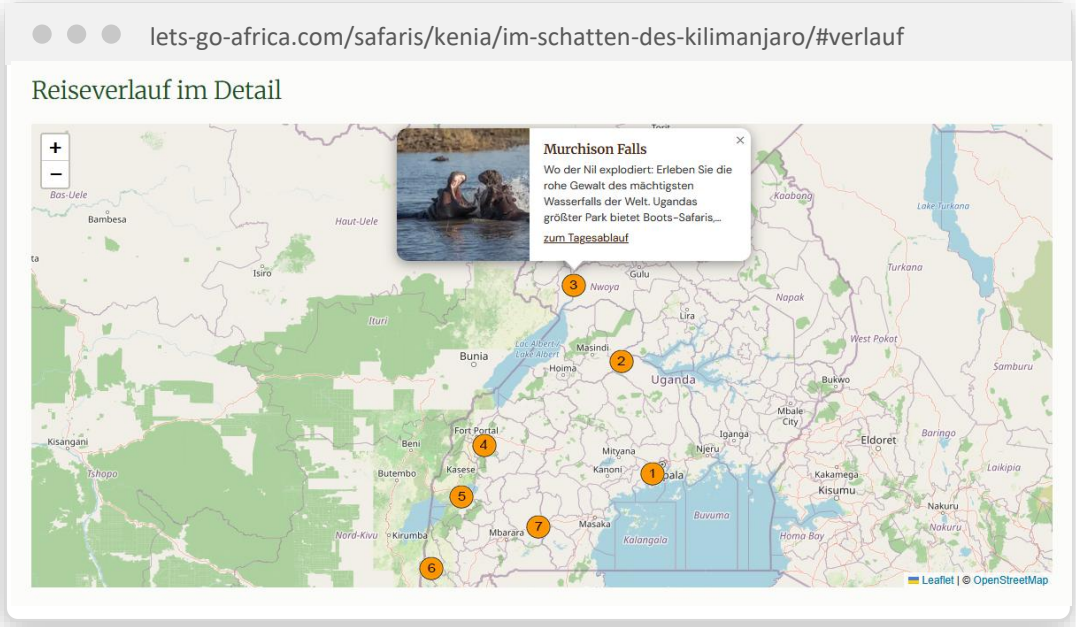
   title, text, image, meals - one subform row per day

2. **Linked accommodation**

   the lodge is an article; its region and country follow automatically

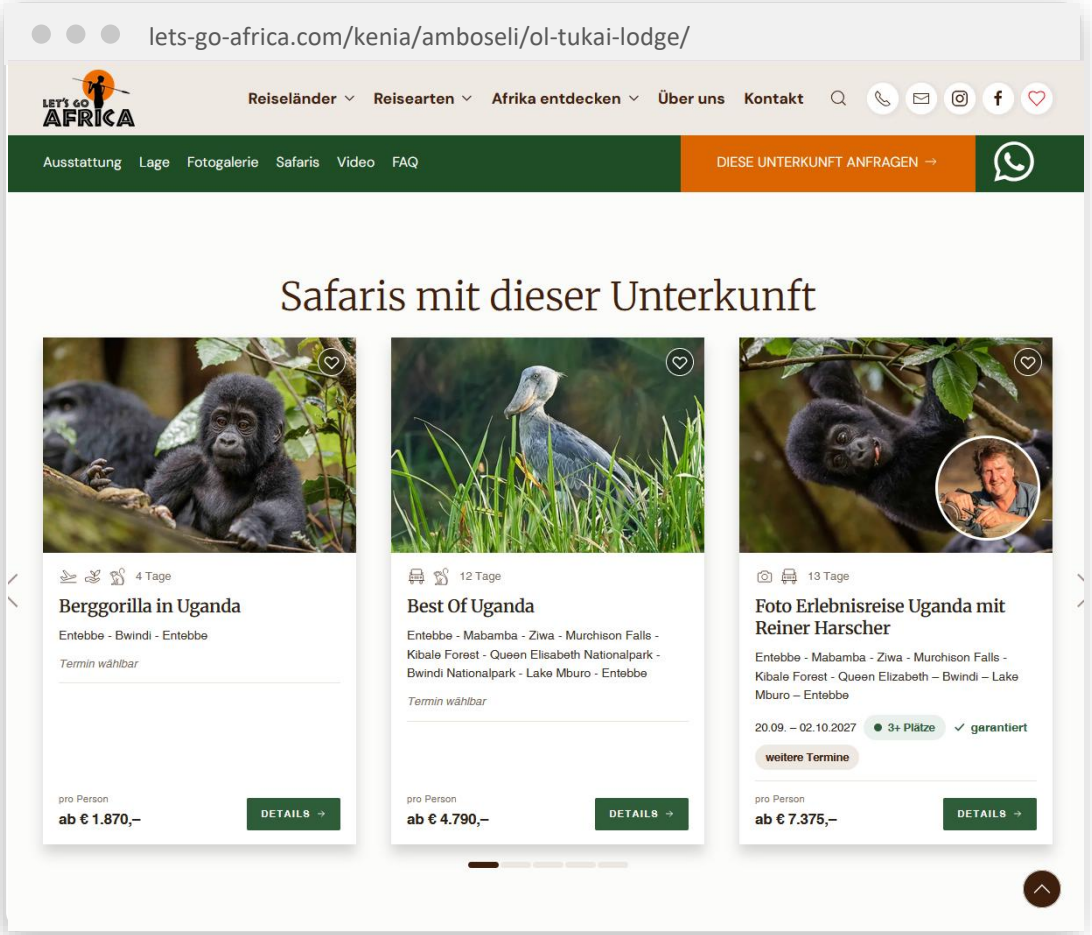
3. **Map from data**

   markers generated from the lodges' locations - Leaflet + OpenStreetMap, no Google



Enter once - show everywhere

One reference field, three directions - all module overrides



Safari → lodges

safariunterkuenfte

all lodges referenced in the itinerary

Lodge → safaris

unterkunftsafaris

every safari whose itinerary uses this lodge

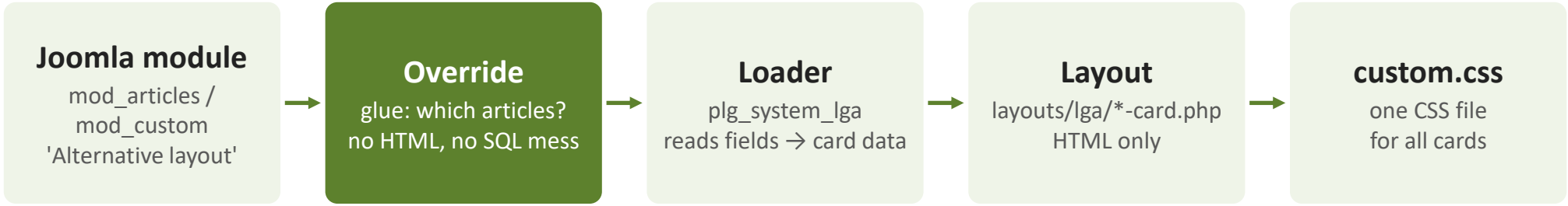
Region → safaris

regionsafaris

derived from the region's lodges

Overrides - the glue layer

One pattern for every module override



1 **One card design everywhere**

sliders, region pages, search results - safari, lodge and article cards look the same

2 **Data access in one place**

loaders know the field IDs - templates never touch the database

3 **Swap the template?**

rewrite layouts + CSS - modules, loaders and data stay

Module overrides: relations

All built on one reference field and tags

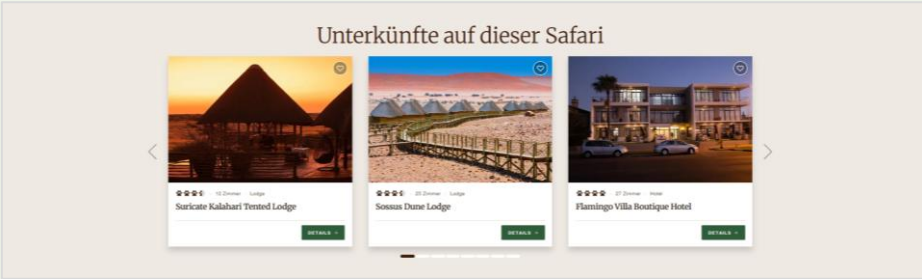
Safaris with this accommodation

unterkunftsafaris · reverse lookup via the itinerary



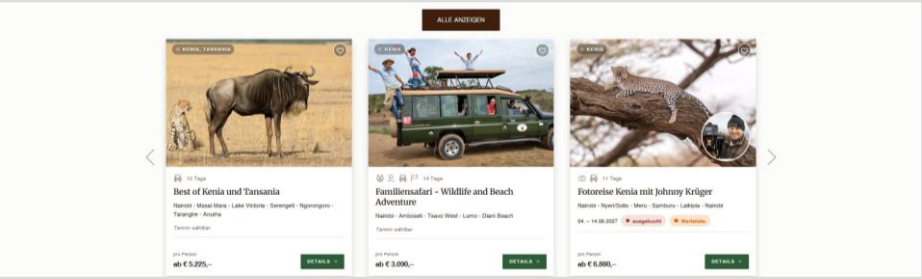
Accommodations on this safari

safariunterkuenfte · lodges from the itinerary



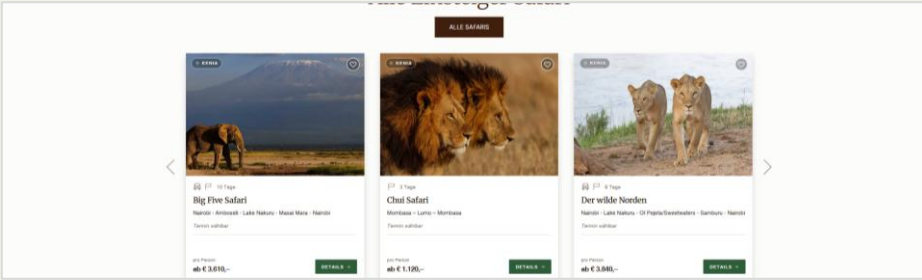
Safaris in this region

regionsafaris · derived from the region's lodges



Safaris of this travel type

reiseartsafaris · shared travel-type tags



Module overrides: lists, tables & FAQ

Core modules - new behaviour through alternative layouts

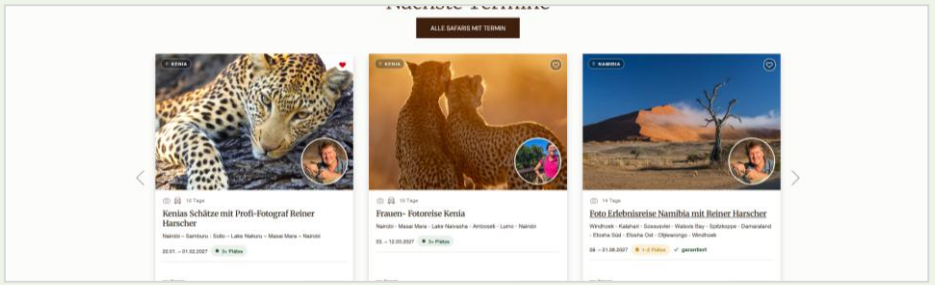
Dates & prices

safaripreise · table from subform, waitlist, request links



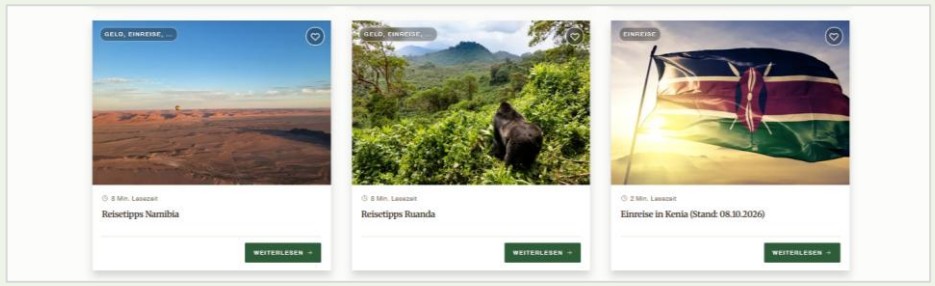
Next departures

terminsafaris · fixed dates, sorted by arrival field



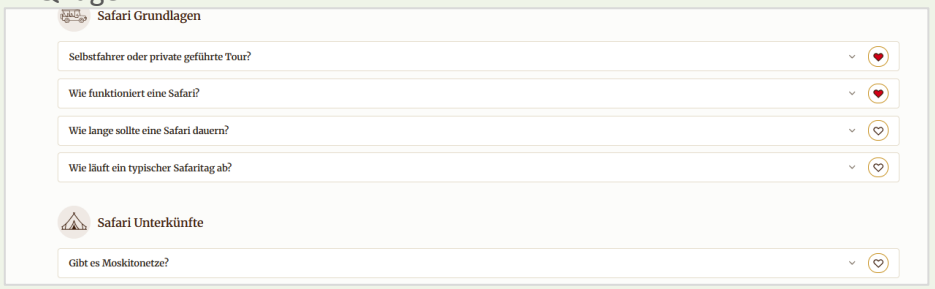
Card sliders

safaris · unterkuenfte · afrikaentdecken · same cards on every page



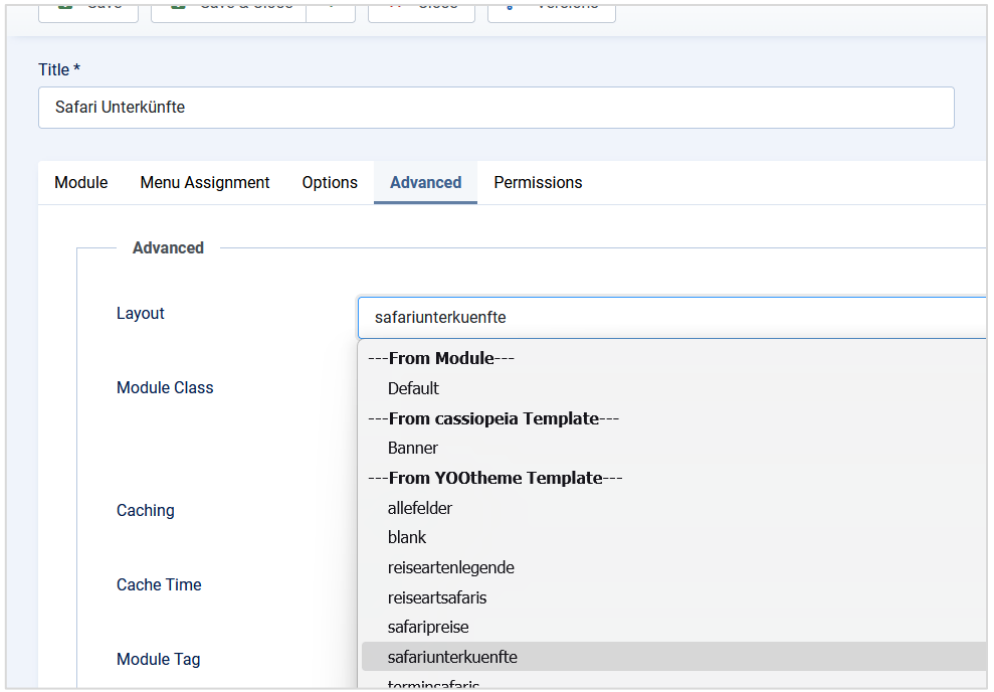
FAQ per page type

faq · faqland · faqreiseart · faqunterkunft · accordion + schema.org FAQPage

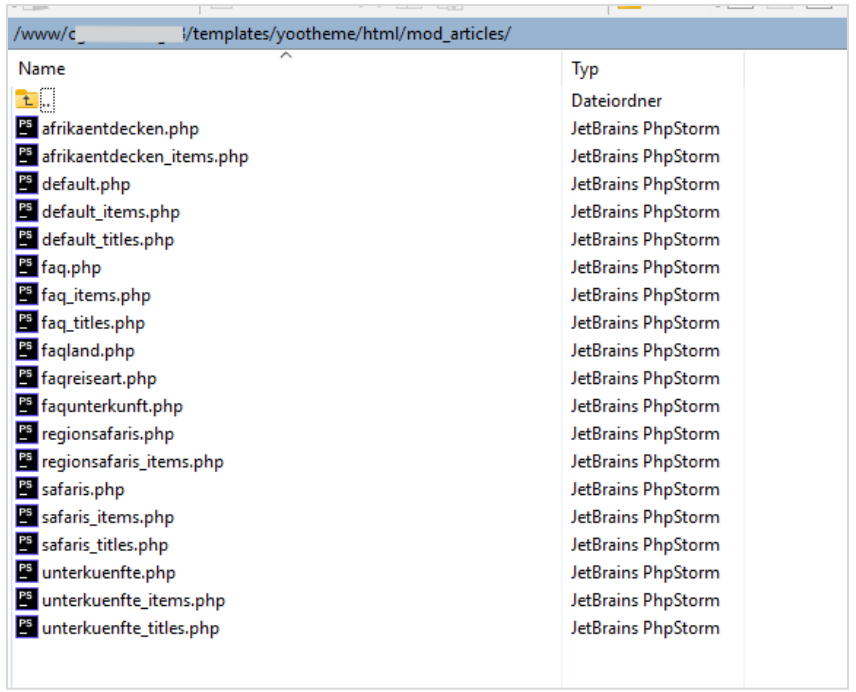


In the backend: activating an override

No core hacks - just an alternative layout



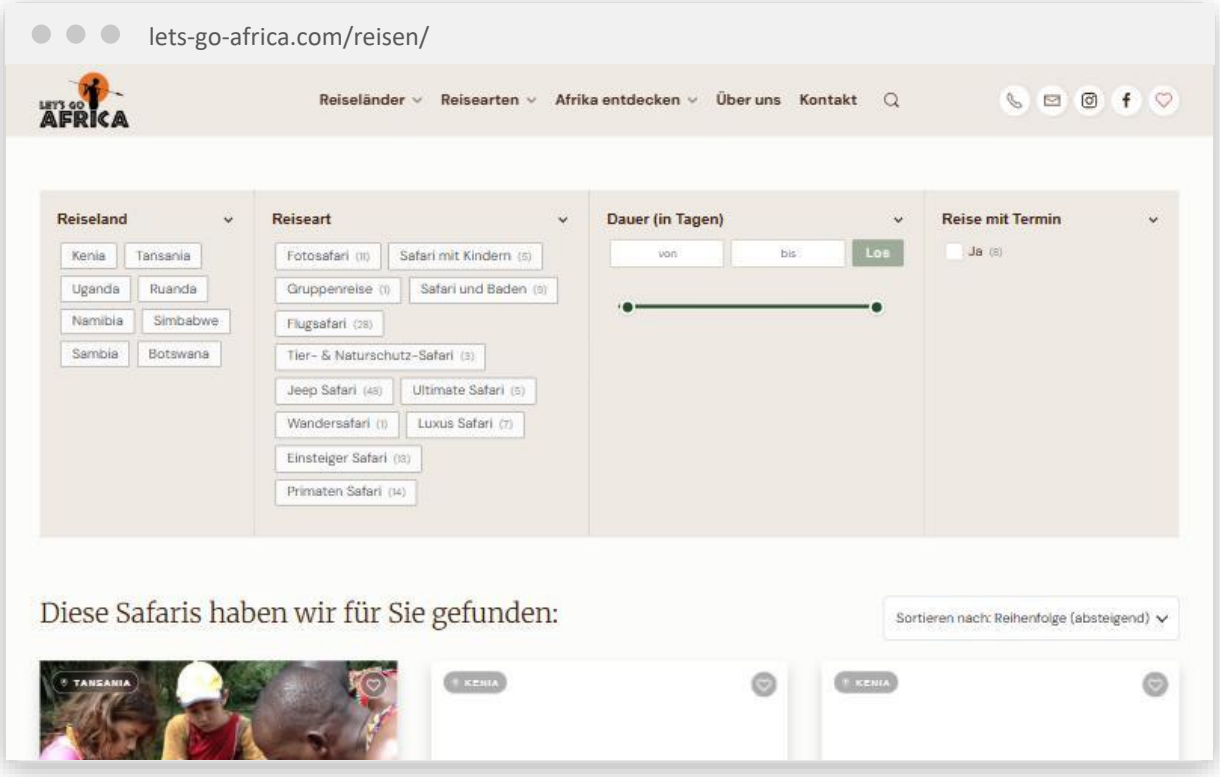
Module → Advanced → Layout: pick the alternative layout



Template: the override files in templates/yootheme/html

Find your safari

Filters only work when the data is structured



Country	category
Travel type	tags
Duration	number field, range
Fixed date	checkbox field

JFilters = one deliberate extension: it reads Joomla categories and custom fields - no own data storage.

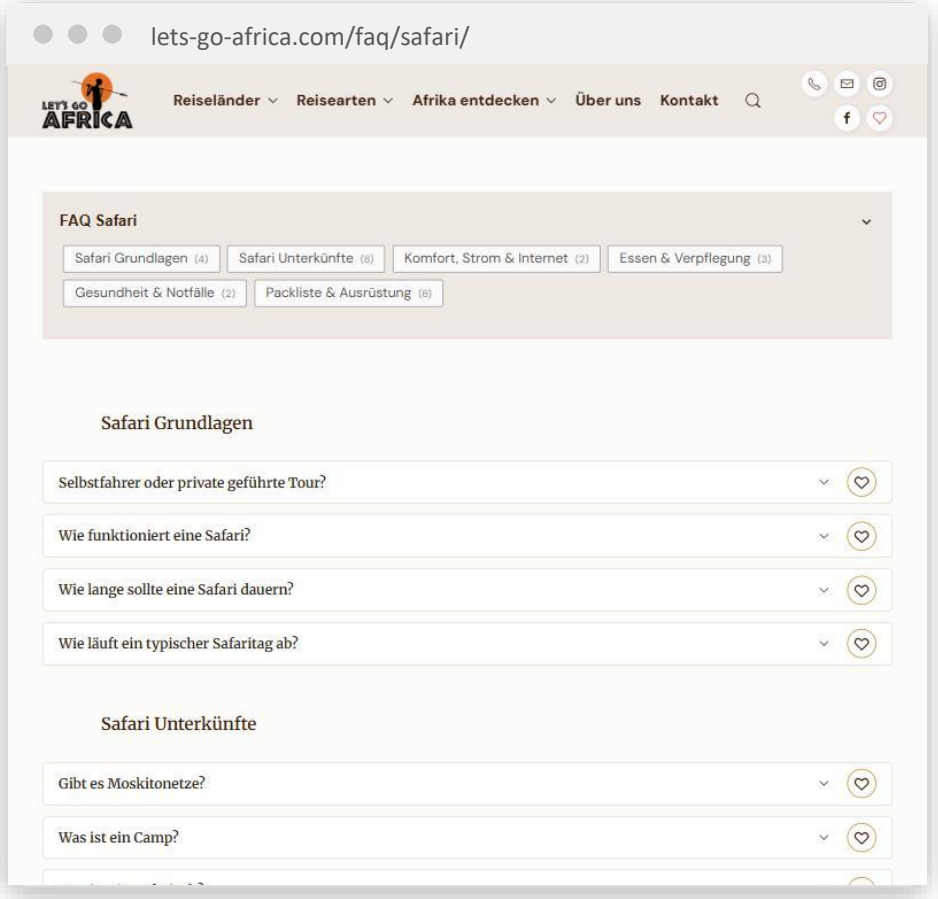
One results override - four content types

JFilters finds, our override decides how it looks

Category tree	Rendered as
Safaris	safari cards
Accommodations	lodge cards
Knowledge & blog	article cards
FAQ	accordion, grouped by tags

Switched per menu item by a page class (lga-cards) - without it, Joomla's standard result list.

Same card layouts as the module overrides.



Not everything is Joomla core.
But all **business data**
lives in **Joomla fields.**



YOOtheme Pro - as a design layer

How we use a page builder without losing the data

1. **Templates per view** — one builder template for all safari articles, one for lodges, one for countries - assigned by category
2. **Dynamic content** — elements are mapped to custom fields and subforms - no text typed into the builder
3. **Editors stay in Joomla** — they work in the normal article form, with field groups and notes
4. **Overrides where the builder ends** — cards, relations, price table, search results - plus accessibility fixes

The swap test

What happens if we replace a layer?

Replace …	What changes	What stays
YOOtheme Pro → another template	builder templates, lga layouts, custom.css	✓ all safaris, lodges, regions, prices, itineraries
JFilters → another filter component	one results override	✓ all safaris, lodges, regions, prices, itineraries
RSForm!Pro → another form extension	request form + pre-fill	✓ all safaris, lodges, regions, prices, itineraries
Regular Labs Articles Field → another field or SQL field	the lodge field in the itinerary subform	✓ all safaris, lodges, regions, prices, itineraries

The migration path

From K2 to Joomla core - in five steps

1 Model	2 Move	3 Rebuild	4 Redirect	5 Relaunch
map K2 extra fields to categories, custom fields, field groups and subforms	K2 items & categories → articles & categories	builder templates and overrides per content type	301 for every old URL, Search Console baseline	test, go live, monitor - and train the editors

Step 1 takes the longest - and saves the most. A clean data model makes every following step mechanical.



Core first - extensions by decision

Judged by maintainability, GDPR and long-term cost

Joomla core

holds the data

- Articles & categories
- Custom fields & subforms
- Tags
- Menus, URLs, access levels
- Mail templates, CSP, schema.org

Our code

small & documented

- 2 plugins (loaders, tag sync)
- ~35 overrides & layouts
- 1 CSS file
- Override overview doc

Extensions

one job each

- Regularlabs Articles Field
- YOOtheme Pro – design
- JFilters - search
- RSForm!Pro - requests
- AcyMailing - newsletter
- Pretty Reviews
- Kicktemp Wishlist (Faved)

Joomla 6 makes it even better

New core features that support 'Just Joomla'

6.0 **Number field**
with currency formatting - made for prices

6.0 **Note field**
guidance for editors right inside the form

6.0 **Versioning**
now includes custom field data

6.0 **Automated core updates**
less maintenance effort, more security

6.1 **Media custom fields**
audio, video and documents - not just images

6.1 **Proof-of-work captcha**
spam protection without a third-party service

Lessons learned

What we would do the same way again - and what hurt

1	**Model first, migrate second**	the field plan is the real project plan
2	**K2 extra fields ≠ custom fields 1:1**	types, groups and subforms need decisions
3	**Every URL matters**	redirect map + Search Console baseline before go-live
4	**Don't let the builder become the CMS**	if a text is in the builder, ask: why not a field?
5	**Write help into the edit form**	note fields and field descriptions train editors every day



When the
foundation is right,
Joomla grows with you.



Takeaways

Just Joomla!

1. **Separate data, structure and design - always**
2. Categories + custom fields are your data model = Content types!
3. Business data only in Joomla fields - nothing else owns it
4. Templates, builders, filters, forms: replaceable layers on top
5. Joomla 6: the principle stays, the tools get smarter

Thank you!

www.webgras.at/joomla

www.lets-go-africa.com

www.linkedin.com/in/sgramlinger

Questions? Let's talk.