Personal Website Next.js, Versel, Sanity.io - Step by Step

- Tools
  - Next.js
  - Sanity.io
  - TailwindCSS
  - TypeScript
  - Versel
- Vorher:
  - Node.js installieren

- Next.js starten:
  - npx create-next-app@latest
  - npm install @portabletext/react
  - npm install sanity
  - npm run dev

- Sanity.io starten:
  - Note: Anderes Verzeichnis als Next.js
  - npm create sanity@latest
  - npm run dev

- Sanity Basics zum Start:
  - sanity.config.ts: projetID und dataset sind wichtig
  - cors origins unter api bei sanity.io: localhost:3000 hinzufügen → next app kann alle daten des projekts sehen

▼ GROQ query language

unter VISION auf localhost:3333

▼ Routing in Next

nested routing, nested layout

folder korrespondiert mit der url

neuer folder foldername → neue page.tsx → via localhost/:3000/foldername erreichbar

globale layout.tsx und in den folders dann genested layout.tsx

▼ Embedding sanity studio into next app

aus sanity.config.ts: projectId: **********

Im folder für next app:

npm install sanity next-sanity

▼ sanity.config.ts erstellen und anpassen

```
import {defineConfig} from 'sanity';
import {structureTool} from 'sanity/structure';

const config = defineConfig({
    projectId: 'itontg9m',
    dataset: 'production',
    title: 'Personal Website',
    apiVersion: '2025-03-09',
    basePath: '/admin',
    plugins: [structureTool()],
})

export default config;
```

▼ next-sanity library

→ admin/[[...index]]/page.tsx]] → catch all route

▼ page.tsx configurieren:

```
"use client"; //nicht auf server rendern

import { NextStudio } from "next-sanity/studio";
import config from "@/sanity.config";


export default function AdminPage() {
  return <NextStudio config={config} />
}
```

über localhost/3000/admin nun sanity studio erreichbar

▼ sanity schema

▼ new Folder in Next App Folder → sanity/schemas + file project-schemas.ts

```
const project = {
  name: 'project',
  title: 'Projects',
  type: 'document',
  fields: [
    {
      name: 'name',
      title: 'Name',
      type: 'string',
    },
    {
      name: 'slug',
      title: 'Slug',
      type: 'slug',
    },
    {
      name: 'image',
      title: 'Image',
```

```
            type: 'image',
            options: {
               hotspot: true,
            },
            fields: [
               {
                  name: 'alt',
                  title: 'Alt',
                  type: 'string',
               },
            ],
         },

         {
            name: 'url',
            title: 'URL',
            type: 'url',
         },

         {
            name: 'content',
            title: 'Content',
            type: 'array',
            of: [{type: 'block'}],
         }
      ],
}

export default project;
```

▼ in sanity.config.ts importieren:

```
import {defineConfig} from 'sanity';
import {structureTool} from 'sanity/structure';
import project from './sanity/schemas/project-schema'; ***
const config = defineConfig({
   projectId: 'itontg9m',
   dataset: 'production',
```

```
    title: 'Personal Website',
    apiVersion: '2025-03-09',
    basePath: '/admin',
    plugins: [structureTool()],
    schema: {
       types: [project], ***
    },
})

export default config;
```

▼ Organisieren - alle schemas in ein file Barrel file

damit wir es nicht neu importieren müsse

alles über index.js im ordner sanity/schemas

▼ index.ts:

```
import project from './project-schema';

const schemas = [project];

export default schemas;
```

▼ neue sanity.config.ts

```
import {defineConfig} from 'sanity';
import {structureTool} from 'sanity/structure';
import schemas from './sanity/schemas';


const config = defineConfig({
    projectId: 'itontg9m',
    dataset: 'production',
    title: 'Personal Website',
    apiVersion: '2025-03-09',
```

```
    basePath: '/admin',
    plugins: [structureTool()],
    schema: {
      types: schemas,
    },
  })

  export default config;
```

▼ displaying data - groq query

neues projekte über das sanity studio projects +

Bsp 2 Test Projekte

▼ Neues File im Ordner sanity → sanity-utils.ts

enthält alle funktion die wir nutzen um daten zu greifen

nutzt groq

```
import { createClient } from "next-sanity";
import { groq } from "next-sanity";


export async function getProjects() {
  const client = createClient({
    projectId: 'itontg9m',
    dataset: 'production',
    apiVersion: '2025-03-09',
  });

  return client.fetch(
    groq`*[_type == "project"]{
      _id,
      _createdAt,
      name,
      "slug": slug.current,
      "image": image.asset→url,
      url,
```

```
          content
        }`
    )

}
```

▼ mapping over project - homepage anpassen

▼ page.tsx im app ordner bearbeiten

alles löschen und from scratch aufbauen

es holt sich so alle projects welche wir im sanity studio anlegen

```
import { getProjects } from "@/sanity/sanity-utils";

export default async function Home() {

  const projects = await getProjects();

  return (
    <div>
      {projects.map((project: any) => (
        <div key={project._id}>{project.name}</div>
      ))}
    </div>
  );
}
```

▼ setting up typescript types

neuer folder → types

▼ file → Project.ts

```
import { PortableTextBlock } from "next-sanity";
```

```
export type Project = {
    _id: string;
    _createdAt: Date;
    name: string;
    slug: string;
    image: string;
    url: string;
    content: PortableTextBlock[];
}
```

▼ sanity-utily.ts anpassen - hier alle types definieren

```
import { Project } from "@/types/Project";
import { createClient } from "next-sanity";
import { groq } from "next-sanity";


export async function getProjects(): Promise<Project[]> { ***
    const client = createClient({
        projectId: 'itontg9m',
        dataset: 'production',
        apiVersion: '2025-03-09',
    });

    return client.fetch(
        groq`*[_type == "project"]{
            _id,
            _createdAt,
            name,
            "slug": slug.current,
            "image": image.asset->url,
            url,
            content
        }`
    )

}
```

wenn wir funktion getProjects nutzen wissen wir nun wir nun den type

▼ tailwindcss hinzufügen

▼ install

npm install tailwindcss @tailwindcss/postcss postcss

▼ psotcss.config.mjs hinzufügen und anpassen

```
const config = {
plugins: {
"@tailwindcss/postcss": {},
},
};
export default config;
```

@import "tailwindcss"; zur global.css hinzufügen

▼ Nutzung via className = “ “

```
<h1 className="text-3xl font-bold underline">
Hello world!
</h1>
```

▼ Code beispiel

```
import { getProjects } from "@/sanity/sanity-utils";
import Image from "next/image";

export default async function Home() {

  const projects = await getProjects();

  return (
    <div className="max-w-5xl mx-auto py-20">

      <h1 className="text-4xl font-extrabold">
        Hi, ich bin {""}
```

```
  <span className="bg-gradient-to-r from-orange-400 via-red-5
    Franz
  </span>
  !
</h1>

<p className="mt-3 text-gray-600">
  Alle Projekte an einem Ort
</p>

<h2 className="mt-24 font-bold text-gray-700 text-3xl">
  Meine Projekte
</h2>



<div className="mt-2.5 grid grid-cols-1 md:grid-cols-2 lg:grid-cc
    {projects.map((project) => (

      <div key={project._id} className="border-2 border-gray-500

        {project.image && (
          <Image
            src={project.image}
            alt={project.name}
            width={750}
            height={300}
            className="object-cover rounded-lg border border-gray-
          />
        )}

        <div className="font-bold bg-gradient-to-r from-orange-4

          {project.name}

        </div>

      </div>
```

```
          ))}
      </div>



    </div>
  );
}
```

▼ mehr im studio - images in next

github project hinzufügen - link und image

▼ Error: Invalid src prop fixen

in next.config.ts ergänzen

```
import type { NextConfig } from "next";

const nextConfig: NextConfig = {
  images: {
    remotePatterns: [
      {
        protocol: 'https',
        hostname: 'cdn.sanity.io',
        port: '',
        search: '',
      }
    ]
  }
};

export default nextConfig;
```

▼ Linkbar machen

```
<Link
   key={project._id}
```

```
  href={`/projects/${project.slug}`}
  //direkter link zur url href={project.url}
    target="_blank"
    className="border-2 border-gray-500 rounded-lg p-1 hover:scale-
>
</Link>
```

▼ individuelle project pages

Neuer Folder projects/[project] - neues file page.tsx

▼ page.tsx

```
import { getProject } from "@/sanity/sanity-utils";

type Props = {
    params: {
        project: string;
    };
};

export default async function Project({ params }: Props) {

    const slug = params.project;

    const project = await getProject(slug);



    return <div>{project.name}</div>;
}
```

▼ anpassungen in sanity-utils.ts

```
export async function getProject(slug: string): Promise<Project> {

    const client = createClient({
        projectId: 'itontg9m',
```

```
    dataset: 'production',
    apiVersion: '2025-03-09',
  });

  return client.fetch(
    groq`*[_type == "project" && slug.current == $slug][0]{
      _id,
      _createdAt,
      name,
      "slug": slug.current,
      "image": image.asset->url,
      url,
      content
    }`,
    { slug }
  )

}
```

▼ auslagern von create client

▼ neuer folder unter sanity → config + neues file →client-config-ts

```
const config = {
  projectId: 'itontg9m',
  dataset: 'production',
  apiVersion: '2025-03-09',
}



export default config;
```

▼ sanity-utils.ts anpassen

```
import { Project } from "@/types/Project";
import { createClient } from "next-sanity";
import { groq } from "next-sanity";
import clientConfig from "./config/client-config";
```

```
export async function getProjects(): Promise<Project[]> {

   return createClient(clientConfig).fetch( ***
      groq`*[_type == "project"]{
         _id,
         _createdAt,
         name,
         "slug": slug.current,
         "image": image.asset->url,
         url,
         content
      }`
   )

}

export async function getProject(slug: string): Promise<Project> {


   return createClient(clientConfig).fetch( ***
      groq`*[_type == "project" && slug.current == $slug][0]{
         _id,
         _createdAt,
         name,
         "slug": slug.current,
         "image": image.asset->url,
         url,
         content
      }`,
      { slug }
   )

}
```

▼ styling der project pages

▼ arbeit in projects/[project] → page.tsx

▼ alles im header

```
import { getProject } from "@/sanity/sanity-utils";

type Props = {
   params: {
      project: string;
   };
};

export default async function Project({ params }: Props) {

   const slug = params.project;

   const project = await getProject(slug);

   return (
      <div className="max-w-3xl mx-auto py-20">
         <header
            className="flex items-center justify-between"
         >
            <h1
               className="bg-gradient-to-r from-orange-400 via-re
            >
               {project.name}
            </h1>

            <a
               href={project.url}
               title="View Project"
               rel="noopener noreferrer" target="_blank"
               className="bg-gray-100 text-gray-500 px-3 py-2 rou
            >
               View Project
            </a>
         </header>
```

```
        {/* Content */}

        {/* Image */}
      </div>
    );

}
```

▼ content

```
<div>
          {project.content}
        </div>
```

erzeugt error

▼ Lösung:

npm install @portabletext/react

```
import { PortableText } from "@portabletext/react";
<div>
            <PortableText value={project.content} />
          </div>
```

▼ image

```
{/* Image */}
        <Image
          src={project.image}
          alt={project.name}
          width={1920}
          height={1080}
          className="mt-10 border-2 border-gray-700 rounded-x
        />
```

▼ navigationsbar (HOME)

▼ im globalen layout.tsx file

alles was hier definiert ist gilt für alle pages , der individuelle inhalt wird in main {children} geholt

```
import type { Metadata } from "next";
import { Geist, Geist_Mono } from "next/font/google";
import "./globals.css";
import Link from "next/link";

const geistSans = Geist({
  variable: "--font-geist-sans",
  subsets: ["latin"],
});

const geistMono = Geist_Mono({
  variable: "--font-geist-mono",
  subsets: ["latin"],
});

export const metadata: Metadata = {
  title: "Meine Homepage",
  description: "Generated by Next + Sanity",
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="en">
      <body
        className="max-w-3xl mx-auto py-10"
      >
        <header>
          <Link
            href="/"
            className="bg-gradient-to-r from-orange-400 via-red-500 to
```

```
          >
            HOME
          </Link>
        </header>

        <main className="py-20">
          {children}
        </main>


      </body>
    </html>
  );
}
```

applied auch auf die admin page → SCHLECHT

bedarf separates layout

▼ Next organizational folders (folder)

behebt unser problem mit der navbar

▼ (site) folder und (studio) folder zur Organisation

→ haben nichts mit routing zu tun → nur orga

globale layout.tsx in beide verschieben und die datei für studio bearbeiten

alle anderen folder/routen auch jeweils passend verschieben

bsp admin in den studio folder, project in den site ordner

▼ Pages schema ergänzen

▼ page-schema.ts unter schemas folder

```
const page = {
   name: "page",
   title: "Pages",
   type: "document",
   fields: [
      {
         name: "title",
         title: "Title",
         type: "string",

      },
      {
         name: "slug",
         title: "Slug",
         type: "slug",
         options: {
            source: "title",
            maxLength: 96,
         },
      },
      {
         name: "content",
         title: "Content",
         type: "array",
         of: [{type: "block"}],
      },
   ],
};

export default page;
```

in index.ts importieren

im sanity studio pages projekte erstellen

▼ unter sanity-utils.ts getpages und getpage ergänzen

```
export async function getPages(): Promise<Page[]> {

    return createClient(clientConfig).fetch(
        groq`*[_type == "page"]{
            _id,
            _createdAt,
            title,
            "slug": slug.current,
        }`
    )

}

export async function getPage(slug: string): Promise<Page> {

    return createClient(clientConfig).fetch(
        groq`*[_type == "page" && slug.current == $slug][0]{
            _id,
            _createdAt,
            title,
            "slug": slug.current,
            content
        }`,
        { slug }
    )

}
```

▼ im ordner types ein file Page.ts anlegen

```
import { PortableTextBlock } from "next-sanity";

export type Page = {
    _id: string;
    _createdAt: Date;
    title: string;
    slug: string;
    content: PortableTextBlock;
```

```
}
```

▼ (site)layout.tsx anpassen

```
import type { Metadata } from "next";
import { Geist, Geist_Mono } from "next/font/google";
import "../globals.css";
import Link from "next/link";
import { getPages } from "@/sanity/sanity-utils";

const geistSans = Geist({
  variable: "--font-geist-sans",
  subsets: ["latin"],
});

const geistMono = Geist_Mono({
  variable: "--font-geist-mono",
  subsets: ["latin"],
});

export const metadata: Metadata = {
  title: "Meine Homepage",
  description: "Generated by Next + Sanity",
};

export default async function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {

  const pages = await getPages();


  return (
    <html lang="en">
      <body
```

```
        className="max-w-3xl mx-auto py-10"
      >
        <header className="flex items-center justify-between">
          <Link
            href="/"
            className="bg-gradient-to-r from-orange-400 via-red-500 to
          >
            HOME
          </Link>

          <div className="flex items-center gap-6 text-sm text-gray-60
            {pages.map((page) => (
              <Link
                key={page._id}
                href={`/${page.slug}`}
                className="text-gray-300 hover:text-white transition"
              >
                {page.title}
              </Link>
            ))}
          </div>
        </header>

        <main className="py-20">
          {children}
        </main>


      </body>
    </html>
  );
}
```

▼ neues file unter (site) → [slug]/page.tsx

dynamische route

▼ Github rep erstellen und pushen

leeres rep auf github erstellen

nur den nextjs folder nutzen nicht die das sanity setup

▼ CMD

git init

git remote add origin https://github.com/HandlessReaper/PersonalWebsite.git
git add .

git commit -m "Initial commit"

git branch -M main

git push -u origin main

via versel deployen