Carl von Ossietzky
Universität
Oldenburg

Erweiterung des Vue-Techstacks

Bessere Schnittstelle zwischen Back- und Frontend

Rami Jasim, 26.02.2026

Carl von Ossietzky
Universität
Oldenburg

Motivation

- Plugin-Entwicklung an der UOL mit Vue
- Grundlegende Modernisierung:
  - Webpack -> Vite
  - Vuex -> Pinia
- Plugin-Vorlage für moderne Vue-Entwicklung
- Fachliche Entwicklung kann beginnen

Motivation II

Nicht so einfach wie gedacht

- Ressourcen Fetchen mühselig
    - Unmengen an Boilerplate für JSON-API Routen
    - Kaum Basisklassen fürs Frontend (API-Calls, State-Stores etc.)
- Stud.IP nicht auf Vue-Entwicklung ausgelegt - Grüne Wiese im Frontend

-> Zielsetzung: Basisstruktur schaffen

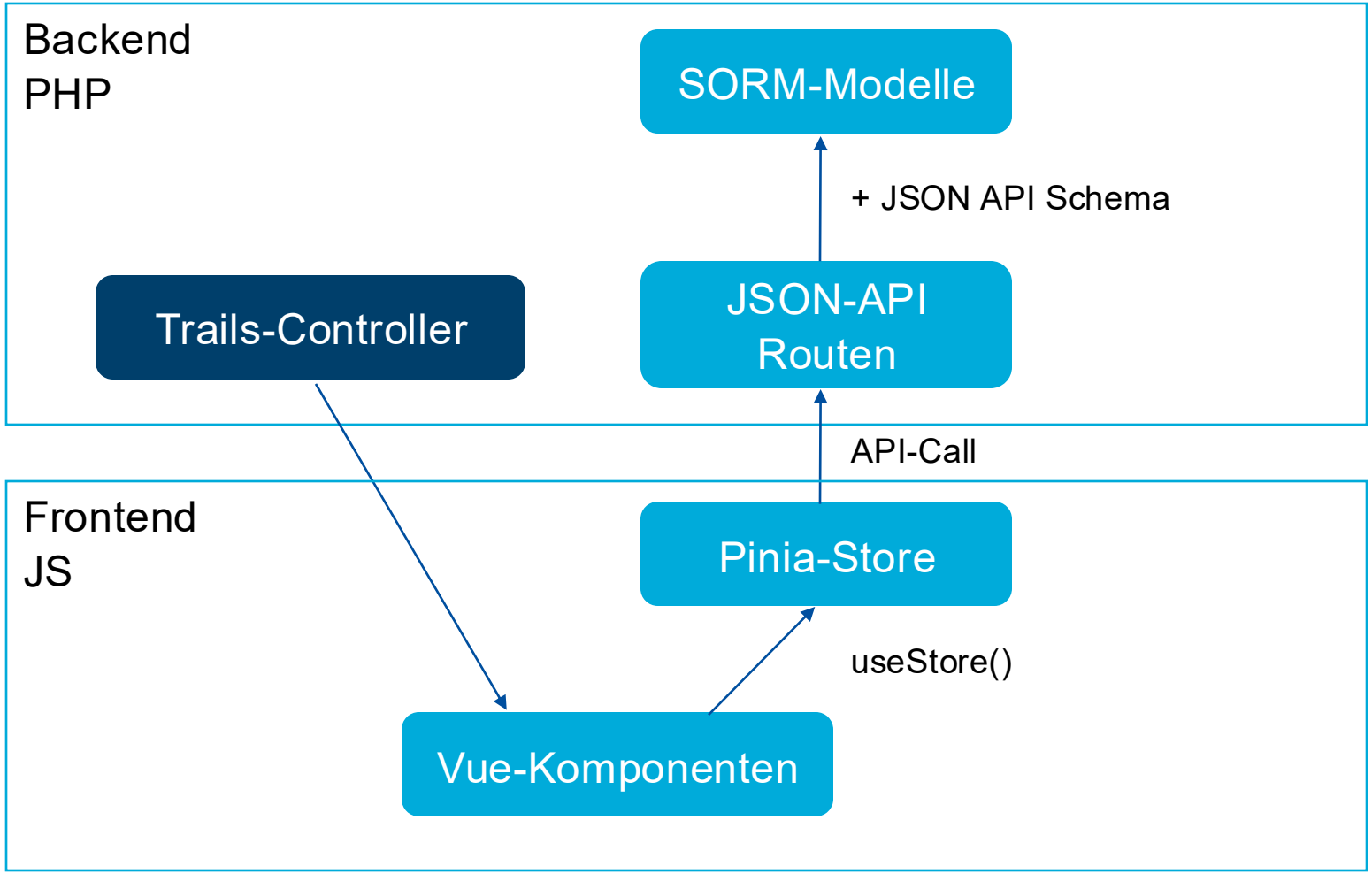
Zielsetzung

Vue Entwicklung **nachvollziehbarer**,
effizienter und **strukturierter** gestalten

- Standard-Funktionalitäten einrichten und abkapseln
- Mehr Hilfestellung für Entwickler im Frontend
- Einheitlicher Technologie-Stack und Arbeitsweise
- Bessere Separation zwischen Front- und Backend

Carl von Ossietzky
Universität
Oldenburg

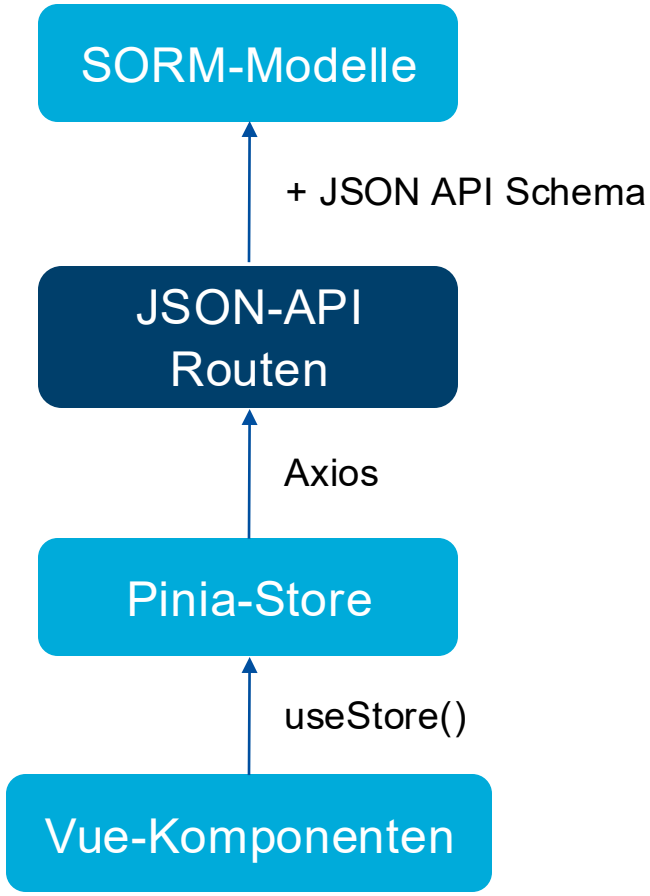
Back- und Frontend



Carl von Ossietzky
Universität
Oldenburg

Agenda

- Backend Schnittstelle
  - **JSON-API Routen & Schema**
- Frontend Schnittstelle
  - Frontend Model
  - Client State vs. Server State



Carl von Ossietzky Universität Oldenburg

JSON-API Routen und Schema

Daten aus dem Backend fetchen

- 1 Schema
  - TYPE & getId() & getAttributes()
  - getRelationships()

```
class Container extends SchemaProvider
{
    const TYPE = 'courseware-containers';

    5 usages
    const REL_BLOCKS = 'blocks';
    const REL_OWNER = 'owner';
    1 usage
    const REL_EDITOR = 'editor';
    2 usages
    const REL_EDITBLOCKER = 'edit-blocker';
    const REL_STRUCTURAL_ELEMENT = 'structural-element';

    /**
     * {@inheritdoc}
     */
    Jan-Hendrik Willms +1
    public function getId($resource): ?string
    {
        return $resource->id;
    }
```

Carl von Ossietzky
Universität
Oldenburg

JSON-API Routen und Schema

Daten aus dem Backend fetchen

- 1 Schema
  - TYPE & getId() & getAttributes()
  - getRelationships()

```php
public function getAttributes($resource, ContextInterface $context)
{
    return [
        'position' => (int) $resource['position'],
        'site' => (int) $resource['site'],
        'container-type' => (string) $resource['container_type'],
        'title' => (string) $resource->type->getTitle(),
        'width' => (string) $resource->type->getContainerWidth(),
        'visible' => (bool) $resource['visible'],
        'payload' => $resource['payload']->getIterator(),
        'mkdate' => date( format: 'c', $resource['mkdate']),
        'chdate' => date( format: 'c', $resource['chdate']),
    ];
}
```

Carl von Ossietzky Universität Oldenburg

JSON-API Routen und Schema

Daten aus dem Backend fetchen

- 1 Schema
  - TYPE & getId() & getAttributes()
  - getRelationships()

```php
public function getRelationships($resource, ContextInterface $context): it
{
    $relationships = [];

    $relationships[self::REL_OWNER] = $resource->owner
        ? [
            self::RELATIONSHIP_LINKS => [
                Link::RELATED => $this->createLinkToResource($resource->ow
            ],
            self::RELATIONSHIP_DATA => $resource->owner,
        ]
        : [self::RELATIONSHIP_DATA => $resource->owner];

    $relationships[self::REL_EDITOR] = $resource->editor
        ? [
            self::RELATIONSHIP_LINKS => [
                Link::RELATED => $this->createLinkToResource($resource->ed
            ],
            self::RELATIONSHIP_DATA => $resource->editor,
        ]
        : [self::RELATIONSHIP_DATA => null];
```

Carl von Ossietzky
Universität
Oldenburg

JSON-API Routen und Schema

Daten aus dem Backend fetchen

- 1 Schema
  - TYPE & getId() & getAttributes()
  - getRelationships()
- Routen (Index, Show, Create, Update, Delete)
  - 1 Klasse pro Route

```php
class ContainersIndex extends JsonApiController
{
    protected $allowedPagingParameters = ['offset', 'limit'];

    protected $allowedIncludePaths = [
        'blocks',
        'blocks.edit-blocker',
        'blocks.editor',
        'blocks.owner',
        'blocks.user-data-field',
        'blocks.user-progress',
        'editor',
        'edit-blocker',
        'owner',
        'structural-element',
];
```

Carl von Ossietzky
Universität
Oldenburg

JSON-API Routen und Schema

Daten aus dem Backend fetchen

- 1 Schema
  - TYPE & getId() & getAttributes()
  - getRelationships()
- Routen (Index, Show, Create, Update, Delete)
  - 1 Klasse pro Route

```php
public function __invoke(Request $request, Response $response, $args)
{
    if (!$resource = StructuralElement::find($args['id'])) {
        throw new RecordNotFoundException();
    }

    if (!Authority::canIndexContainers($this->getUser($request), $resource)) {
        throw new AuthorizationFailedException();
    }

    list($offset, $limit) = $this->getOffsetAndLimit();

    return $this->getPaginatedContentResponse(
        $resource->containers->limit($offset, $limit),
        count($resource->containers)
    );
}
```

JSON-API Routen und Schema

Daten aus dem Backend fetchen

- 1 Schema
  - TYPE & getId() & getAttributes()
  - getRelationships()
- Routen (Index, Show, Create, Update, Delete)
  - 1 Klasse pro Route
- Mapping eintragen
  (SORM -> Schema)
  (Pfad -> Route)

SORM CRUD

- Viele Informationen stecken schon im Modell

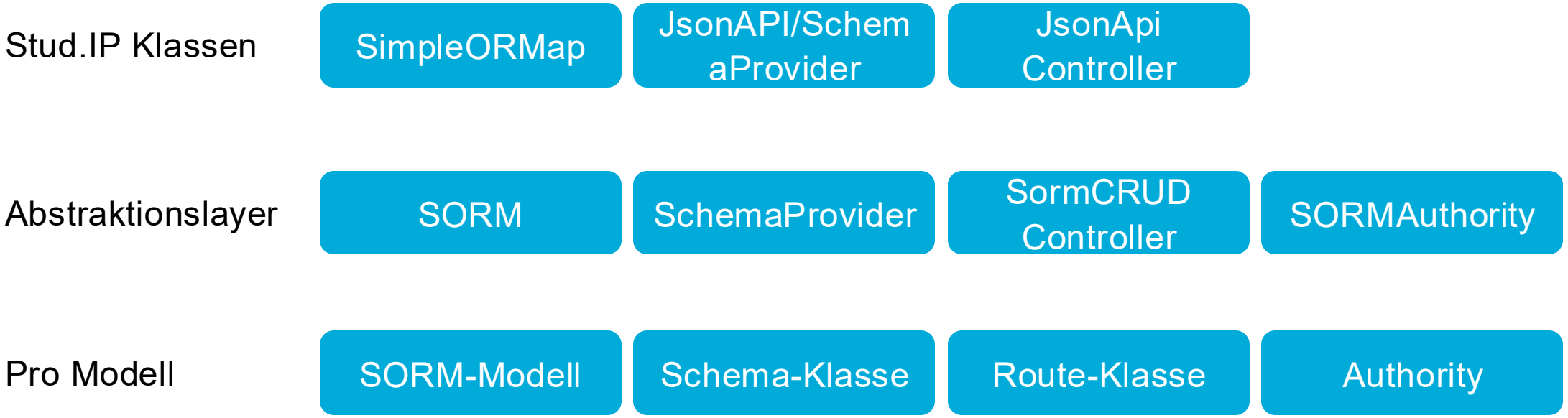
-> Stärkere Verknüpfung von Model und Schema/Routen

Stud.IP Klassen	SimpleORMap	JsonAPI/SchemaProvider	JsonApiController
Pro Modell	SORM-Modell	Schema-Klasse	Route-Klasse

SORM CRUD

- Viele Informationen stecken schon im Modell

-> Stärkere Verknüpfung von Model und Schema/Routen



Carl von Ossietzky
Universität
Oldenburg

SORM CRUD

Model-Klassen

1 Route-Klasse pro Model

- Verknüpfung zum Model und Authority
- Konfiguration fürs GET (data, filters)

1 Schema-Klasse pro Model

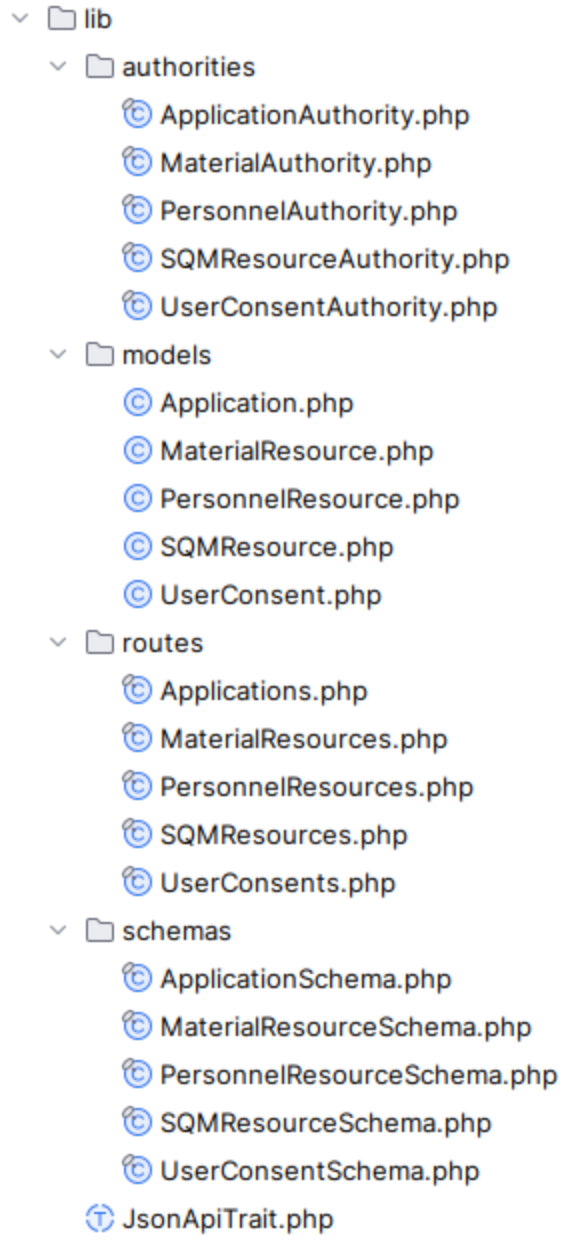
- Konstanten für Relationen & Pfad

1 Authority pro Model

- Zugriffsrechte für access, create etc.

Mapping-Klasse

- SORM -> Schema
- Pfad -> Route-Klasse



SORM CRUD

Vorteile

- Abkapselung von vielem JsonAPI spezifischem Code
- 1 Zentrale Route Klasse
- Nutzung der Model-Klasse
- Grundlegend mächtigere Routen
- Weniger Klassen, weniger Boilerplate, weniger doppelter Code

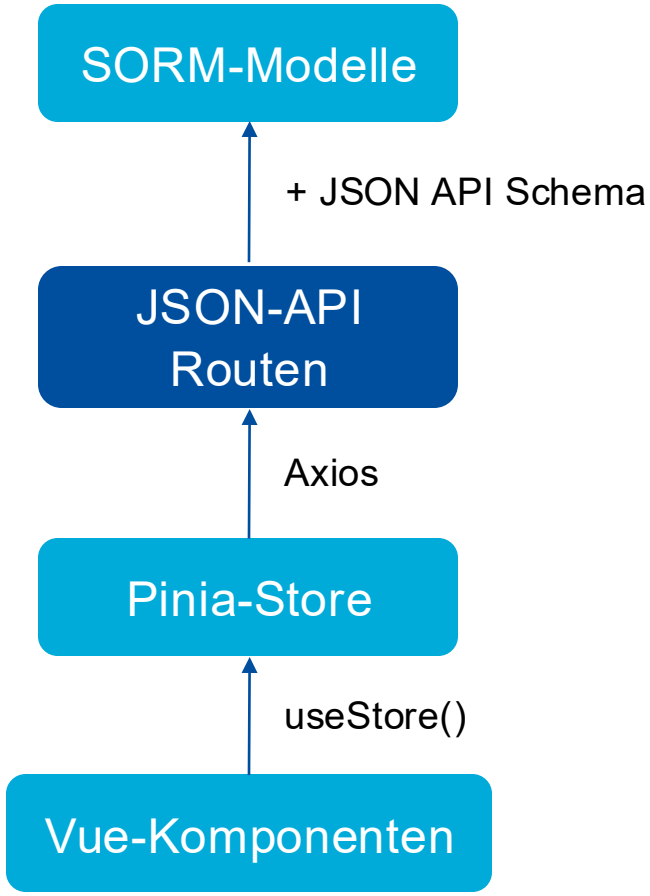
Nachteile

- Bisher unklar, wie gut nicht-Standardfälle funktionieren (bspw. andere Routen)

Carl von Ossietzky
Universität
Oldenburg

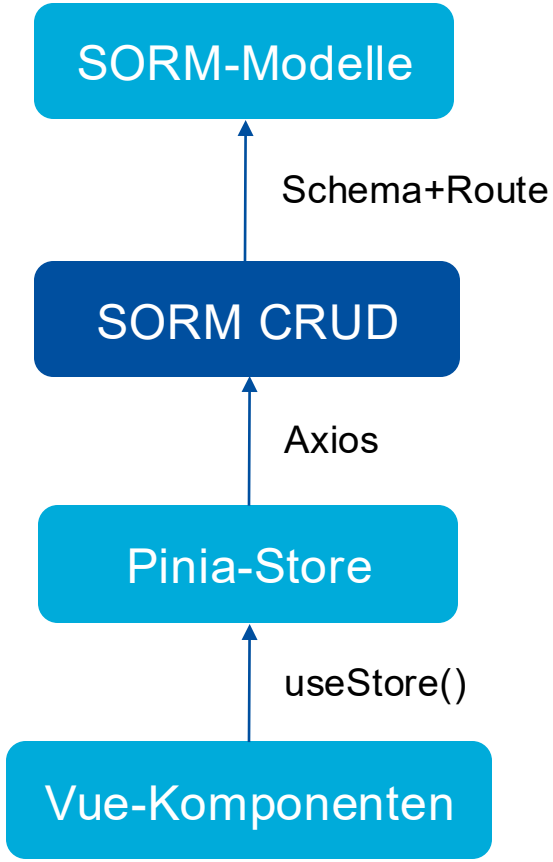
Agenda

- Backend Schnittstelle
  - **JSON-API Routen & Schema**
- Frontend Schnittstelle
  - Frontend Model
  - Client State vs. Server State



Agenda

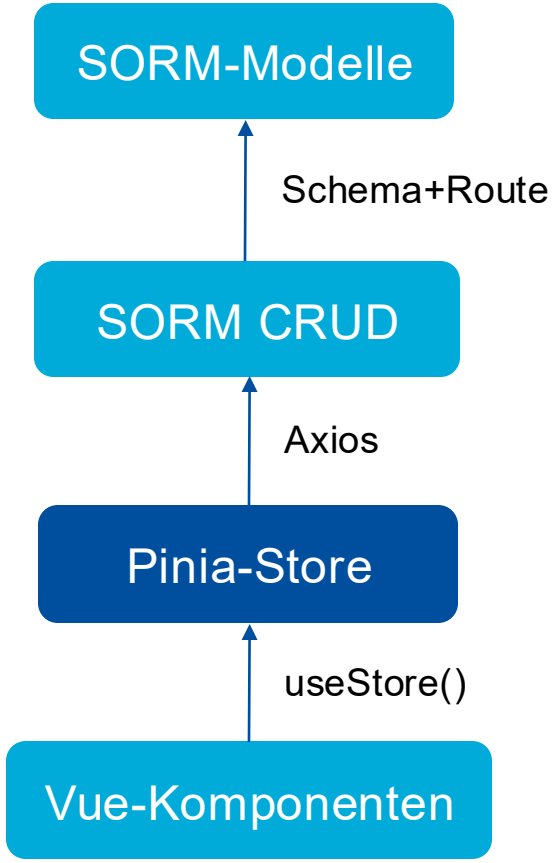
- Backend Schnittstelle
  - **JSON-API Routen & Schema**
- Frontend Schnittstelle
  - Frontend Model
  - Client State vs. Server State



Carl von Ossietzky
Universität
Oldenburg

Agenda

- Backend Schnittstelle
  - JSON-API Routen & Schema
- Frontend Schnittstelle
  - **Frontend Model**
  - Client State vs. Server State



Carl von Ossietzky
Universität
Oldenburg

Frontend Model

- SORM im PHP-Backend mit typehints
- Keine ts-Types – Unbekannte Datentypen im Frontend

Einführung von ts-Klassen für Models

- Datentypen sind bekannt
  + Typehints für Attribute und Methoden
- Klassen-Methoden statt statischen
- SormCRUD vereinheitlich Route&Model

```
export class Semester {

  public id: string = ''
  public title: string = ''
  public token: string = ''
  public start: string = ''
  public 'start-of-lectures': string = ''
  public 'end-of-lectures': string = ''
  public end: string = ''
  public 'is-current': string = ''
  public visible: string = ''
}
```

Frontend Model

Doppeltes Verwalten der Models

- PHP-Backend in SORM
- Vue-Frontend in TS-Klasse

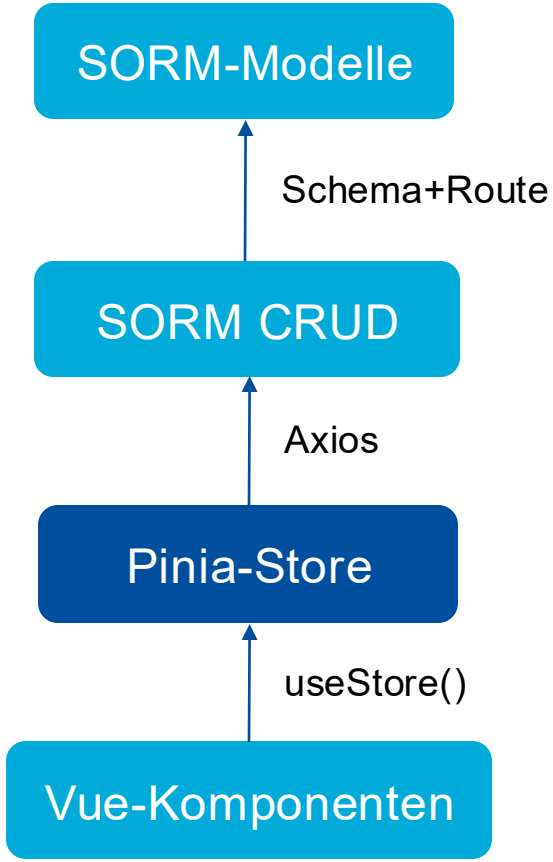
Lösungsansatz: Generieren der Models für Back- & Frontend

- OpenAPI Specification
- Datenbank-Schema als Quelle

Carl von Ossietzky
Universität
Oldenburg

Agenda

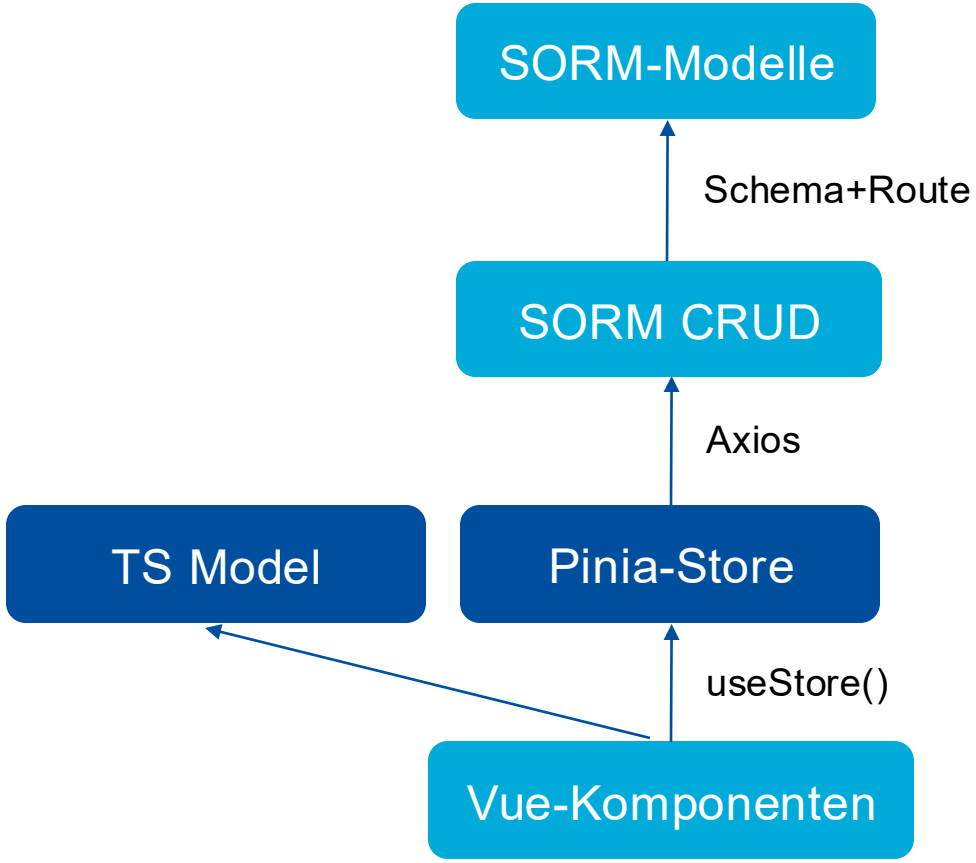
- Backend Schnittstelle
  - JSON-API Routen & Schema
- Frontend Schnittstelle
  - **Frontend Model**
  - Client State vs. Server State



Carl von Ossietzky
Universität
Oldenburg

Agenda

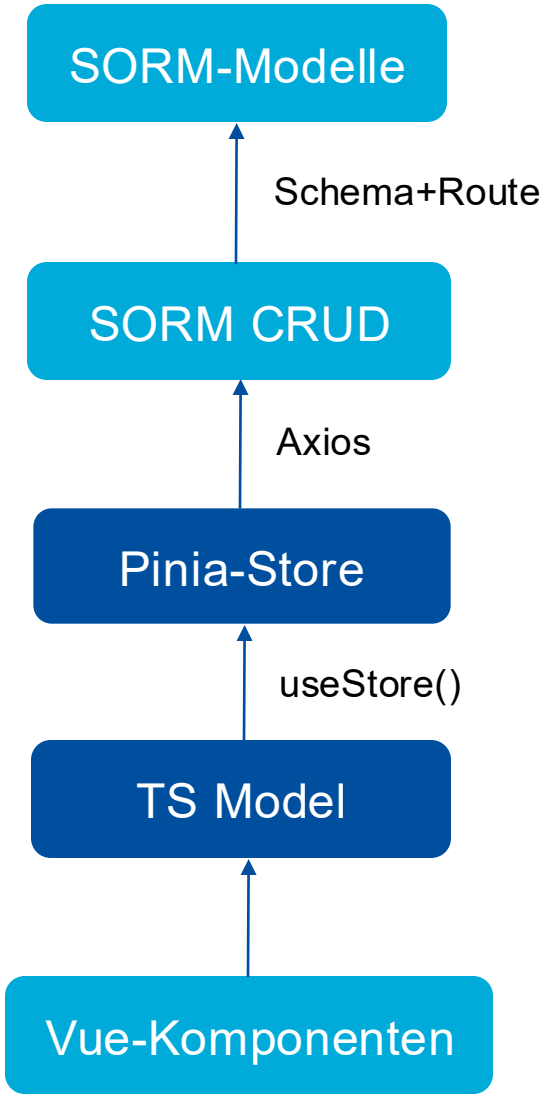
- Backend Schnittstelle
  - JSON-API Routen & Schema
- Frontend Schnittstelle
  - **Frontend Model**
  - Client State vs. Server State



Carl von Ossietzky
Universität
Oldenburg

Agenda

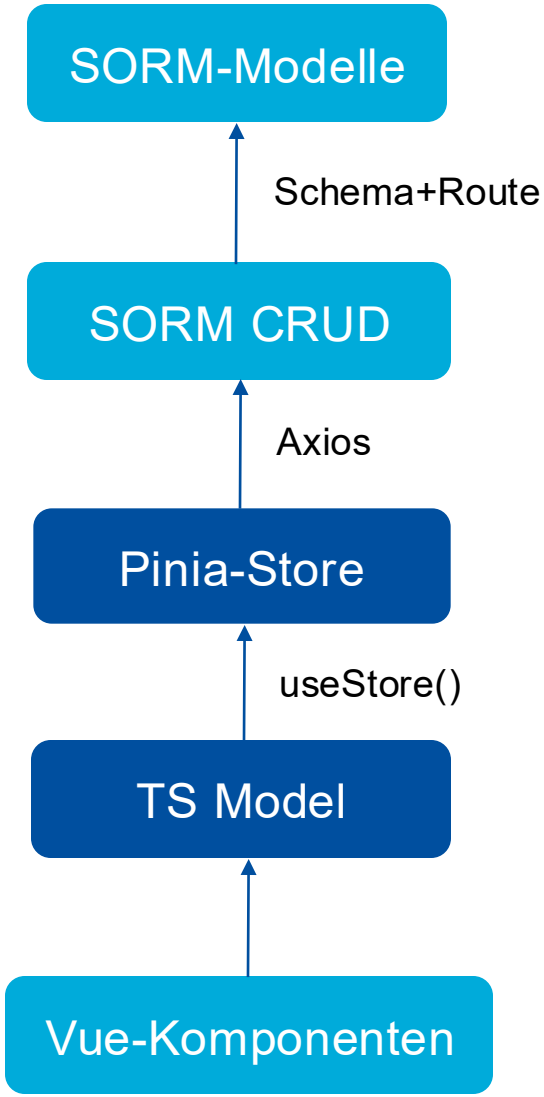
- Backend Schnittstelle
  - JSON-API Routen & Schema
- Frontend Schnittstelle
  - **Frontend Model**
  - Client State vs. Server State



Carl von Ossietzky
Universität
Oldenburg

Agenda

- Backend Schnittstelle
  - JSON-API Routen & Schema
- Frontend Schnittstelle
  - Frontend Model
  - **Client State vs. Server State**



Pinia - Client-State

- Offizielle State Management Library für Vue
- state, getters, setters
- Daten global in allen Komponenten verfügbar

Verwenden wir für

- GUI Daten (bspw. Ladestatus, filters)
- **SORM-Objekte** (bpsw. courses laden)

Tanstack Query - Queries

State Management für Daten vom Server – Server State

Queries

- Funktion, um Daten zu fetchen
- Identifikation der Daten über Query-Keys (Cache-Keys)

```
useQuery({ queryKey: ['todos'], queryFn: fetchTodoList })
useQuery({ queryKey: ['todo', 5], ... })
```

- QueryOptions
  - enabled, select etc.

Tanstack Query - Mutations

Mutations

- Funktion, um Daten zu verändern (Create, Update, Delete)
- Error- und Sucess-States

```
const { isPending, isError, error, isSuccess, mutate } = useMutation({
  mutationFn: (newTodo) => axios.post('/todos', newTodo),
})
```

Tanstack Query - Server State

- State Management für Daten vom Server – Server State
- Fetched Daten erst, wenn sie gebraucht werden
- Abhängige Requests
- Caching der Daten
- Automatisches refetching
- Infinite/Paginated Requests

Carl von Ossietzky Universität Oldenburg

Tanstack Query – SORM

Neue Abstraktionsschicht - Kapselung von Komplexität

resourceName – Pfad

queries – query-Instanz

- useAll
- useOne
- useSave
- useDelete

```
export class Semester extends Resource {
  static resourceName: string = 'semesters'
  static queries : { queryKeys: { all: readonly [string]; li...  = useResourceStore(Semester, CoreApiServices)

  public id: string = ''
  public title: string = ''
  public token: string = ''
  public start: string = ''
  public 'start-of-lectures': string = ''
  public 'end-of-lectures': string = ''
  public end: string = ''
  public 'is-current': string = ''
  public visible: string = ''

  Show usages  rufa9486
  public static findAll() : UseQueryReturnType<Semester[], Error> {
    return useQuery({
      ...Semester.queries.useAll({}, [], page: 1, limit: 50),
      select: (semesters : Semester[] ) : Semester[]  => Semester.sort(semesters),
    })
  }

  Show usages  rufa9486
  public static sort(semesters: Semester[], sort: 'desc' | 'asc' = 'desc') : Semester[]  {...}
}
```

Tanstack Query – SORM

In Vue-Komponente

- Statische Methode der ORM Klasse:

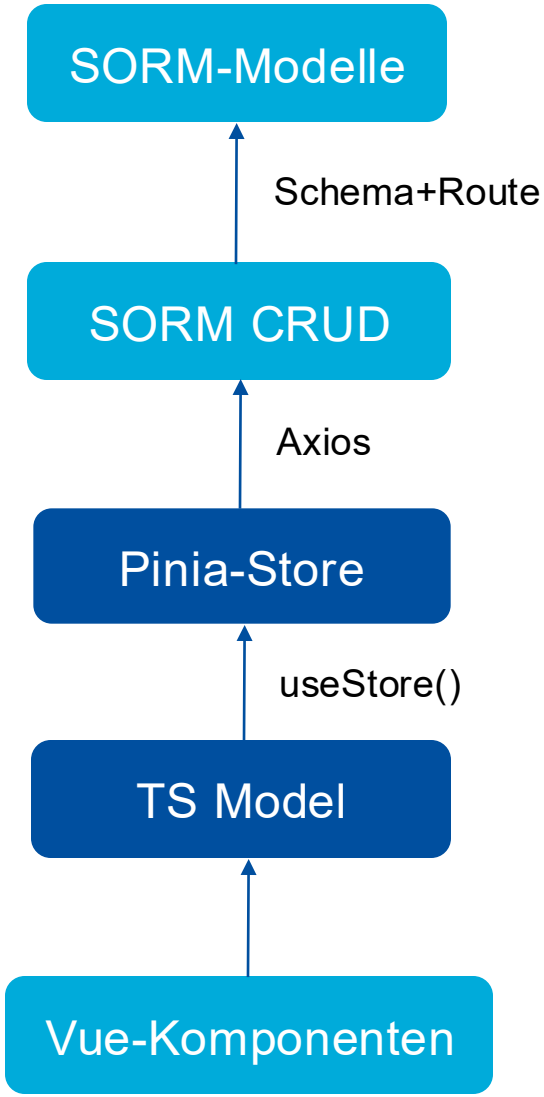
```
const { data: semesters } = Semester.findAll()
const { data: faculties } = Institute.findAllFaculties()
const { data: allOwnApplications } = Application.getAllOwn()
```

- Oder eigene Query:

```
const projectFilter = { project_id: board.project_id }
const { data: tasks, isLoading: t_isLoading } = useQuery({
  ...Task.queries.useAll(projectFilter),
  enabled: !!board.project_id,
})
```

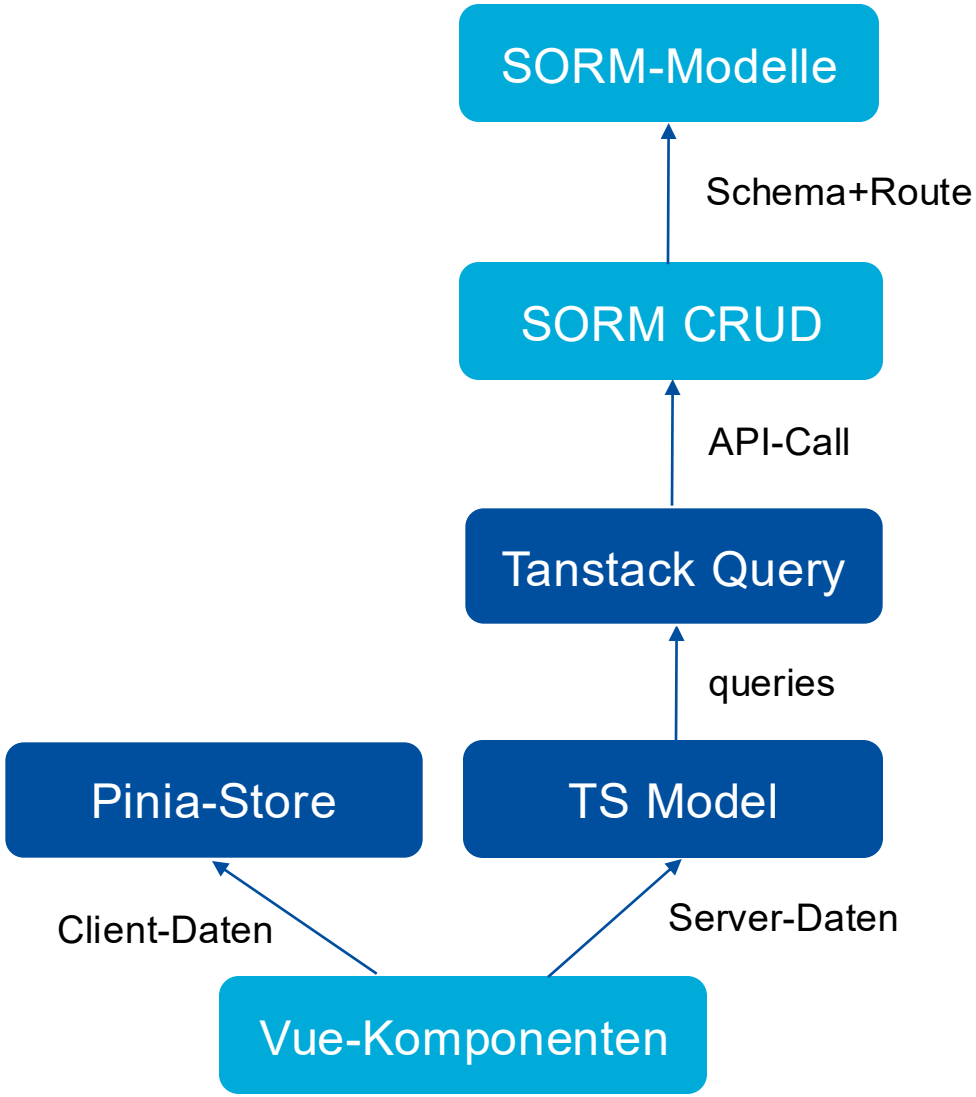
Agenda

- Backend Schnittstelle
  - JSON-API Routen & Schema
- Frontend Schnittstelle
  - Frontend Model
  - **Client State vs. Server State**



Agenda

- Backend Schnittstelle
  - JSON-API Routen & Schema
- Frontend Schnittstelle
  - Frontend Model
  - **Client State vs. Server State**



Frontend ORM & Tanstack Query

Vorteile

- Datentypen sind bekannt
  + Typehints für Attribute & Methoden
- Klassen-Methoden statt statischen
- Verständlicher für PHP-Entwickler
- SormCRUD vereinheitlich Route&Model
- Alle Vorteile von Tanstack
- TanStack für Models einfacher

Nachteile

- Doppelte Verwaltung
- Einführung von Komplexität mit Tanstack Query
- Bindung an Tanstack Query

Zielsetzung

Vue Entwicklung **nachvollziehbarer**,
effizienter und **strukturierter** gestalten

- Bessere Separation zwischen Front- und Backend
- Standard-Funktionalitäten einrichten und abkapseln
- Mehr Hilfestellung für Entwickler im Frontend
- Einheitlicher Technologie-Stack

Diskussion

- SORM CRUD
  - Verknüpfung von Model & JSON API sinnvoll?
  - Generierung der Models
- Frontend ORM
  - Unnötiger Overhead oder Ersparnis?
  - TS-Klassen oder TS-Typen?
- Tanstack Query
  - Zu komplex und unnötig oder praktisch und eine Ersparnis?