

Developer Documentation

Learn how to set up your development environment.

- [Repositories & CI/CD pipeline](#)
- [Writing new API Endpoints](#)
- [In Depth: Editor CRDT](#)

Repositories & CI/CD pipeline

This application consists of one main server and one or many rendering servers. For shared types, there is an exchange library.

Repositories

[Verfassungsbooks/Verfassungsbooks](#)

[Verfassungsbooks/Verfassungsbooks-Rendering-Server](#)

[Verfassungsbooks/Verfassungsbooks-Exchange-Lib](#)

CI/CD Pipeline

Currently, we are using GitHub actions to automatically build and test on new commits or pull requests. Pushes to the `Verfassungsbooks` master branch are also automatically deployed to the [staging test environment](#), pushes to the `Verfassungsbooks-Rendering-Server` master branch are automatically deployed to the staging test rendering server (editor-staging-rendering1.verfassungsblog.de)

GitHub Action Workflows

The repositories have three GitHub workflows configured:

- build-master.yml -> Builds and tests commits to the master branch, automatically deploys them to the staging system
- build-test-all.yml -> Builds and tests commits / pull requests to any branch (except master)
- push-to-prod.yml -> manually triggered; builds and tests a specific commit, deploys to productive system (editor.verfassungsblog.de / editor-rendering1.verfassungsblog.de)

Writing new API Endpoints

Security

Important: All non-public routes **must** use type Session in the routes parameters! Otherwise, no login is required to access it!

Minimal boilerplate:

```
#[get("/api/non/public/endpoint")]
pub async fn my_endpoint(
    _session: Session,
) -> Json<ApiResponse<>> {
    ApiResponse::new_data();
}
```

Endpoints

Typically, you should implement GET, POST, DELETE and PATCH routes. POST routes are only used to create new objects, PATCH routes are used to update an existing object.

GET Route Example

Route to get a specific section in a project

```
/// GET /api/projects/<project_id>/sections/<content_path>?<expand>
///
/// Parameters:
/// * project_id (string) - the projects uuid
/// * content_path (string) - path to a specific section, split by ':'
/// * expand (string, optional) - optionally expand one of these fields: authors, editors,
subsections
///
/// By default strips out subsections & only returns id's for authors and editors.
/// Use the optional expand query parameter to expand these fields
/// E.g. ?expand=authors,editors,subsections will show the full data
```

```
///  
#[get("/api/projects/<project_id>/sections/<content_path>?<expand>")]  
pub async fn get_section(  
    project_id: &str,  
    content_path: &str,  
    expand: Option<&str>,  
    _session: Session,  
    settings: &State<Settings>,  
    project_storage: &State<Arc<ProjectStorage>>,  
    data_storage: &State<Arc<DataStorage>>  
) -> JsonResult<APISectionResult> {  
    ...  
}
```

Test your routes

You may test your new endpoints with curl. First, obtain a session cookie via your browser. Then use curl:

```
curl -v --cookie 'session=INSERTYOURCOOKIEHERE' <host>/api/<route>
```

In Depth: Editor CRDT

Work in Progress! CRDT isn't implemented yet.

Ablauf:

1. Client möchte Kapitel bearbeiten, Client öffnet web socket mit Server
2. Client bekommt yDoc von Server
3. Client wandelt yDoc Änderungen in EditorJS Blockänderungen um <-- Herausforderung 1
4. Bei Änderungen an Blöcken wird Änderung in yDoc Änderung umgewandelt <-- Herausforderung 2
5. yDoc Änderung wird an Server geschickt
6. Server schickt yDoc Änderung an alle anderen Clients

Beispiel EditorJS Block JSON:

```
{
  "id": "jaJh1uNPLq",
  "type": "paragraph",
  "data": {
    "text": "In the spring of 2024, video cameras from numerous global news outlets turned their attention to a court in Strasbourg. People traveled from across Europe, gathering with signs in front of the courthouse. Minors from Portugal stood alongside senior citizens from Switzerland to witness one of the most significant moments in the recent history of the European Convention on Human Rights. For the first time, the European Court of Human Rights (ECtHR) ruled on the impact of the climate crisis on human rights and what this means for the Convention's signatory states. The court's Grand Chamber issued rulings on three cases: the case of <i>Carême v. France</i><citation data-key=\"ECtHR_careme_2024\">C</citation> (<i>Carême</i>), brought by the former mayor of Grande-Synthe, France; the case of <i>Duarte Agostinho and Others v. Portugal and 32 Others</i><citation data-key=\"ECtHR_duarte_2024\">C</citation>&nbsp;(<i>Duarte Agostinho</i>), brought by six youth applicants from Portugal; and the case of <i>Verein KlimaSeniorinnen Schweiz and Others v. Switzerland</i><citation data-key=\"noauthor_hudoc_nodate\">C</citation><i>&nbsp;</i>(<i>KlimaSeniorinnen</i>). While the first two cases were deemed inadmissible, the court handed down a ruling in
```

Klimaseniorinnen, which is already regarded as one of the most important judgments in climate change litigation. The court stated that “the state has a positive duty to adopt, and effectively implement in practice, regulations and measures capable of mitigating the existing and potentially irreversible future effects of climate change”. Regarding the Swiss government, one of the Convention’s signatory states, the court concluded that by failing to put in place a sufficient domestic regulatory framework for climate change mitigation, the government violated Article 8 of the European Convention on Human Rights (ECHR), the right to respect for private and family life. Article 8 requires “that each Contracting State undertake measures for the substantial and progressive reduction of their respective GHG emission levels, with a view to reaching net neutrality within, in principle, the next three decades” (*KlimaSeniorinnen*, para. 548). Moreover, the Court found a violation of the right of access to court (Article 6 of the ECHR). "

```
  },
  "tunes": {}
},
{
  "id": "M29CgX78Fc",
  "type": "header",
  "data": {
    "text": "The three climate rulings",
    "level": 1
  },
  "tunes": {}
},
```

EditorJS hat eine insert Funktion, über die können wir Blöcke dann einfügen:

<https://editorjs.io/blocks/#insert>

Wir müssen ein äquivalent in yrs Typen schaffen, tunes brauchen wir nicht, id's vermutlich auch nicht (?). Data muss eine map sein und hat neben text optionale andere einträge

Protokoll

Clients und Server kommunizieren über Websockets miteinander. Das erste Byte jeder Nachricht bestimmt den Nachrichtentyp:

1st Byte Decimal Value	Name	Binary / JSON?	Sent from	Data	Description

10	CONNECT	JSON	Client	document_id	Connect to server with existing session and edit document with document_id
11	WELCOME	JSON	Server	client_id	todo
20	GETDOC	Binary	Client	StateVector	todo
21	DOCUPDATE	Binary	Server	Document update	todo
30	DOCUPDATE	Binary	Client / Server	Document update	todo
40	SETCURSOR	JSON	Client/Server	client_id, block_id, start, end (optional, only for selections)	todo
41	REMOVECURSOR	JSON	Server	client_id	todo
50	DISCONNECT	JSON	Client	client_id	todo
60	ERROR	JSON	Server	status_code, error_msg	error occurred, e.g. authorization failed due to invalid session_id, document not found etc.

Auth: The client has to send a valid session cookie when establishing the websocket.