Development/Test-System

Entwicklungssystem

* Java 11
* Postgresql 14.0
* Postgis Extension für Postgresql:   
  POSTGIS="3.2.3 3.2.3" [EXTENSION] PGSQL="140" GEOS="3.10.3-CAPI-1.16.1" PROJ="7.2.1" GDAL="GDAL 3.4.3, released 2022/04/22" LIBXML="2.9.9" LIBJSON="0.12" LIBPROTOBUF="1.2.1" WAGYU="0.5.0 (Internal)" RASTER
* GeoServer 2.21.1
* Bidoversitätsatlas – Live-System: Stand Apr 23

Entwicklungsumgebung

* IntelliJ IDEA 2022.3.1
* Kotlin 1.7.10
* KTOR 2.2.4
* Java 11
* Gradle 7.5.1

Repository

* <https://github.com/biodivAtlasAT/ecosys>

Erweiterungen und anschließendes Deployment auf Server

* Download vom Repo
* In Intellij öffnen
* Task “ecosys-shadowJar” ausführen (Gradle-Task)
* Service auf Server stoppen: sudo systemctl stop ecosys
* “ecosys-all.jar” vom “../build/lib”-Verzeichnis auf den Server ins Verzeichnis „/data/ecosys“ kopieren
* Service starten: sudo systemctl start ecosys

Bei der Erstinstallation müssen die entsprechenden Programmversionen installiert werden und es muss eine Datenbank in Postgresql mit dem Namen „ecosys“ existieren. Alle anderen Datenbank-Aktionen werden direkt beim Starten der Applikation ausgeführt. Die Konfigurationen müssen am Server in der Datei „/data/ecosys/ecosys.conf“ erfolgen.  
Wenn Änderungen in der Datenbank notwendig sind, dann müssen diese mittels Migrationen (flyway Struktur) ausgeführt werden. Diese befinden sich im Verzeichnis „..\resources\migration“.

Ecosys – Frontend/Backend Integration

## Ökosystemleistungen:

Aufruf direkt im Root-Verzeichnis: http…../

Sobald eine neue Version des Frontends verfügbar ist, müssen die Dateien im Verzeichnis „..ecosys\src\main\resources\static\frontend“ hineinkopiert werden.

Bei einem Server-Restart wird die allgemeine Atlas Navigation (lt. Konfiguration) vom Server downgeloadet und mit der index.html verknüpft. Alle Pfade zu den Ressourcen werden ausgetauscht, damit diese von Ktor aus erreichbar sind. Um einen korrekten Austausch zu ermöglichen, müssen die vereinbarten Variablen in der Navigation (siehe Branding-Url) existieren. Den entsprechenden Codeteil findet man in der Klasse „DataCache“. Die verarbeitete Datei wird im Datencache (siehe Konfiguration) unter „navigation.html“ abgespeichert.

## Biotoptypen:

Aufruf direkt im Root-Verzeichnis: http…../biotop

Sobald eine neue Version des Frontends verfügbar ist, müssen die Dateien im Verzeichnis „..ecosys\src\main\resources\static\frontendBiotop“ hineinkopiert werden.

Die Logik entspricht der der Ökosystemleistungen. Die verarbeitete Datei lautet jedoch „navigationBiotop.html“.

Ecosys – Frontend/Backend Interaktion

Alle verfügbaren API-Requests sind unter der URL: <https://ecosys.biodivdev.at/swagger> verfügbar und dokumentiert. In diesem Dokument werden lediglich die logischen Zusammenhänge näher erläutert.

## Ökosystemleistungen - Logik

1. Frontend: Anzeige eines Layers (zB: „ALA:hauptregion\_noe“)

2. Frontend: bei Klick auf eine spezifische Region des Layers, werden vom Geoserver die zugehörigen Eigenschaften dieser Region zurückgeliefert (zB: fid,HRNR,HRCD,HRNAME,EMASST,LASTUPDATE).

3. Backend Request: Auslieferung der Liste der verfügbaren Layer – es wird für jeden Layer eine „KEY“-Spalte ausgeliefert!

<http://127.0.0.1:8080/api/layers>

|  |
| --- |
| {  "error": {  "no": 0,  "msg": ""  },  "layers": [  {  "id": 3,  "name": "hauptregion\_noe",  "key": "HRNAME"  },  {  "id": 2,  "name": "viertelnoe\_iso\_wgs84",  "key": "VIERTELNAM"  }  ]  } |

4. Um die Rasterdaten (Package) mit einer spezifischen Region eines Layers zu verschneiden, muss der Wert der Eigenschaft(„Waldviertel“) aus dem Frontend (zB: HRNAME=“Waldviertel“) beim API-Request als „layerKey“ mitgegeben werden:

<http://127.0.0.1:8080/api/layerData?packageID=1&layerID=2&layerKey=Waldviertel>

## Biotoptypen - Logik

1. Auslieferung aller freigeschalteten Projekte, um die IDs und die zugehörigen Projektnamen zu erhalten

<http://127.0.0.1:8080/api/bt/projects>

|  |
| --- |
| {  "error": {  "no": 0,  "msg": ""  },  "projects": [  {  "id": 1,  "name": "Biotoptypen - UBA"  },  {  "id": 2,  "name": "Projekt 2024 - coopNatura"  }  ]  } |

2. Auslieferung der Detaildaten zu einem Projekt – unter der Eigenschaft dataResource“ werden alle Detaildaten aus der Datenressource im Collectory angezeigt (im Beispiel nicht angezeigt)

<http://127.0.0.1:8080/api/bt/projects>/{projectId}

|  |
| --- |
| {  "error": {  "no": 0,  "msg": ""  },  "project": {  "id": 1,  "name": "Biotoptypen - UBA",  "enabled": true,  "epoch": "August 23",  "area": "Mariazellerland",  "resource": "4",  "classId": 1,  "classInfo": null,  "classMap": "",  "geoserverWorkspace": "ECO",  "geoserverLayer": "bezirk\_wgs84\_iso",  "colTypesCode": "BKZ",  "colTypesDescription": "PB",  "colSpeciesCode": null,  "speciesFileName": null,  "speciesColCodeName": null,  "speciesColNameName": null,  "hierarchyId": -1,  "dataResource": null  }  } |

3. Link zu Geoserver (aus Frontend)

Um einen Layer für Biotoptypen im Frontend anzuzeigen, muss ein WMS-Request zum Geoserver abgesetzt werden.

Geoserver Aufruf, um diesen Layer „bezirk\_wgs84\_iso“ anzuzeigen

http://127.0.0.1:8081/geoserver/ECO/wms?service=WMS&version=1.1.0&request=GetMap&layers=ECO%3Abezirk\_wgs84\_iso&bbox=9.530749012203303%2C46.37230501153543%2C17.16077331128593%2C49.020525464124105&width=768&height=330&srs=EPSG%3A4326&styles=&format=application/openlayers#

Datenursprung:

Geoserver-Url: definiert im Frontend „index.html“ im letzten JS-Block (diese Konfiguration wird durch die Ecosys-Konfigurationsvariable „geoserver.url“ beim Aufbau des Navigations-Caches System-abhängig gesetzt):

var url\_geoserver = "https://spatial.biodiversityatlas.at/geoserver";

🡪tbd (Georg)

Geoserver Arbeitsbereich: dieser wird in der Ecosys-Konfiguration fix gesetzt: „geoserver.workspace“ und über die Detaildaten zum Projekt (siehe JSON oberhalb) ausgeliefert

Name des Layers: wird über die Detaildaten zum Projekt (siehe JSON oberhalb) ausgeliefert

4. Filter für Projekt

Über den API-Request: “./api/bt/projects/4/filter” kann die Biotoptypen-Hierarchie angezeigt werden. Dabei werden die einzelnen Ebenen des Hierarchiebaumes linear in der richtigen Reihenfolge übergeben, sodass die Liste im Frontend (als Liste!) ohne zusätzlicher Logik angezeigt werden kann.

Bei jedem Eintrag wird auch der entsprechende CQL-Filter mitgegeben, der bei der Anzeige der Daten aus dem Geoserver verwendet werden kann. D.h. clientseitig ist keine Logik notwendig, um übergeordnete Hierarchien auszuwählen.

Die Level-Number gibt an in welcher Ebene sich dieser Eintrag befindet, sodass durch „Einrücken“ eine Art Baumstruktur in Listenform angezeigt werden kann.

5. Ausgewählte Bereiche eines Layers anzeigen („Filter“-Mechanismus)

Dazu wird die CQL-Query Language verwendet.

http://127.0.0.1:8081/geoserver/ECO/wms?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&FORMAT=image/png&TRANSPARENT=true&STYLES&LAYERS=ECO:bezirk\_wgs84\_iso&exceptions=application/vnd.ogc.se\_inimage&CQL\_FILTER=BKZ=506&SRS=EPSG:4326&WIDTH=768&HEIGHT=330&BBOX=9.129638671875,45.87890625,17.567138671875,49.50439453125

Die Eigenschaft, die zur Einschränkung dient (zB „BKZ“), wird in den Projektdetails ausgeliefert. Der Wert „506“ ergibt sich aus der Auswahl aus der Filter-Hierarchie. Achtung: die Kriterien müssen „XML-encoded“ ausgeliefert werden

Zum Testen des CQL-Scripts kann am einfachsten die Layer-Anzeige im Geoserver verwendet werden