УТВЕРЖДЕН
RU.17701729.04.16–01 12 01–ЛУ

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ ”WEB GIS”

Текст программы

RU.17701729.04.16–01 12 01

Листов 124

Инв. № подп.	Подп. и дата	Взам инв. №	Инв. № дубл.	Подп. и дата

Москва 2018

АННОТАЦИЯ

В данном программном документе представлен текст программы ”Клиент-серверное приложение ’Web GIS’”. В разделе ”Текст программы” находится текст программы, распределенный по файлам.

Оформление данного документа произведено по требованиям ЕСПД (ГОСТ 19.102-77[1], ГОСТ 19.103-77[2], ГОСТ 19.104-78[3], ГОСТ 19.105-78[4], ГОСТ 19.106-78[5], ГОСТ 19.401-78[6]

[1]Стадии разработки ГОСТ 19.102-77.

[2]Обозначения программ и программных документов ГОСТ 19.103-77.

[3]Основные надписи ГОСТ 19.104-78.

[4]Общие требования к программным документам ГОСТ 19.105-78.

[5]Требования к программным документам, выполненным печатным способом ГОСТ 19.106-78.

[6]Текст программы. Требования к содержанию и оформлению ГОСТ 19.401-78.

СОДЕРЖАНИЕ

1 Текст программы **85**
1.1 web-gis-front-end/static.json 85
1.2 web-gis-front-end/.eslintrc 85
1.3 web-gis-front-end/.stylelintrc 85
1.4 web-gis-front-end/package.json 85
1.5 web-gis-front-end/public/index.html 86
1.6 web-gis-front-end/public/index.css 88
1.7 web-gis-front-end/public/manifest.json 88
1.8 web-gis-front-end/src/auth.js 89
1.9 web-gis-front-end/src/index.js 90
1.10 web-gis-front-end/src/config.js 91
1.11 web-gis-front-end/src/history.js 91
1.12 web-gis-front-end/src/propTypes.js 91
1.13 web-gis-front-end/src/registerServiceWorker.js 92
1.14 web-gis-front-end/src/api.js 95
1.15 web-gis-front-end/src/utils/data.js 96
1.16 web-gis-front-end/src/utils/olUtils.js 98
1.17 web-gis-front-end/src/utils/index.jsx 106
1.18 web-gis-front-end/src/utils/common.js 107
1.19 web-gis-front-end/src/components/FormSlider.jsx 109
1.20 web-gis-front-end/src/components/OutputItem.jsx 110
1.21 web-gis-front-end/src/components/Bold.jsx 111
1.22 web-gis-front-end/src/components/FormColorField.jsx 112
1.23 web-gis-front-end/src/components/LayerProperty/index.jsx 114
1.24 web-gis-front-end/src/components/LayerDetails/index.jsx 114
1.25 web-gis-front-end/src/components/Form/index.js 115
1.26 web-gis-front-end/src/components/SubmitDialog/index.jsx 115
1.27 web-gis-front-end/src/components/Loading/index.jsx 116
1.28 web-gis-front-end/src/components/MenuBar/index.jsx 117
1.29 web-gis-front-end/src/components/LoadableList/index.jsx 118
1.30 web-gis-front-end/src/components/LoadableTreeList/index.jsx 119
1.31 web-gis-front-end/src/components/ProjectListCard/index.jsx 121
1.32 web-gis-front-end/src/components/TreeList/index.jsx 122
1.33 web-gis-front-end/src/components/ProjectsPage/index.jsx 124
1.34 web-gis-front-end/src/components/CloseButton/index.jsx 125
1.35 web-gis-front-end/src/components/IndexPage/index.jsx 125
1.36 web-gis-front-end/src/components/FormTextField/index.jsx 126
1.37 web-gis-front-end/src/components/PaperCard/index.jsx 127
1.38 web-gis-front-end/src/components/CheckButton/index.jsx 128
1.39 web-gis-front-end/src/components/ProjectDetails/index.jsx 128
1.40 web-gis-front-end/src/components/AddButton/index.jsx 130

1.41 web-gis-front-end/src/components/Link/index.jsx 130
1.42 web-gis-front-end/src/components/Logo/index.jsx 131
1.43 web-gis-front-end/src/components/Callback/index.jsx 131
1.44 web-gis-front-end/src/components/FormSelectField/index.jsx 132
1.45 web-gis-front-end/src/components/LayerProperties/index.jsx 133
1.46 web-gis-front-end/src/actions/layerCategories.js 134
1.47 web-gis-front-end/src/actions/commandLine.js 135
1.48 web-gis-front-end/src/actions/auth.js 136
1.49 web-gis-front-end/src/actions/index.js 136
1.50 web-gis-front-end/src/actions/projects.js 137
1.51 web-gis-front-end/src/actions/layers.js 138
1.52 web-gis-front-end/src/actions/sideMenu.js 138
1.53 web-gis-front-end/src/actions/map.js 139
1.54 web-gis-front-end/src/containers/Root.jsx 139
1.55 web-gis-front-end/src/containers/ProjectLayersContainer/index.jsx 141
1.56 web-gis-front-end/src/containers/LayersContainer/index.jsx 144
1.57 web-gis-front-end/src/containers/Layout/index.jsx 146
1.58 web-gis-front-end/src/containers/ProjectDetailsContainer/index.jsx 148
1.59 web-gis-front-end/src/containers/LayersPage/styles.css 149
1.60 web-gis-front-end/src/containers/LayersPage/index.jsx 150
1.61 web-gis-front-end/src/containers/CommandLine/index.jsx 152
1.62 web-gis-front-end/src/containers/ViewProjectPage/index.jsx 154
1.63 web-gis-front-end/src/containers/SideMenu/index.jsx 161
1.64 web-gis-front-end/src/containers/CreateLayerForm/index.jsx 162
1.65 web-gis-front-end/src/containers/CreateProjectForm/index.jsx 164
1.66 web-gis-front-end/src/containers/ProjectsList/index.jsx 165
1.67 web-gis-front-end/src/containers/LayerDetailsContainer/index.jsx 167
1.68 web-gis-front-end/src/containers/LayersList/index.jsx 167
1.69 web-gis-front-end/src/reducers/layerCategories.js 168
1.70 web-gis-front-end/src/reducers/commandLine.js 169
1.71 web-gis-front-end/src/reducers/auth.js 170
1.72 web-gis-front-end/src/reducers/index.js 170
1.73 web-gis-front-end/src/reducers/projects.js 171
1.74 web-gis-front-end/src/reducers/layers.js 173
1.75 web-gis-front-end/src/reducers/sideMenu.js 174
1.76 web-gis-front-end/src/reducers/map.js 174
1.77 web-gis-front-end/src/store/configureStore.js 175
1.78 web-gis-tile-service/retile.sh 176
1.79 web-gis-tile-service/build.gradle 176
1.80 web-gis-tile-service/settings.gradle 177
1.81 web-gis-tile-service/docker.sh 177
1.82 web-gis-tile-service/gradle/wrapper/gradle-wrapper.properties 177
1.83 web-gis-tile-service/src/main/java/com/webgis/tileservice/ App.java 177

1.84 web-gis-tile-service/src/main/java/com/webgis/tileservice/ server/Response.java 180
1.85 web-gis-tile-service/src/main/java/com/webgis/tileservice/ server/Handler.java 180
1.86 web-gis-tile-service/src/main/java/com/webgis/tileservice/ server/Route.java 180
1.87 web-gis-tile-service/src/main/java/com/webgis/tileservice/ server/RouteTable.java 182
1.88 web-gis-tile-service/src/main/java/com/webgis/tileservice/ server/WebServer.java 182
1.89 web-gis-tile-service/src/main/java/com/webgis/tileservice/ server/Request.java 192
1.90 web-gis-tile-service/build.gradle 194
1.91 web-gis-tile-service/src/test/java/com/webgis/backend/ DemoApplicationTests.java 195
1.92 web-gis-tile-service/src/main/java/com/webgis/backend/ DemoApplication.java 196
1.93 web-gis-tile-service/src/main/java/com/webgis/backend/ configurations/SecurityConfig.java 196
1.94 web-gis-tile-service/src/main/java/com/webgis/backend/ configurations/RepositoryConfig.java 197
1.95 web-gis-tile-service/src/main/java/com/webgis/backend/repositories/ LayerRepository.java 198
1.96 web-gis-tile-service/src/main/java/com/webgis/backend/repositories/ ProjectRepository.java 199
1.97 web-gis-tile-service/src/main/java/com/webgis/backend/repositories/ LayerCategoryRepository.java 199
1.98 web-gis-tile-service/src/main/java/com/webgis/backend/models/ LayerCategory.java 200
1.99 web-gis-tile-service/src/main/java/com/webgis/backend/models/ Project.java 200
1.100 web-gis-tile-service/src/main/java/com/webgis/backend/models/ Layer.java 201
1.101 web-gis-tile-service/src/main/java/com/webgis/backend/controllers/ APIController.java 201

1. ТЕКСТ ПРОГРАММЫ

1.1. web-gis-front-end/static.json

```
}
```

1.2. web-gis-front-end/.eslintrc

```
{
  "extends": ["react-app", "airbnb"],
  "rules": {
    "jsx-ally/anchor-is-valid": ["error", {
      "specialLink": ["to"]
    }]
  }
}
```

1.3. web-gis-front-end/.stylelintrc

```
{
  "processors": ["stylelint-processor-styled-components"],
  "extends": [
    "stylelint-config-standard",
    "stylelint-config-styled-components"
  ],
  "syntax": "scss"
}
```

1.4. web-gis-front-end/package.json

```
{
  "name": "web-gis-front-end",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "auth0-js": "^9.2.3",
    "axios": "^0.18.0",
    "geotiff": "^0.4.1",
```

```
    "history": "^4.7.2",
    "immutable": "^3.8.2",
    "lodash": "^4.17.5",
    "material-ui": "^0.20.0",
    "ol": "^4.6.4",
    "plotty": "^0.3.0",
    "prop-types": "^15.6.0",
    "react": "^16.2.0",
    "react-color": "^2.14.1",
    "react-dom": "^16.2.0",
    "react-redux": "^5.0.6",
    "[illegible]
    "[illegible]
    "[illegible]
    "[illegible]2",
    "[illegible]
    "[illegible]
    "redux-form": "^7.3.0",
    "redux-logger": "^3.0.6",
    "redux-thunk": "^2.2.0",
    "styled-components": "^3.2.3"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject",
    "lint:css": "stylelint './src/**/*.js'",
  },
  "devDependencies": {
    "babel-core": "^6.26.0",
    "babel-runtime": "^6.26.0",
    "eslint-config-airbnb": "^16.1.0",
    "stylelint": "^9.1.3",
    "stylelint-config-standard": "^18.2.0",
    "stylelint-config-styled-components": "^0.1.1",
    "stylelint-processor-styled-components": "^1.3.1"
  }
}
```

1.5. web-gis-front-end/public/index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
  ↪ initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <!--
      manifest.json provides metadata used when your web app
  ↪ is added to the
      homescreen on Android. See https://developers.google.
```

```
    <link rel="stylesheet" href="https://openlayers.org/en/v4
  ↪ .6.4/css/ol.css" type="text/css">
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder
  ↪  during the build.
      Only files inside the `public` folder can be referenced
  ↪  from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/
  ↪ favicon.ico" will
      work correctly both with client-side routing and a non-
  ↪ root public URL.
      Learn how to configure a non-root public URL by running
  ↪  `npm run build`.
    -->
    <title>Web GIS</title>
  </head>
  <body>
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an
```

```
↪  empty page.

    You can add webfonts, meta tags, or analytics to this
↪ file.
    The build step will place the bundled scripts into the
↪ <body> tag.

    To begin the development, run ‘npm start‘ or ‘yarn
↪ start‘.
    To create a production bundle, use ‘npm run build‘ or ‘
↪ yarn build‘.
  -->
  </body>
</html>
```

1.6. web-gis-front-end/public/index.css

```
html,
body {
  margin: 0;
  padding: 0;
}

#root {
  flex: 1;
  display: flex;
}
```

1.7. web-gis-front-end/public/manifest.json

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
```

```
    }
  ],
  "start_url": "./index.html",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

1.8. web-gis-front-end/src/auth.js

```
import { WebAuth } from 'auth0-js';

import history from './history';

const auth0 = new WebAuth({
  domain: 'web-gis.eu.auth0.com',
  clientID: 'mQdoIvkoFkpFqThFJRhl4pUwuprZ2kfL',
  redirectUri: `${window.location.origin}/callback`,
  audience: 'https://web-gis-back-end.herokuapp.com/api/',
  responseType: 'token id_token',
  scope: 'openid'
```

```
  localStorage.setItem('access_token', authResult.accessToken
  ↪ );
  localStorage.setItem('id_token', authResult.idToken);
  localStorage.setItem('expires_at', expiresAt);
  localStorage.setItem('user_id', authResult.idTokenPayload.
  ↪ sub);
  // navigate to the home route
  history.replace('/');
}

export function logout() {
  // Clear access token and ID token from local storage
  localStorage.removeItem('access_token');
  localStorage.removeItem('id_token');
  localStorage.removeItem('expires_at');
  localStorage.removeItem('user_id');
```

```
  // navigate to the home route
  history.replace('/');
}

export function isAuthenticated() {
  // Check whether the current time is past the
  // access token's expiry time
  const expiresAt = JSON.parse(localStorage.getItem('
  ↪ expires_at'));
  return new Date().getTime() < expiresAt;
}

export function getUserId() {
  return localStorage.getItem('user_id');
}

export function login() {
  auth0.authorize();
}

export function handleAuthentication(dispatch,
  ↪ successActionCreator, failureActionCreator) {
  auth0.parseHash(window.location.hash, (err, authResult) =>
  ↪ {
    if (authResult && authResult.accessToken && authResult.
  ↪ idToken) {
      setSession(authResult);
      dispatch(successActionCreator(authResult.idTokenPayload
  ↪ .sub));
      history.replace('/');
                                                            e for
    }
  });
}
```

1.9. web-gis-front-end/src/index.js

```
// @flow
```

```
/* eslint-disable react/jsx-filename-extension */
import React from 'react';
import { render } from 'react-dom';

import Root from './containers/Root';
import configureStore from './store/configureStore';

const store = configureStore();

const root = document.getElementById('root');

if (root) {
  render(
    <Root store={store} />,
    root,
  );
}
```

1.10. web-gis-front-end/src/config.js

```
// @flow
/* eslint-disable import/prefer-default-export */
export const BACKEND_URL =
  ↪ eyJ1IjoiYXNichIIa2hvdIIsImEiOiJjamc5awdybm8ycIFjMndtb2M4YnZ0
  ↪ .2BQvFMVU5xd8WvMD_B6UzA';
```

1.11. web-gis-front-end/src/history.js

```
// @flow
import { createBrowserHistory } from 'history';

export default createBrowserHistory();
```

1.12. web-gis-front-end/src/propTypes.js

```
import PropTypes from 'prop-types';

/* eslint-disable import/prefer-default-export */
```

```
export const treePropType = PropTypes.arrayOf(PropTypes.shape
  ↪ ({
  id: PropTypes.string.isRequired,
  name: PropTypes.string.isRequired,
  children: PropTypes.arrayOf(PropTypes.shape({
    id: PropTypes.string.isRequired,
    name: PropTypes.string.isRequired,
  })),
}));
/* eslint-enable */

export const layerPropType = PropTypes.shape({
  id: PropTypes.string.isRequired,
  name: PropTypes.string.isRequired,
});
```

1.13. web-gis-front-end/src/registerServiceWorker.js

```
// In production, we register a service worker to serve
  ↪ assets from local cache.

// This lets the app load faster on subsequent visits in
```

```
// To learn more about the benefits of this model, read https
  ↪ ://goo.gl/KwvDNy.
// This link also includes instructions on opting out of this
  ↪  behavior.

const isLocalhost = Boolean(window.location.hostname === '
  ↪ localhost' ||
    // [::1] is the IPv6 localhost address.
    window.location.hostname === '[::1]' ||
    // 127.0.0.1/8 is considered localhost for IPv4.
    window.location.hostname.match
  ↪ (/^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$
  ↪ /));
```

```
export default function register() {
  if (process.env.NODE_ENV === 'production' && 'serviceWorker
  ↪ ' in navigator) {
    // The URL constructor is available in all browsers that
  ↪ support SW.
    const publicUrl = new URL(process.env.PUBLIC_URL, window.
  ↪ location);
    if (publicUrl.origin !== window.location.origin) {
      // Our service worker won't work if PUBLIC_URL is on a
  ↪ different origin
      // from what our page is served on. This might happen
  ↪ if a CDN is used to
      // serve assets; see https://github.com/
```

[redacted]

```
      if (isLocalhost) {
        // This is running on localhost. Lets check if a
  ↪ service worker still exists or not.
        checkValidServiceWorker(swUrl);
      } else {
        // Is not local host. Just register service worker
        registerValidSW(swUrl);
      }
    });
  }
}

function registerValidSW(swUrl) {
  navigator.serviceWorker
    .register(swUrl)
    .then((registration) => {
      registration.onupdatefound = () => {
        const installingWorker = registration.installing;
        installingWorker.onstatechange = () => {
          if (installingWorker.state === 'installed') {
            if (navigator.serviceWorker.controller) {
              // At this point, the old content will have
```

```
  ↪ been purged and
                // the fresh content will have been added to
  ↪ the cache.
                // It's the perfect time to display a "New
  ↪ content is
                // available; please refresh." message in your
  ↪ web app.
                console.log('New content is available; please
  ↪ refresh.');
              } else {
                // At this point, everything has been precached
  ↪ .
                // It's the perfect time to display a
                // "Content is cached for offline use." message
```

```
      };
    })
    .catch((error) => {
      console.error('Error during service worker registration
  ↪ :', error);
    });
}

function checkValidServiceWorker(swUrl) {
  // Check if the service worker can be found. If it can't
  ↪ reload the page.
  fetch(swUrl)
    .then((response) => {
      // Ensure service worker exists, and that we really are
  ↪  getting a JS file.
      if (
        response.status === 404 ||
        response.headers.get('content-type').indexOf('
  ↪ javascript') === -1
      ) {
        // No service worker found. Probably a different app.
  ↪  Reload the page.
        navigator.serviceWorker.ready.then((registration) =>
```

```
  ↪ {
          registration.unregister().then(() => {
            window.location.reload();
          });
        });
      } else {
        // Service worker found. Proceed as normal.
        registerValidSW(swUrl);
      }
    })
    .catch(() => {
      console.log('No internet connection found. App is
  ↪ running in offline mode.');
    });
}

export function unregister() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.ready.then((registration) => {
      registration.unregister();
    });
  }
}
```

1.14. web-gis-front-end/src/api.js

```
import axios from 'axios';
import _ from 'lodash';
```

```
});

export const ping = () => createInstance().get('/ping');

export const getProjects = async (ownerId) => {
  const res = await createInstance().get('/projects/search/
  ↪ findByOwnerId', { params: { ownerId } });
  return _.get(res, 'data._embedded.projects');
```

```
};

export const postProject = async (body) => {
  const res = await createInstance().post('/projects', body);

  return res.data;
};

export const updateProject = async (projectId, body) => {
  const res = await createInstance().patch(`/projects/${
  ↪ projectId}`, body);

  return res.data;
};

export const deleteProject = projectId => createInstance().
  ↪ delete(`/projects/${projectId}`);

export const getLayers = async (userId) => { // eslint-
  ↪ disable-line no-unused-vars
  const res = await instance.get('/layers/search/findByUserId
  ↪ ', { params: { userId } });
  return _.get(res, 'data._embedded.layers');
};
```

1.15. web-gis-front-end/src/utils/data.js

```
import { Set } from 'immutable';

/* eslint-disable no-param-reassign, import/prefer-default-
  ↪ export */
export const getTree = (nodes, parent = {}, tree = []) => {
  if ('layerCategoryId' in parent) {
    return tree;
  }
```

```
  let children;
  if (parent.id === undefined) {
    children = nodes.filter(child => !child.parentId && !('
  ↪ layerCategoryId' in child));
  } else {
    children = nodes.filter(child => child.parentId ===
  ↪ parent.id && !('layerCategoryId' in child));
  }
  if (children.length === 0) {
    children = nodes.filter(child => child.layerCategoryId
  ↪ === parent.id);
  }
```

```
    children.forEach(child => getTree(nodes, child));
  }

  return tree;
};
/* eslint-enable */

export const getSetOfParentIds = ({ parentId },
  ↪ categoriesById) => {
  const setOfIds = new Set();

  if (parentId === null || parentId === undefined) {
    return setOfIds;
  }

  return setOfIds
    .add(parentId)
    .union(getSetOfParentIds(categoriesById[parentId] || {},
  ↪ categoriesById));
};

export const getPropertiesValuesFromLayer = layer =>
  Object.values(layer.properties || {}).reduce((
```

```
  ↪ propertiesValues, propertyGroup) => ({
    ...propertiesValues,
    ...propertyGroup,
  }), {});
```

1.16. web-gis-front-end/src/utils/olUtils.js

```
import { Set } from 'immutable';
import OLMap from 'ol/map';
import OLControl from 'ol/control';
import OLEvents from 'ol/events';
import OLMousePosition from 'ol/control/mouseposition';
import OLCoordinate from 'ol/coordinate';
import OLView from 'ol/view';
import OLImage from 'ol/layer/image';
import OLTile from 'ol/layer/tile';
import OLOSM from 'ol/source/osm';
import OLXYZ from 'ol/source/xyz';
import OLWMTSSource from 'ol/source/wmts';
import OLGeoJSON from 'ol/format/geojson';
import OLKML from 'ol/format/kml';
import OLStyle from 'ol/style/style';
import OLFill from 'ol/style/fill';
import OLStroke from 'ol/style/stroke';
import 'plotty';
import GeoTIFF from 'geotiff';

import { getByKeyDeep } from './common';

export const hexToRgba = (hex, alpha = 1) => {
  const shorthandRegex = /^#?([a-f\d])([a-f\d])([a-f\d])$/i;
  const fullHex = hex.replace(
    shorthandRegex,
    (match, red, green, blue) => red + red + green + green +
  ↪ blue + blue,
  );
```

```
  const result = /^#?([a-f\d]{2})([a-f\d]{2})([a-f\d]{2})$/i.
  ↪ exec(fullHex);
  if (result) {
    return [parseInt(result[1], 16), parseInt(result[2], 16),
  ↪  parseInt(result[3], 16), alpha];
  }

  return null;
};

export const generateOlStyle = (properties, defaultProperties
  ↪ ) => {
  const styleConfig = {};

  const color = getByKeyDeep(properties, 'Color') ||
  ↪ getByKeyDeep(defaultProperties, 'Color');
  if (color) {
    styleConfig.fill = new OLFill({ color: hexToRgba(color,
  ↪ 0.4) });
    styleConfig.stroke = new OLStroke({ color: hexToRgba(
  ↪ color) });
  }

  return new OLStyle(styleConfig);
};
```

```
        default:
          return;
        case 'Opacity':
          olLayer.setOpacity(nextProperties[property]);
          break;
        case 'Color':
          olLayer.setStyle(generateOlStyle(nextProperties,
  ↪ prevProperties));
          break;
        case 'Sea level': {
          const raster = olLayer.getSource();
```

```
            raster.on('beforeoperations', (event) => {
              const { data } = event;
              data.level = nextProperties[property];
            });
            raster.changed();
          }
        }
      }
    });
  };

  export const formatCoordinates = (coords) => {
    const template = `{x} ${coords[0] > 0 ? 'E' : 'W'}, {y} ${
    ↪ coords[1] < 0 ? 'S' : 'N'}`;
    return OLCoordinate.format(coords.map(coord => Math.abs(
```

```
    view: new OLView({
      projection: OLProj.get('EPSG:4326'),
      center: [0, 0],
      zoom: 2,
      minZoom: 1,
    }),
    controls: OLControl.defaults().extend([
      new OLMousePosition({
        coordinateFormat: formatCoordinates,
        projection: 'EPSG:4326',
      }),
    ]),
  });

  export const handleLayersChange = (map, prevLayers,
    ↪ nextLayers) => {
    if (prevLayers.size !== nextLayers.size) {
      if (prevLayers.isSubset(nextLayers)) {
        const newLayerId = new Set(nextLayers.keys()).subtract(
    ↪ new Set(prevLayers.keys())).first();

        map.addLayer(nextLayers.get(newLayerId));
```

```
    } else {
      const oldLayerId = new Set(prevLayers.keys()).subtract(
  ↪ new Set(nextLayers.keys())).first();

      map.removeLayer(prevLayers.get(oldLayerId));
    }
  }
};

export const getOlLayerFromLayer = (layer, properties) => {
  const opacity = getByKeyDeep(properties[layer.id], 'Opacity
  ↪ ') || getByKeyDeep(layer.properties, 'Opacity') || 0.75;
  let olLayer;
  if (layer.layerType === 'vector' && layer.format === '
  ↪ geojson') {
```

```
      style: generateOlStyle(properties[layer.id], layer.
  ↪ properties),
    });
  } else if (layer.layerType === 'heatmap' && layer.format
  ↪ === 'kml') {
    olLayer = new OLHeatmap({
      source: new OLVectorSource({
        url: layer.source,
        format: new OLKML({ extractStyles: false }),
      }),
      opacity,
    });
  } else if (layer.layerType === 'image' && layer.format ===
  ↪ 'xyz') {
    const raster = new OLRaster({
      sources: [new OLXYZ({
        url: layer.source,
        crossOrigin: 'anonymous',
        transition: 0,
      })],
      operation: (pixels, data) => {
        const pixel = pixels[0];
```

```
        if (pixel[3]) {
          const height = -10000 + (((pixel[0] * 256 * 256) +
↪ (pixel[1] * 256) + pixel[2]) * 0.1);
          if (height <= data.level) {
            pixel[0] = 145;
            pixel[1] = 175;
            pixel[2] = 186;
            pixel[3] = 255;
          } else {
            pixel[3] = 0;
          }
        }
        return pixel;
      },
```

```
    olLayer = new OLImage({
      source: raster,
      opacity,
    });
  } else if (layer.layerType === 'tile' && layer.format === '
↪ wmts') {
    olLayer = new OLTile({
      source: new OLWMTSSource({
        url: layer.source,
        tileGrid: new OLWMTSTileGrid({
          extent: [-123.4326739, 36.4002547, -120.7931878,
↪ 38.5263291],
          resolutions: [
            0.009779269924205,
            0.004889634962102,
            0.002444817481051,
            0.001222408740526,
            0.000611204370263,
            0.000305602185131,
          ],
          matrixIds: [0, 1, 2, 3, 4, 5],
        }),
        requestEncoding: 'REST',
```

```
        transition: 0,
      }),
    });

    const datafunctions = {};
    datafunctions.NDVI = (b) => {
      if (b[1] + b[2] + b[3] + b[4] + b[5] === 0) return 10;
      return (b[5] - b[4]) / (b[5] + b[4]);
    };
    datafunctions.EVI = (b) => {
```

```
    datafunctions.SAVI = (b) => {
      if (b[1] + b[2] + b[3] + b[4] + b[5] === 0) return 10;
      /* eslint-disable no-mixed-operators */
      return ((b[5] - b[4]) / (b[5] + b[4] + 0.5)) * 1.5;
      /* eslint-enable */
    };

    /* eslint-disable no-use-before-define */
    wrapGeotiffLayer({ layer: olLayer, dataFunction:
  ↪ datafunctions.EVI });
    /* eslint-enable */
  }

  if (layer.format === 'kml') {
    olLayer.getSource().on('addfeature', (event) => {
      const name = event.feature.get('name');
      const magnitude = parseFloat(name.substr(2));
      event.feature.set('weight', magnitude - 5);
    });
  }

  return olLayer;
};

const wrapGeotiffLayer = ({
  layer, domain = [-1, 1], palette = 'summer', noDataValue =
  ↪ 10, dataFunction = bands => bands[1],
```

```
}) => {
  this.data = (rasters) => {
    const dataArray = [];

    for (let i = 0; i < rasters[0].length; i += 1) {
      const bands = [null];

      rasters.forEach(band => bands.push(band[i]));

      dataArray.push(dataFunction(bands));
    }

    return dataArray;
  };

  this.tiffByUrl = {};

  this.plot = new global.plotty.plot({}); // eslint-disable-
  ↪ line new-cap

  this.fetchTiff = async (url, listener, errorListener) => {
    const { tiffByUrl } = this;
```

```
      } else { // in this case the tiff was already requested
        tiffByUrl[url].listeners.push(listener);
        tiffByUrl[url].errorListeners.push(errorListener);
      }
    } else { // in this case the tiff was not yet requested
      tiffByUrl[url] = {
        rasters: null,
        error: null,
        listeners: [listener],
        errorListeners: [errorListener],
      };
      // send new request
      try {
        const response = await fetch(url);
        const data = await response.arrayBuffer();
```

```
//                      final String kid = JWT.decode(
  ↪ accessToken).getKeyId();
//                      final RSAPublicKey publicKey = (
  ↪ RSAPublicKey) jwkProvider.get(kid).getPublicKey();
//
//                      try {
//                          Algorithm algorithm = Algorithm.
  ↪ RSA256(publicKey);
//                          JWT.require(algorithm).withIssuer(
  ↪ 6 ? "." : Integer.toString(6 - z);
                      final Path levelDirectoryPath =
  ↪ pyramidDirectoryPath.resolve(Paths.get(
  ↪ levelDirResolveString));
                      Path tilePath = levelDirectoryPath.
  ↪ resolve(String.format("SF_%d_%d.tif", x, y));

                      if (Files.exists(tilePath)) {
                          return Files.readAllBytes(tilePath);
                      }

                      tilePath = levelDirectoryPath.resolve(
  ↪ String.format("SF_%01d_%01d.tif", x, y));

                      if (Files.exists(tilePath)) {
                          return Files.readAllBytes(tilePath);
                      }

                      tilePath = levelDirectoryPath.resolve(
  ↪ String.format("SF_%02d_%02d.tif", x, y));

                      if (Files.exists(tilePath)) {
                          return Files.readAllBytes(tilePath);
                      }

                      return Files.readAllBytes(
```

```
  ↪ pyramidDirectoryPath.resolve(Paths.get("./SF_16_16.tif")
  ↪ ));
                    })
                    // Start the server
                    .start();
    }
}
```

1.84. web-gis-tile-service/src/main/java/com/webgis/tileservice/server/Response.java

```
package com.webgis.tileservice.server;

public class Response {
}
```

1.85. web-gis-tile-service/src/main/java/com/webgis/tileservice/server/Handler.java

```
package com.webgis.tileservice.server;

public interface Handler {

    Object handle(Request request, Response response) throws
  ↪ Exception;

}
```

1.86. web-gis-tile-service/src/main/java/com/webgis/tileservice/server/Route.java

```
package com.webgis.tileservice.server;
```

```
 * The Route class represents a single entry in the
  ↪ RouteTable.
 */
```

```
public class Route {
    private final HttpMethod method;
    private final String path;
    private final Handler handler;

    public Route(final HttpMethod method, final String path,
  ↪ final Handler handler) {
        this.method = method;
        this.path = path;
        this.handler = handler;
    }

    public HttpMethod getMethod() {
        return method;
    }

    public String getPath() {
        return path;
```

```
        if (!this.method.equals(method)) {
            return false;
        }

        final String pathWithoutQuery = this.path.split
  ↪ ("\\?")[0];

        if (pathWithoutQuery.contains(":")) {
            List<String> routePathAfterSplit = Arrays.asList(
  ↪ this.path.split("/"));
            List<String> incomingPathAfterSplit = Arrays.
  ↪ asList(path.split("/"));

            return routePathAfterSplit.size() ==
  ↪ incomingPathAfterSplit.size();
        }
```

```
        return pathWithoutQuery.equals(path);
    }
}
```

1.87. web-gis-tile-service/src/main/java/com/webgis/tileservice/server/RouteTable.java

```
package com.webgis.tileservice.server;

import java.util.ArrayList;

import io.netty.handler.codec.http.HttpMethod;

/**
```

```
    }

    public void addRoute(final Route route) {
        this.routes.add(route);
    }

    public Route findRoute(final HttpMethod method, final
  ↪ String path) {
        for (final Route route : routes) {
            if (route.matches(method, path)) {
                return route;
            }
        }

        return null;
    }
}
```

1.88. web-gis-tile-service/src/main/java/com/webgis/tileservice/server/WebServer.java

```
package com.webgis.tileservice.server;

import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.buffer.ByteBuf;
import io.netty.buffer.PooledByteBufAllocator;
import io.netty.buffer.Unpooled;
import io.netty.channel.Channel;
import io.netty.channel.ChannelFutureListener;
import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.epoll.EpollServerSocketChannel;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.SocketChannel;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.handler.codec.http.DefaultFullHttpResponse;
import io.netty.handler.codec.http.DefaultHttpHeaders;
import io.netty.handler.codec.http.FullHttpRequest;
import io.netty.handler.codec.http.FullHttpResponse;
import io.netty.handler.codec.http.HttpHeaderNames;
import io.netty.handler.codec.http.HttpHeaderUtil;
import io.netty.handler.codec.http.HttpMethod;
import io.netty.handler.codec.http.HttpObjectAggregator;
import io.netty.handler.codec.http.HttpRequestDecoder;
import io.netty.handler.codec.http.HttpResponseEncoder;
import io.netty.handler.codec.http.HttpResponseStatus;
import io.netty.handler.codec.http.HttpVersion;

/**
 * The WebServer class is a convenience wrapper around the
   ↪ Netty HTTP server.
 */
```

```
public class WebServer {
    public static final String TYPE_PLAIN = "text/plain;
```

```
    /**
     * Creates a new WebServer.
     */
    public WebServer() {
        this.routeTable = new RouteTable();
        this.port = 4567;
    }


    /**
     * Adds a GET route.
     *
     * @param path The URL path.
     * @param handler The request handler.
     * @return This WebServer.
     */
    public WebServer get(final String path, final Handler
↪ handler) {
        this.routeTable.addRoute(new Route(HttpMethod.GET,
↪ path, handler));
        return this;
    }


    /**
     * Adds a POST route.
     *
     * @param path The URL path.
     * @param handler The request handler.
     * @return This WebServer.
     */
    public WebServer post(final String path, final Handler
↪ handler) {
```

```
        this.routeTable.addRoute(new Route(HttpMethod.POST,
↪ path, handler));
        return this;
    }


    /**
     * Starts the web server.
     *
     * @throws Exception
     */
    public void start() throws Exception {
```

```
    /**
     * Initializes the server, socket, and channel.
     *
     * @param loopGroup The event loop group.
     * @param serverChannelClass The socket channel class.
     * @throws InterruptedException on interruption.
     */
    private void start(
            final EventLoopGroup loopGroup,
            final Class<? extends ServerChannel>
↪ serverChannelClass)
            throws InterruptedException {

        try {
            final InetSocketAddress inet = new
↪ InetSocketAddress(port);

            final ServerBootstrap b = new ServerBootstrap();
            b.option(ChannelOption.SO_BACKLOG, 1024);
            b.option(ChannelOption.SO_REUSEADDR, true);
            b.group(loopGroup).channel(serverChannelClass).
```

```
↪ childHandler(new WebServerInitializer());
            b.option(ChannelOption.MAX_MESSAGES_PER_READ,
↪ Integer.MAX_VALUE);
            b.childOption(ChannelOption.ALLOCATOR, new
↪ PooledByteBufAllocator(true));
            b.childOption(ChannelOption.SO_REUSEADDR, true);
            b.childOption(ChannelOption.MAX_MESSAGES_PER_READ
↪ , Integer.MAX_VALUE);

            final Channel ch = b.bind(inet).sync().channel();
            ch.closeFuture().sync();

        } finally {
```

```
    private class WebServerInitializer extends
↪ ChannelInitializer<SocketChannel> {

        /**
         * Initializes the channel pipeline with the HTTP
↪ response handlers.
         *
         * @param ch The Channel which was registered.
         */
        @Override
        public void initChannel(SocketChannel ch) throws
↪ Exception {
            final ChannelPipeline p = ch.pipeline();
            p.addLast("decoder", new HttpRequestDecoder(4096,
↪  8192, 8192, false));
            p.addLast("aggregator", new HttpObjectAggregator
↪ (100 * 1024 * 1024));
            p.addLast("encoder", new HttpResponseEncoder());
            p.addLast("handler", new WebServerHandler());
        }
    }
```

```
  /**
   * The Handler class handles all inbound channel messages
↪ .
   */
  private class WebServerHandler extends
↪ SimpleChannelInboundHandler<Object> {

      /**
       * Handles a new message.
       *
       * @param ctx The channel context.
       * @param msg The HTTP request message.
       */
```

```
↪   msg;

            if (HttpHeaderUtil.is100ContinueExpected(request)
↪ ) {
                send100Continue(ctx);
            }

            final HttpMethod method = request.method();
            final String uri = request.uri();

            final Route route = WebServer.this.routeTable.
↪ findRoute(method, uri);
            if (route == null) {
                writeNotFound(ctx, request);
                return;
            }

            try {
                final Request requestWrapper = new Request(
↪ request, route);
                final Object obj = route.getHandler().handle(
```

```
↪ requestWrapper, null);
                if (obj.getClass().isArray()) {
                    final byte[] content = (byte[])obj;
                    writeResponse(ctx, request,
↪ HttpResponseStatus.OK, "image/tiff", content);
                } else {
                    final String content = obj == null ? "" :
↪  obj.toString();
                    writeResponse(ctx, request,
↪ HttpResponseStatus.OK, TYPE_PLAIN, content);
                }
            } catch (final Exception ex) {
```

```
         * @param ctx The channel context.
         * @param cause The exception.
         */
        @Override
        public void exceptionCaught(final
↪ ChannelHandlerContext ctx, final Throwable cause) {
            ctx.close();
        }


        /**
         * Handles read complete event.  Flushes the context.
         *
         * @param ctx The channel context.
         */
        @Override
        public void channelReadComplete(final
↪ ChannelHandlerContext ctx) {
            ctx.flush();
        }
    }
```

```
  /**
   * Writes a 404 Not Found response.
   *
   * @param ctx The channel context.
   * @param request The HTTP request.
   */
  private static void writeNotFound(
          final ChannelHandlerContext ctx,
          final FullHttpRequest request) {

      writeErrorResponse(ctx, request, HttpResponseStatus.
↪ NOT_FOUND);
  }
```

```
  private static void writeInternalServerError(
          final ChannelHandlerContext ctx,
          final FullHttpRequest request) {

      writeErrorResponse(ctx, request, HttpResponseStatus.
↪ INTERNAL_SERVER_ERROR);
  }


  /**
   * Writes a HTTP error response.
   *
   * @param ctx The channel context.
   * @param request The HTTP request.
   * @param status The error status.
   */
  private static void writeErrorResponse(
          final ChannelHandlerContext ctx,
          final FullHttpRequest request,
          final HttpResponseStatus status) {
```

```
        writeResponse(ctx, request, status, TYPE_PLAIN,
↪ status.reasonPhrase().toString());
    }


    /**
     * Writes a HTTP response.
     *
     * @param ctx The channel context.
     * @param request The HTTP request.
     * @param status The HTTP status code.
     * @param contentType The response content type.
     * @param content The response content.
     */
    private static void writeResponse(
            final ChannelHandlerContext ctx,
            final FullHttpRequest request,
            final HttpResponseStatus status,
            final CharSequence contentType,
            final String content) {
```

[redacted]

```
    private static void writeResponse(
            final ChannelHandlerContext ctx,
            final FullHttpRequest request,
            final HttpResponseStatus status,
            final CharSequence contentType,
            final byte[] content) {
        final ByteBuf entity = Unpooled.wrappedBuffer(content
↪ );
        writeResponse(ctx, request, status, entity,
↪ contentType, content.length);
    }
```

```
/**
 * Writes a HTTP response.
 *
 * @param ctx The channel context.
 * @param request The HTTP request.
 * @param status The HTTP status code.
 * @param buf The response content buffer.
 * @param contentType The response content type.
 * @param contentLength The response content length;
```

```
        // Decide whether to close the connection or not.
        final boolean keepAlive = HttpHeaderUtil.isKeepAlive(
↪ request);

        // Build the response object.
        final FullHttpResponse response = new
↪ DefaultFullHttpResponse(
                HttpVersion.HTTP_1_1,
                status,
                buf,
                false);

        final ZonedDateTime dateTime = ZonedDateTime.now();
        final DateTimeFormatter formatter = DateTimeFormatter
↪ .RFC_1123_DATE_TIME;

        final DefaultHttpHeaders headers = (
↪ DefaultHttpHeaders) response.headers();
        headers.set(HttpHeaderNames.SERVER, SERVER_NAME);
        headers.set(HttpHeaderNames.DATE, dateTime.format(
↪ formatter));
        headers.set(HttpHeaderNames.CONTENT_TYPE, contentType
↪ );
        headers.set(HttpHeaderNames.CONTENT_LENGTH, Integer.
↪ toString(contentLength));
```

```
        headers.set(HttpHeaderNames.
↪ ACCESS_CONTROL_ALLOW_ORIGIN, "*");

        // Close the non-keep-alive connection after the
↪ write operation is done.
        if (!keepAlive) {
            ctx.writeAndFlush(response).addListener(
↪ ChannelFutureListener.CLOSE);
        } else {
            ctx.writeAndFlush(response, ctx.voidPromise());
        }
    }


    /**
```

```
                    HttpResponseStatus.CONTINUE));
    }
}
```

1.89. web-gis-tile-service/src/main/java/com/webgis/tileservice/server/Request.java

```
package com.webgis.tileservice.server;

import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
import java.util.OptionalInt;
import java.util.stream.IntStream;

import io.netty.handler.codec.http.FullHttpRequest;

/**
 * The Request class provides convenience helpers to the
 ↪ underyling
```

```
 * HTTP Request.
 */
public class Request {
    private final FullHttpRequest request;
    private final Route route;

    /**
     * Creates a new Request.
     *
     * @param request The Netty HTTP request.
     */
    public Request(final FullHttpRequest request, final Route
↪   route) {
        this.request = request;
        this.route = route;
    }
```

```
↪ UTF_8);
    }

    public String uri() {
        return request.uri();
    }

    public String header(String name) {
        final String header = (String)request.headers().get(
↪ name);
        if (header != null) {
            return header;
        }
        return "";
    }

    public String param(String name) {
        List<String> routePathAfterSplit = Arrays.asList(this
↪ .route.getPath().split("/"));
```

```
        List<String> incomingPathAfterSplit = Arrays.asList(
    ↪ request.uri().split("\\?")[0].split("/"));

        OptionalInt indexOfParam = IntStream.range(0,
    ↪ routePathAfterSplit.size())
```

```
        if (!indexOfParam.isPresent()) {
            return "";
        }

        return incomingPathAfterSplit.get(indexOfParam.
    ↪ getAsInt());
    }
}
```

1.90. web-gis-tile-service/build.gradle

```
buildscript {
    ext {
        springBootVersion = '2.0.0.RELEASE'
    }
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-
    ↪ gradle-plugin:${springBootVersion}")
    }
}

apply plugin: 'java'
apply plugin: 'idea'
apply plugin: 'org.springframework.boot'
apply plugin: 'io.spring.dependency-management'

group = 'com.web-gis'
```

```
version = '0.0.1-SNAPSHOT'
sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    compile('org.projectlombok:lombok:1.16.20')
    compile('org.springframework.boot:spring-boot-starter-web
  ↪ ')
    compile("org.springframework.boot:spring-boot-starter-
  ↪ data-mongodb")
    compile('org.springframework.boot:spring-boot-starter-
  ↪ security')
```

```
    compile('org.json:json:20171018')
    compile('com.auth0:auth0-spring-security-api:1.0.0-rc.3')
    testCompile('org.springframework.boot:spring-boot-starter
  ↪ -test')
}
```

1.91. web-gis-tile-service/src/test/java/com/webgis/backend/DemoApplicationTests.java

```
package com.webgis.backend;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class DemoApplicationTests {
```

```
        @Test
        public void contextLoads() {
        }

}
```

1.92. web-gis-tile-service/src/main/java/com/webgis/backend/DemoApplication.java

```
package com.webgis.backend;

@SpringBootApplication
@ComponentScan(basePackages = "com.webgis.backend")
@EnableAutoConfiguration
@PropertySources({
        @PropertySource("classpath:application.properties"),
        @PropertySource("classpath:auth0.properties")
})
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

1.93. web-gis-tile-service/src/main/java/com/webgis/backend/configurations/SecurityConfig.java

```
package com.webgis.backend.configurations;

import com.auth0.spring.security.api.JwtWebSecurityConfigurer
  ↪ ;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
```

```
import org.springframework.http.HttpMethod;
import org.springframework.security.config.annotation.web.
  ↪ builders.HttpSecurity;
import org.springframework.security.config.annotation.web.
  ↪ configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.
  ↪ configuration.WebSecurityConfigurerAdapter;
```

```
    @Value(value = "${auth0.issuer}")
    private String issuer;

    @Override
    protected void configure(HttpSecurity http) throws
  ↪ Exception {
        JwtWebSecurityConfigurer
                .forRS256(apiAudience, issuer)
                .configure(http)
                .cors().and()
                .authorizeRequests()
                .antMatchers(HttpMethod.GET, "/api/ping").
  ↪ fullyAuthenticated()
                .antMatchers("/api/projects").
  ↪ fullyAuthenticated()
                .antMatchers("/api/layers").
  ↪ fullyAuthenticated()
                .antMatchers("/api/layerCategories").
  ↪ fullyAuthenticated();
    }
}
```

1.94. web-gis-tile-service/src/main/java/com/webgis/backend/configurations/RepositoryConfig.java

```
package com.webgis.backend.configurations;

import com.webgis.backend.models.Layer;
```

```
import com.webgis.backend.models.LayerCategory;
import com.webgis.backend.models.Project;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.rest.core.config.
  ↪ RepositoryRestConfiguration;
import org.springframework.data.rest.webmvc.config.
  ↪ RepositoryRestConfigurerAdapter;

@Configuration
public class RepositoryConfig extends
  ↪ RepositoryRestConfigurerAdapter {
    @Override
    public void configureRepositoryRestConfiguration(
  ↪ RepositoryRestConfiguration config) {
        config.exposeIdsFor(Project.class);
        config.exposeIdsFor(Layer.class);
        config.exposeIdsFor(LayerCategory.class);
    }
}
```

1.95. web-gis-tile-service/src/main/java/com/webgis/backend/repositories/LayerRepository.java

```
package com.webgis.backend.repositories;

import org.springframework.web.bind.annotation.CrossOrigin;

import java.util.List;

@CrossOrigin
@RepositoryRestResource(path = "layers")
public interface LayerRepository extends MongoRepository<
  ↪ Layer, String> {
    @Query("{ $or: [{ type: 'public' }, { ownerId: ?0 }, {
  ↪ sharedWithIds: ?0 }] }")
    List<Layer> findByUserId(@Param("userId") String userId);
```

```
}
```

1.96. web-gis-tile-service/src/main/java/com/webgis/backend/repositories/ ProjectRepository.java

```
package com.webgis.backend.repositories;

import com.webgis.backend.models.Project;
import org.springframework.data.mongodb.repository.
  ↪ MongoRepository;
import org.springframework.data.repository.query.Param;
```

```
public interface ProjectRepository extends MongoRepository<
  ↪ Project, String> {
    List<Project> findByOwnerId(@Param("ownerId") String
  ↪ ownerId);
}
```

1.97. web-gis-tile-service/src/main/java/com/webgis/backend/repositories/ LayerCategoryRepository.java

```
package com.webgis.backend.repositories;

import com.webgis.backend.models.LayerCategory;
import org.springframework.data.mongodb.repository.
  ↪ MongoRepository;
import org.springframework.data.rest.core.annotation.
  ↪ RepositoryRestResource;
import org.springframework.web.bind.annotation.CrossOrigin;

@CrossOrigin
@RepositoryRestResource(path = "layerCategories")
public interface LayerCategoryRepository extends
  ↪ MongoRepository<LayerCategory, String> {}
```

1.98. web-gis-tile-service/src/main/java/com/webgis/backend/models/LayerCategory.java

```
package com.webgis.backend.models;

import lombok.Builder;
import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document
  ↪ ;

import java.util.List;

@D
}
```

1.99. web-gis-tile-service/src/main/java/com/webgis/backend/models/Project.java

```
package com.webgis.backend.models;

import lombok.Builder;
import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document
  ↪ ;

import java.util.List;

@Data
@Builder
@Document(collection = "projects")
public class Project {
    @Id
    private String id;
```

```
    private String name;
    private String ownerId;
    private List<String> layersIds;
}
```

1.100. web-gis-tile-service/src/main/java/com/webgis/backend/models/Layer.java

```
package com.webgis.backend.models;

import lombok.Builder;
import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document
  ↪ ;

import java.util.List;

@Data
@Builder
@Document(collection = "layers")
```

```
    private List<String> sharedWithIds;
    private List<String> projectsIds;

    private String layerCategoryId;
}
```

1.101. web-gis-tile-service/src/main/java/com/webgis/backend/controllers/APIController.java

```
package com.webgis.backend.controllers;

import org.json.JSONObject;
import org.springframework.stereotype.Component;
```

```
import org.springframework.web.bind.annotation.*;

@RestController
@Component
@RequestMapping(path = "/api")
public class APIController {
    @RequestMapping(value = "/ping", method = RequestMethod.
  ↪ GET, produces = "application/json")
    @ResponseBody
    @CrossOrigin
    public String privateEndpoint() {
        return new JSONObject()
                .put("message", "pong")
                .toString();
    }
}
```

Лист регистрации изменений

Номера листов (страниц)					Всего листов (страниц) в докум	№ документа	Входящий № сопроводительного докум и дата	Подп	Дата
Изм	изменённых	заменённых	новых	аннулированных					