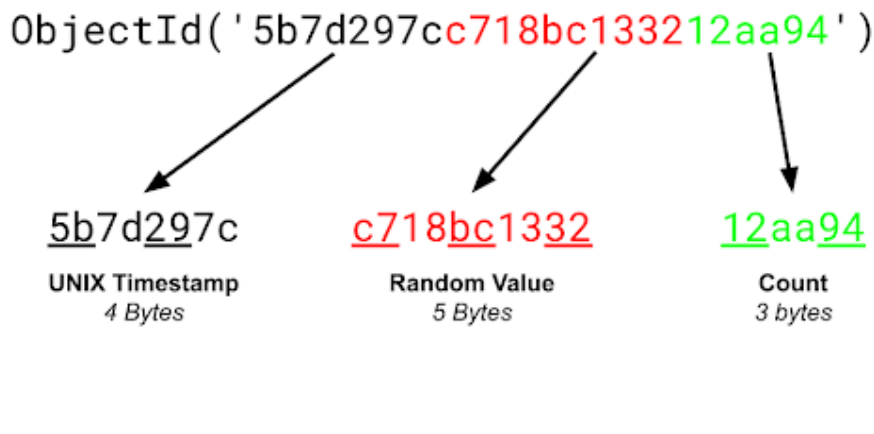
A. mongodb atlas 설정

- 참고 : https://velog.io/@yeonlisa/MongoDB-Atlas-Cluster-생성하기
- my mongodb atlas
  - login : google id
  - mongodb 접속

```
mongodb+srv://iwbaek:iw&100410-
@cluster0.g4kupuc.mongodb.net/?retryWrites=true&w=majority
```

MongDB ObjectId Structure



In [2]:
```
# 강의자료 원본 : https://www.youtube.com/watch?v=7CqJlxBYj-M
from IPython.display import YouTubeVideo
YouTubeVideo('7CqJlxBYj-M')
```

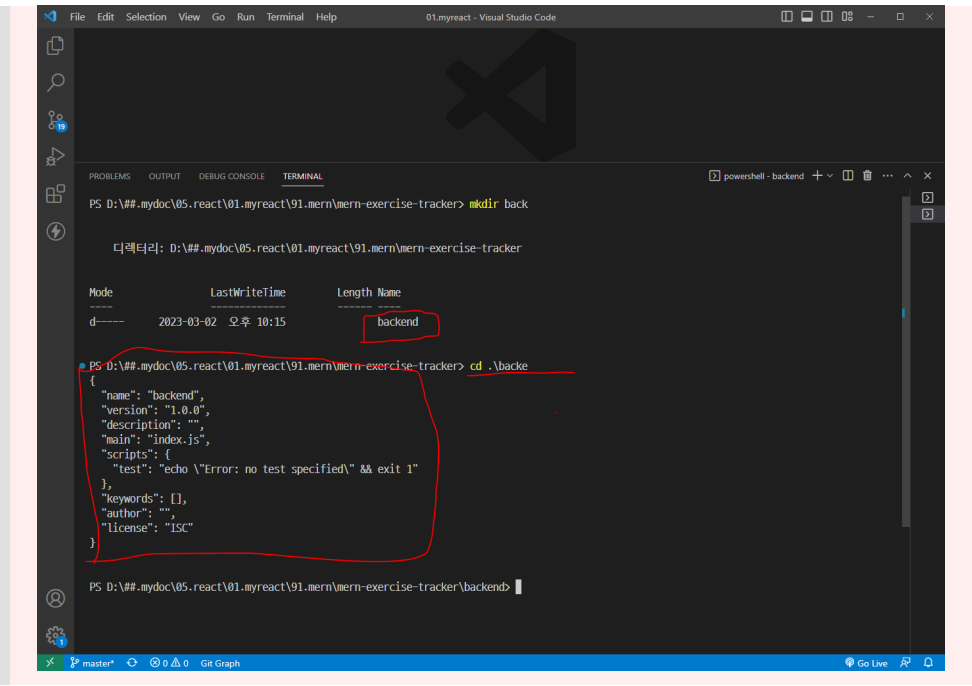
Out[2]:

B. backend

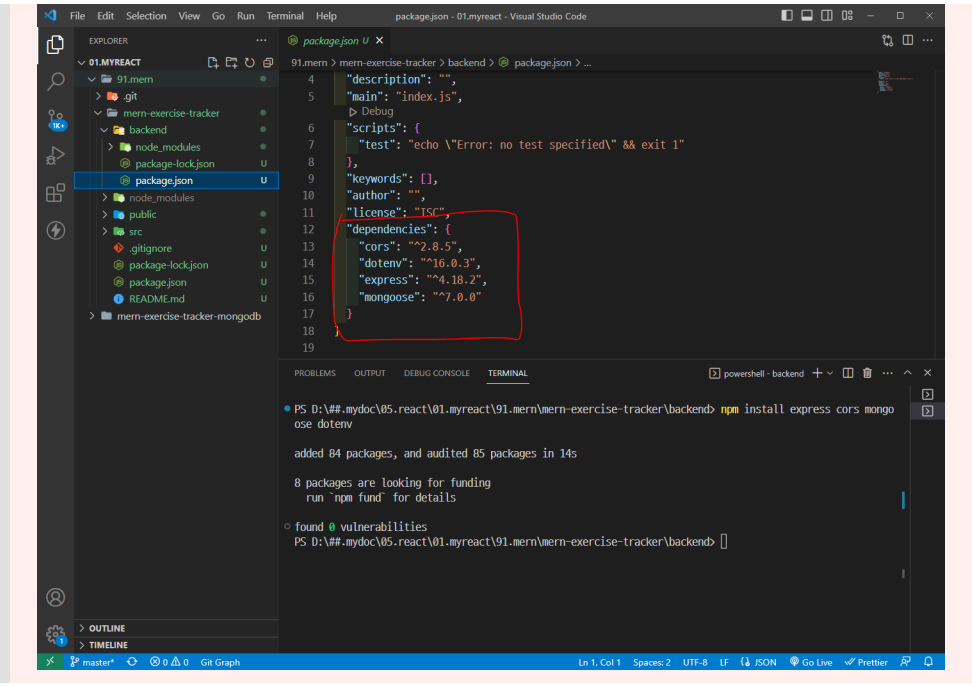
1. 환경설정

1. 폴더생성 : `91.mern`
2. 앱생성 : `npx create-react-app .`
3. 폴더생성 : backend

- mkdir backend
- cd backend
- npm init -y
  - y 라는 속성(yes의미) 을 이용하면 default값으로 설정된 package.json을 만들겠다라는 의미



- npm install express cors mongoose dotenv



- npm install -g nodemon

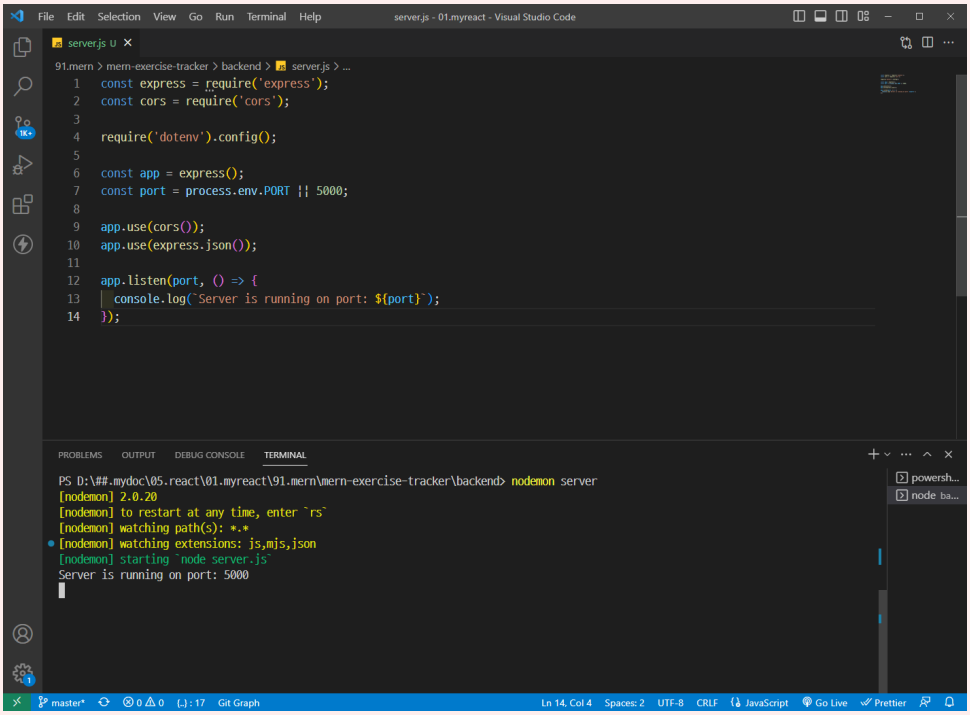
2. backend program

2.1. server.js

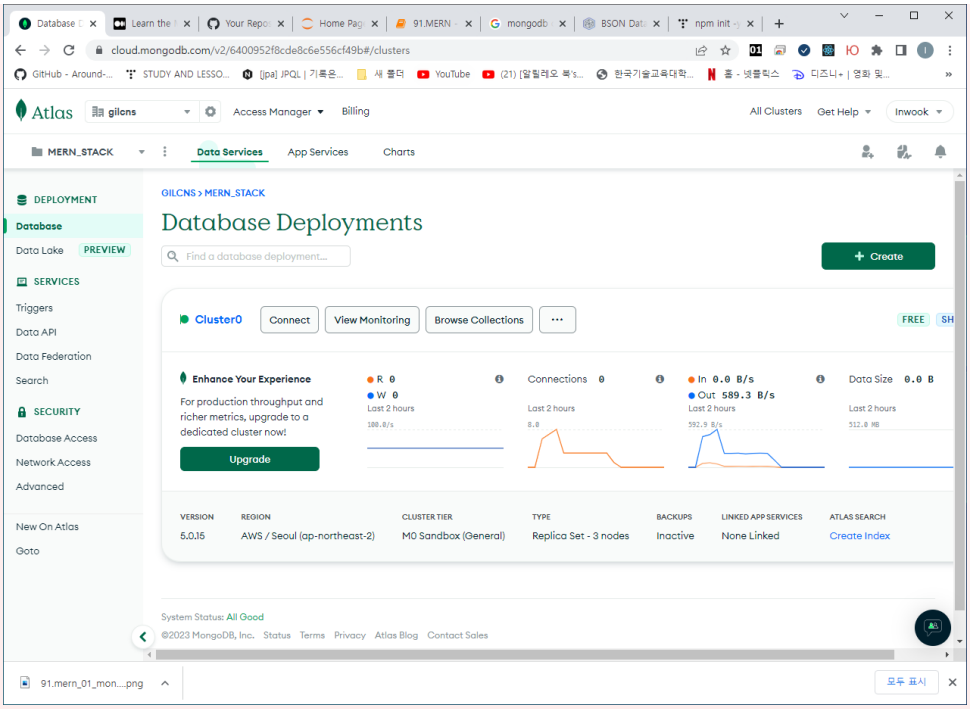
```
backend/server.js
const express = require('express');
const cors = require('cors');

require('dotenv').config();

const app = express();
const port = process.env.PORT || 5000;

app.use(cors());
app.use(express.json());

app.listen(port, () => {
  console.log(`Server is running on port: ${port}`);
});
```

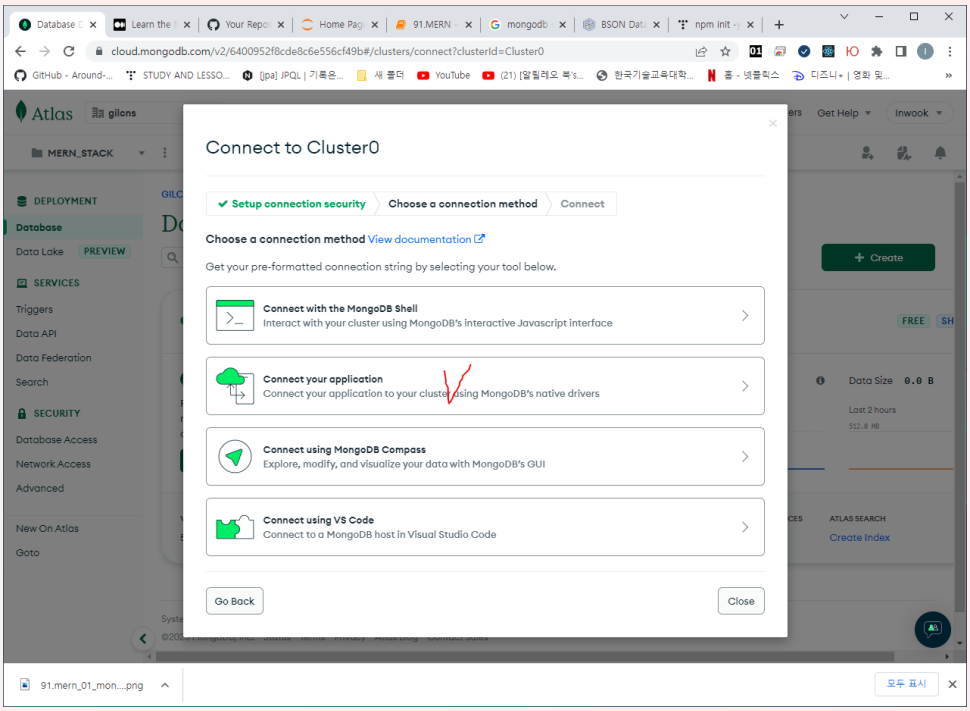
2.2 server start

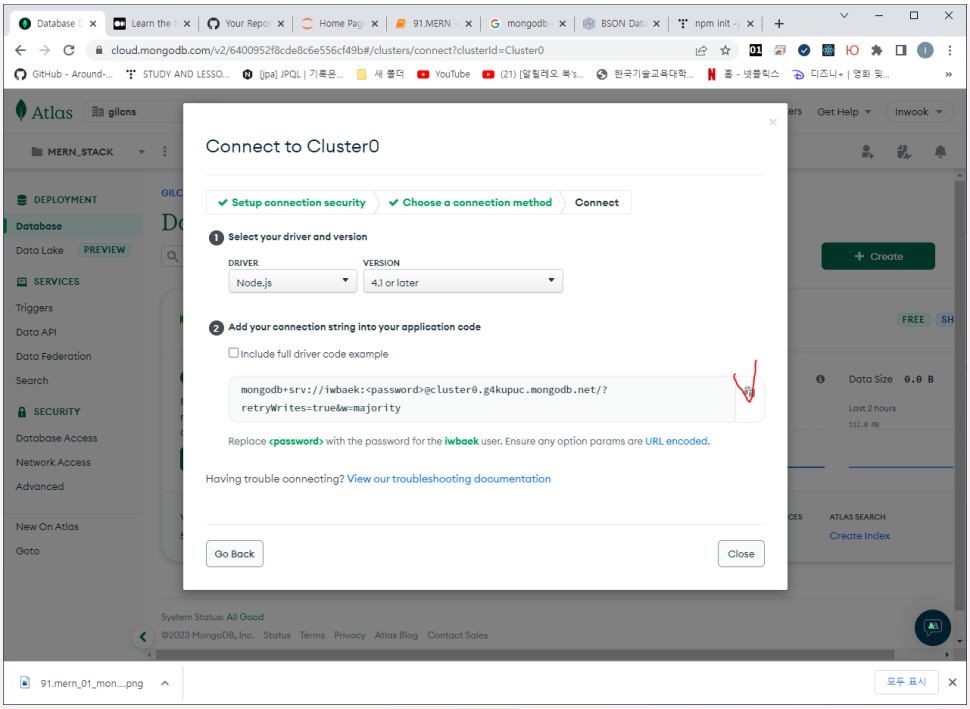
```
nodemon server
```



2.3 mongdb connect


cloud.mongodb.com/v2/6400952f8cde8c6e556cf49b#/clusters
Atlas
gilcns
Access Manager
Billing
All Clusters
Get Help
Inwook
MERN_STACK
Data Services
App Services
Charts
DEPLOYMENT
Database
Data Lake
PREVIEW
SERVICES
Triggers
Data API
Data Federation
Search
SECURITY
Database Access
Network Access
Advanced
New On Atlas
Goto
GILCNS > MERN_STACK
Database Deployments
Find a database deployment...
Create
Cluster0
Connect
View Monitoring
Browse Collections
Enhance Your Experience
For production throughput and richer metrics, upgrade to a dedicated cluster now!
Upgrade
Connections 0
In 0.0 B/s
Out 589.3 B/s
Data Size 0.0 B
Last 2 hours
VERSION
REGION
CLUSTER TIER
TYPE
BACKUPS
LINKED APP SERVICES
ATLAS SEARCH
5.0.15
AWS / Seoul (ap-northeast-2)
M0 Sandbox (General)
Replica Set - 3 nodes
Inactive
None Linked
Create Index
System Status: All Good
©2023 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales
91.mern_01_mon....png
모두 표시


cloud.mongodb.com/v2/6400952f8cde8c6e556cf49b#/clusters/connect?clusterId=Cluster0
Connect to Cluster0
Setup connection security
Choose a connection method
Connect
Choose a connection method View documentation
Get your pre-formatted connection string by selecting your tool below.
Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface
Connect your application
Connect your application to your cluster using MongoDB's native drivers
Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI
Connect using VS Code
Connect to a MongoDB host in Visual Studio Code
Go Back
Close
91.mern_01_mon....png
모두 표시



backend/.env

```
ATLAS_URI=mongodb+srv://iwbaek:
<password>@cluster0.g4kupuc.mongodb.net/?
retryWrites=true&w=majority
```

MongoDB 와 mongoose의 차이점

MongoDB는 로우 레벨의 드라이버 역할을 하는 모듈이다.
mongoose는 ODB(Object Document Modeling)도구이다. 즉, 관계 모델링 툴이라 생각하면 된다.

이처럼 도큐에서는 MongoDB 드라이버를 사용하여 로우레벨에서 개발이 가능하도록 도큐를 제공하고 있다. 하지만, 다양한 기능과 데이터모델을 이용하기 위해서는 mongoose를 이용하여 개발을 하는 것이 편리하다. 또한 MongoDB 드라이버를 사용하게 되면 매번 connection을 확인하고 재접속을 해야되는 불편함이 있다. (코드가 매우 더러워진다.) 그리고 여기서 신기한 점은 mongoose의 드라이버와 MongoDB의 드라이버는 useUnifiedTopology: true 옵션값을 통하여 같은 소켓에서 데이터가 왔다갔다 할 수 있다는 점이다.

backend/server.js

backend/server.js

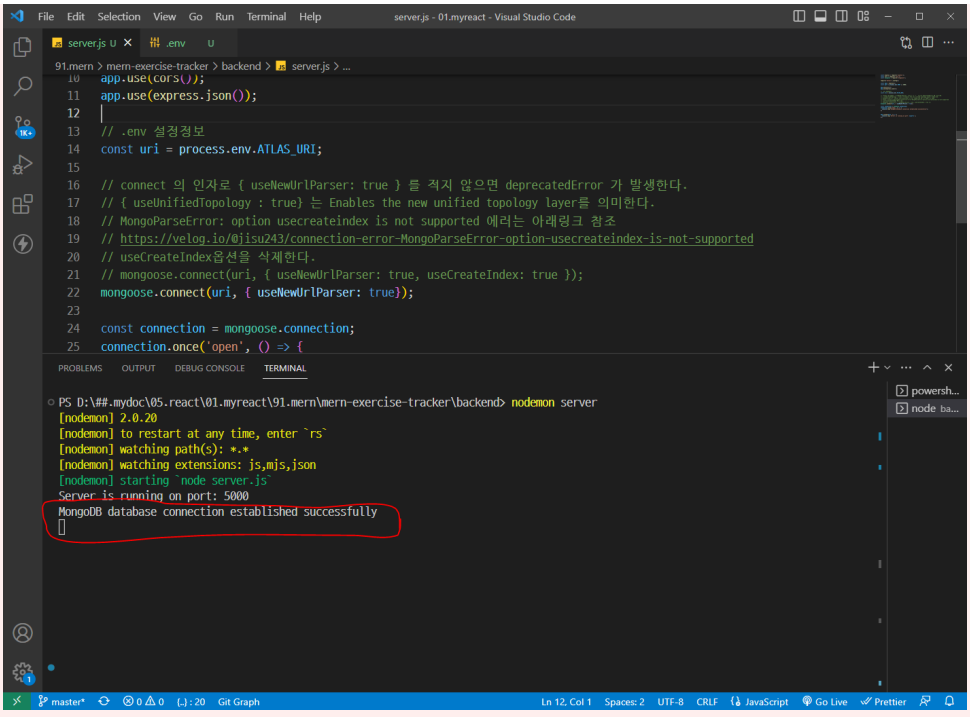
```
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');

require('dotenv').config();

const app = express();
const port = process.env.PORT || 5000;

app.use(cors());
app.use(express.json());

// .env 설정정보
```

```
const uri = process.env.ATLAS_URI;

// connect 의 인자로 { useNewUrlParser: true } 를 적지 않으면 deprecatedError 가 발생한다.
// { useUnifiedTopology : true} 는 Enables the new unified topology layer를 의미한다.
// MongoParseError: option usecreateindex is not supported 에러는 아래링크 참조
// https://velog.io/@jisu243/connection-error-MongoParseError-option-usecreateindex-is-not-supported
// useCreateIndex옵션을 삭제한다.
// mongoose.connect(uri, { useNewUrlParser: true, useCreateIndex: true });

const connection = mongoose.connection;
connection.once('open', () => {
  console.log("MongoDB database connection established successfully");
})


app.listen(port, () => {
  console.log(`Server is running on port: ${port}`);
});
```

2.4 restart express server

```
nodemon server
```

- MongoParseError: option usecreateindex is not supported 에러는 아래링크 참조
- https://velog.io/@jisu243/connection-error-MongoParseError-option-usecreateindex-is-not-supported
- useCreateIndex옵션을 삭제한다.



2.5 mongodb models 작성

1. 폴더생성 : backend/models
2. model작성

User model

user.model.js

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const userSchema = new Schema({
  username: {
    type: String,
    required: true,
    unique: true,
    trim: true,
    minlength: 3
  },
}, {
  timestamps: true,
});

const User = mongoose.model('User', userSchema);

module.exports = User;
```

Exercise model

exercise.model.js

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const exerciseSchema = new Schema({
  username: { type: String, required: true },
  description: { type: String, required: true },
  duration: { type: Number, required: true },
  date: { type: Date, required: true },
}, {
  timestamps: true,
});

const Exercise = mongoose.model('Exercise', exerciseSchema);

module.exports = Exercise;
```

2.6 routes 작성

1. 폴더생성 : backend/routes
2. server.js 수정
3. route 작성

users.js

```
users.js
const router = require('express').Router();
let User = require('../models/user.model');

router.route('/').get((req, res) => {
  User.find()
    .then(users => res.json(users))
    .catch(err => res.status(400).json('Error: ' + err));
});

router.route('/add').post((req, res) => {
  const username = req.body.username;

  const newUser = new User({username});

  newUser.save()
    .then(() => res.json('User added!'))
    .catch(err => res.status(400).json('Error: ' + err));
});

module.exports = router;
```

exercises.js

```
exercises.js
const router = require('express').Router();
let Exercise = require('../models/exercise.model');

router.route('/').get((req, res) => {
  Exercise.find()
    .then(exercises => res.json(exercises))
    .catch(err => res.status(400).json('Error: ' + err));
});

router.route('/add').post((req, res) => {
  const username = req.body.username;
  const description = req.body.description;
  const duration = Number(req.body.duration);
  const date = Date.parse(req.body.date);

  const newExercise = new Exercise({
    username,
    description,
    duration,
    date,
  });

  newExercise.save()
  .then(() => res.json('Exercise added!'))
  .catch(err => res.status(400).json('Error: ' + err));
});

router.route('/:id').get((req, res) => {
  Exercise.findById(req.params.id)
    .then(exercise => res.json(exercise))
    .catch(err => res.status(400).json('Error: ' + err));
});
```

```
router.route('/:id').delete((req, res) => {
  Exercise.findByIdAndDelete(req.params.id)
    .then(() => res.json('Exercise deleted.'))
    .catch(err => res.status(400).json('Error: ' + err));
});

router.route('/update/:id').post((req, res) => {
  Exercise.findById(req.params.id)
    .then(exercise => {
      exercise.username = req.body.username;
      exercise.description = req.body.description;
      exercise.duration = Number(req.body.duration);
      exercise.date = Date.parse(req.body.date);

      exercise.save()
        .then(() => res.json('Exercise updated!'))
        .catch(err => res.status(400).json('Error: ' + err));
    })
    .catch(err => res.status(400).json('Error: ' + err));
});

module.exports = router;
```

1. server.js 수정

server.js

```
server.js
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');

require('dotenv').config();

const app = express();
const port = process.env.PORT || 5000;

app.use(cors());
app.use(express.json());

// .env 설정정보
const uri = process.env.ATLAS_URI;

// connect 의 인자로 { useNewUrlParser: true } 를 적지 않으면
deprecatedError 가 발생한다.
// { useUnifiedTopology : true} 는 Enables the new unified
topology layer를 의미한다.
// MongoParseError: option usecreateindex is not supported 에러는
아래링크 참조
// https://velog.io/@jisu243/connection-error-MongoParseError-
option-usecreateindex-is-not-supported
// useCreateIndex옵션을 삭제한다.
// mongoose.connect(uri, { useNewUrlParser: true, useCreateIndex:
true });
mongoose.connect(uri, { useNewUrlParser: true});
```

```
const connection = mongoose.connection;
connection.once('open', () => {
  console.log("MongoDB database connection established
successfully");
})

const exercisesRouter = require('./routes/exercises');
const usersRouter = require('./routes/users');

app.use('/exercises', exercisesRouter);
app.use('/users', usersRouter);

app.listen(port, () => {
  console.log(`Server is running on port: ${port}`);
});
```

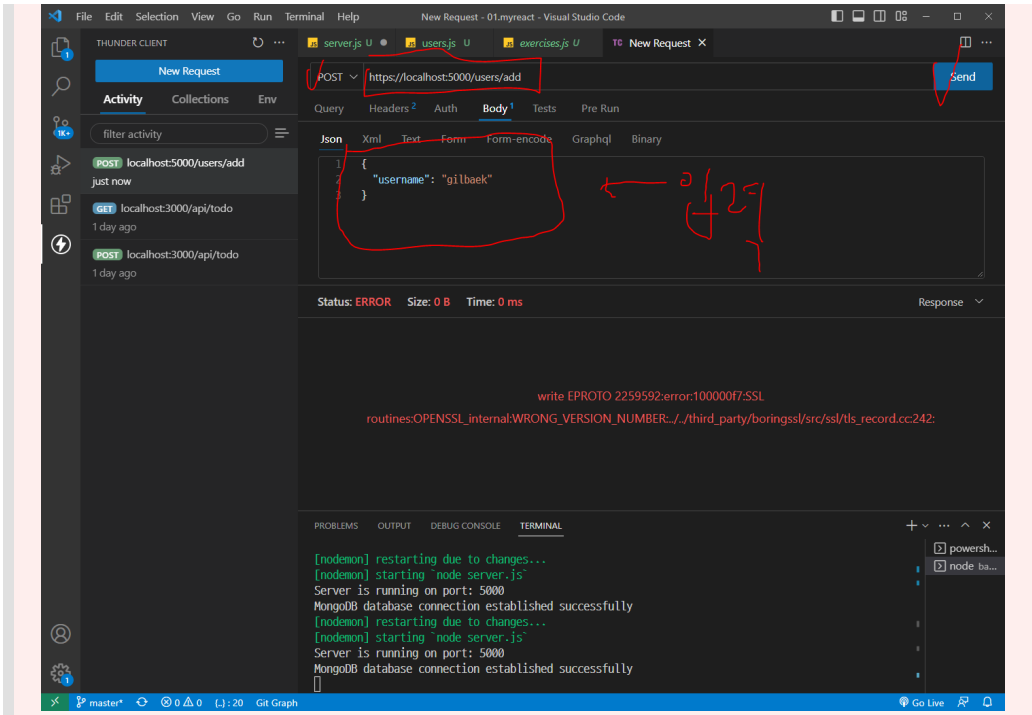
2.7 Rest API 테스트 - users

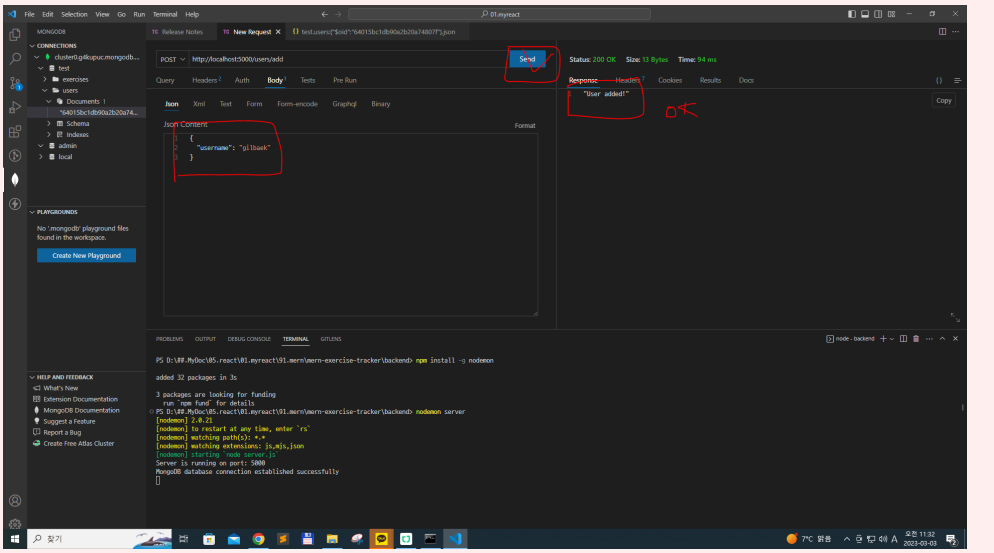
- postman or insomnia를 이용해도 되지만 `vscode 확장팩 Thunder Client` 사용

2.7.1 post

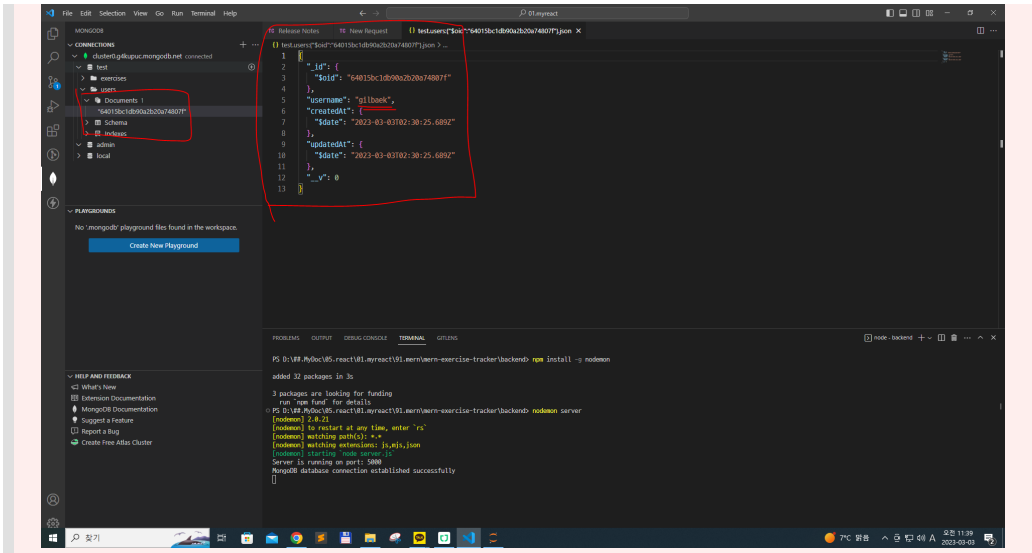
- `http://localhost:5000/users/add`

`Thunder Client` post 처리 및 확인

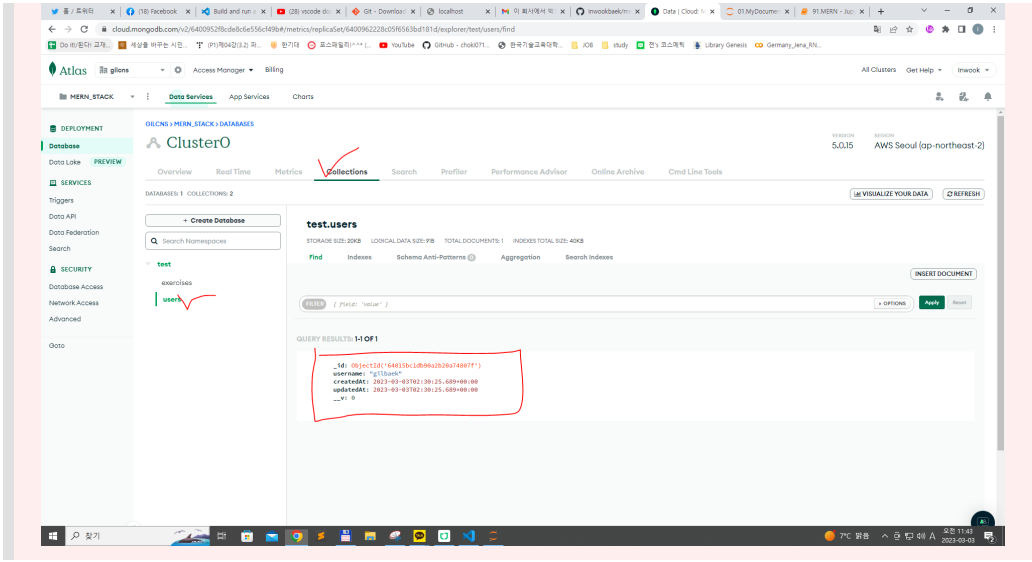




mongodb for vscode : vscode 확장팩 mongodb 사용

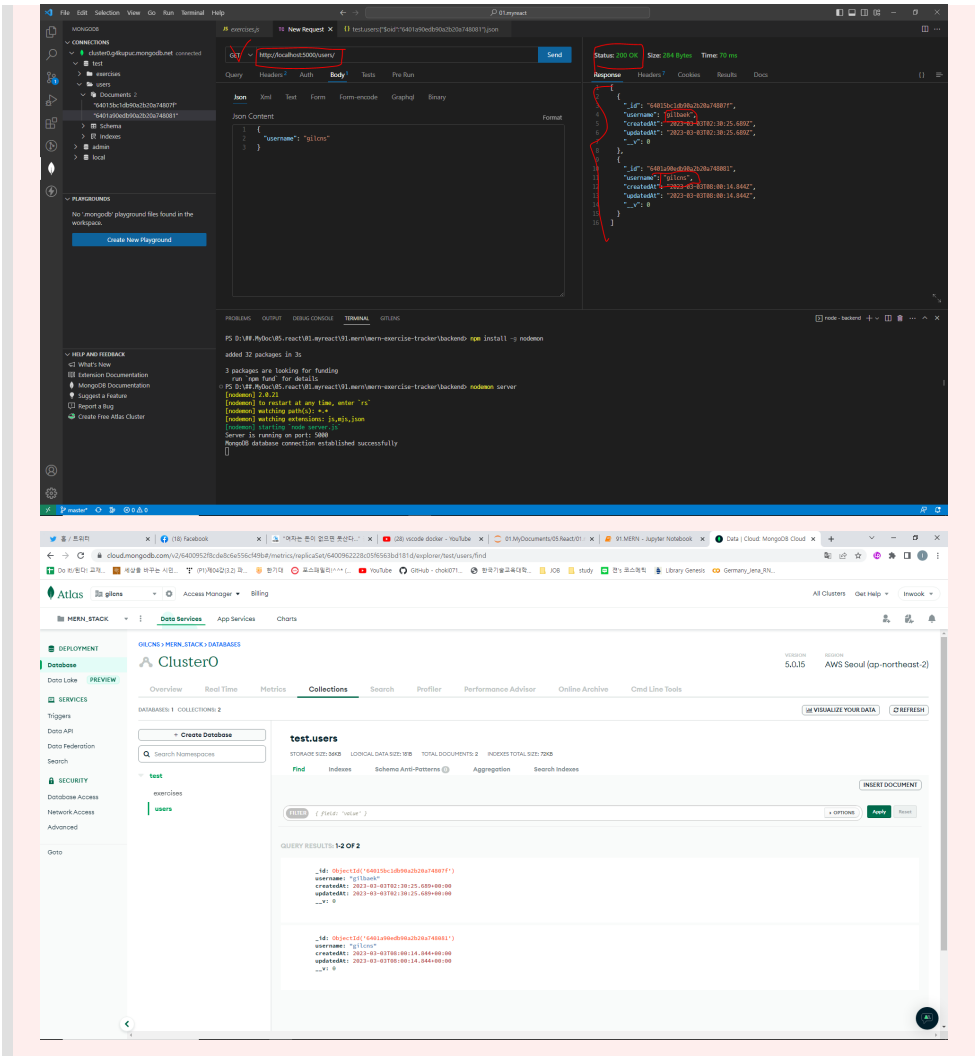


mongodb atlas



2.7.2 get

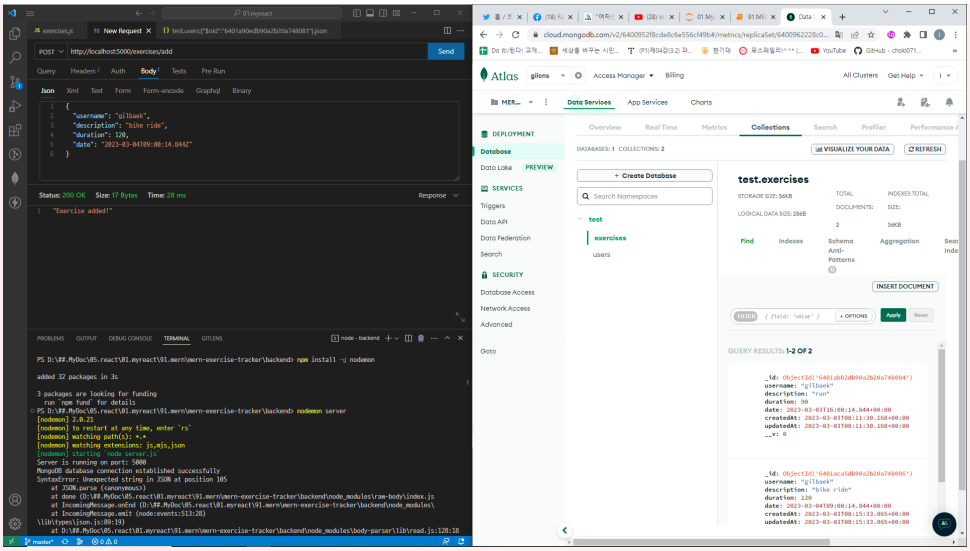
- `http://localhost:5000/users`



2.8 Rest API 테스트 - excercies

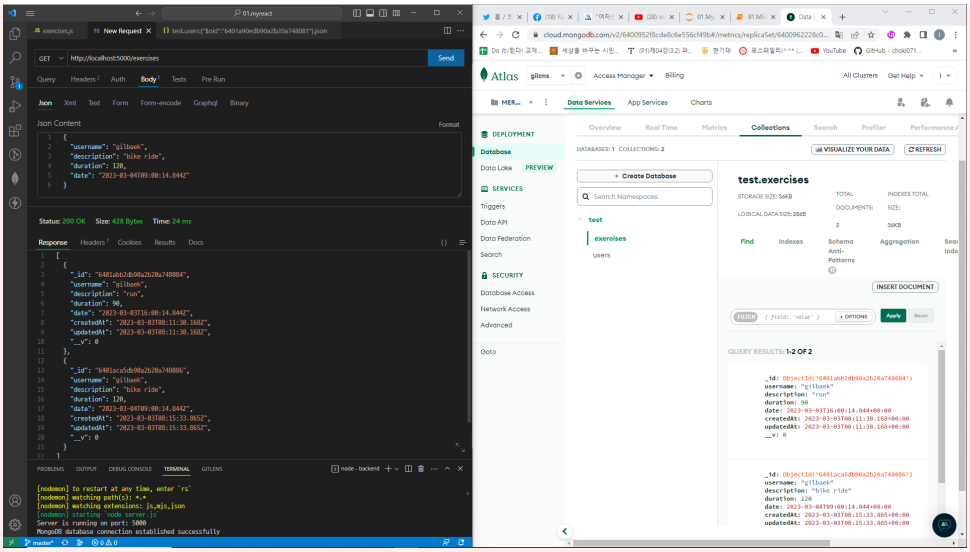
2.8.1 post

- `http://localhost:5000/exercises/add`



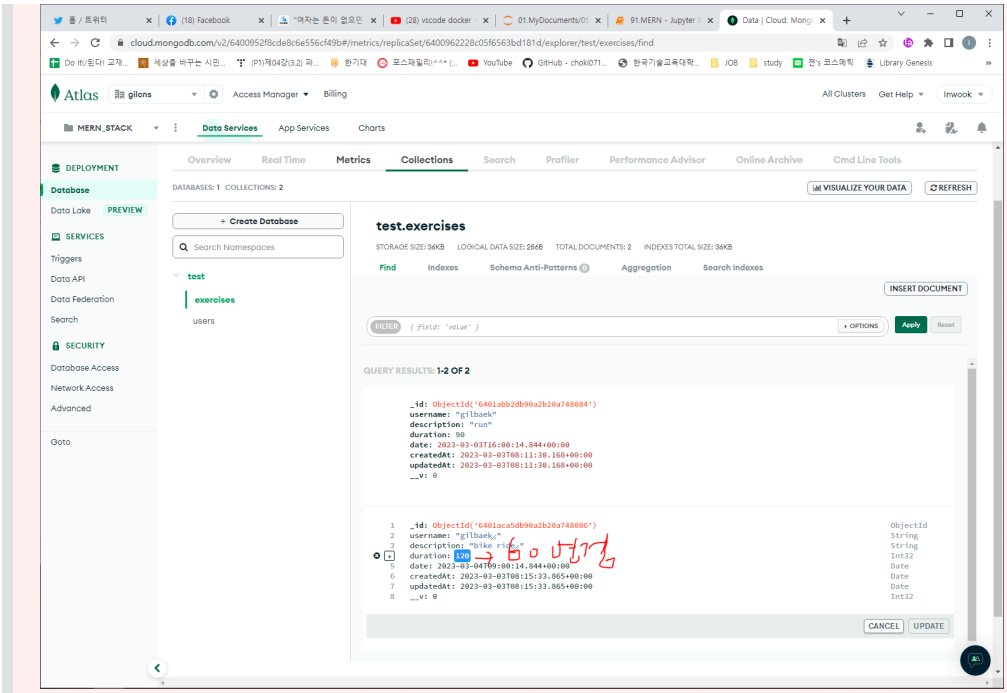
2.8.2 get

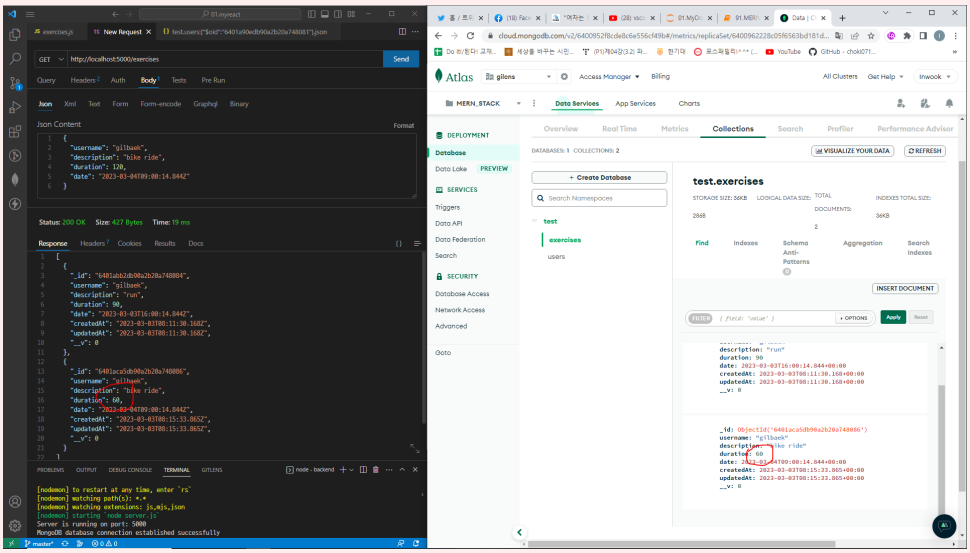
- `http://localhost:5000/exercises`



2.8.3 mongodb 수정후 get

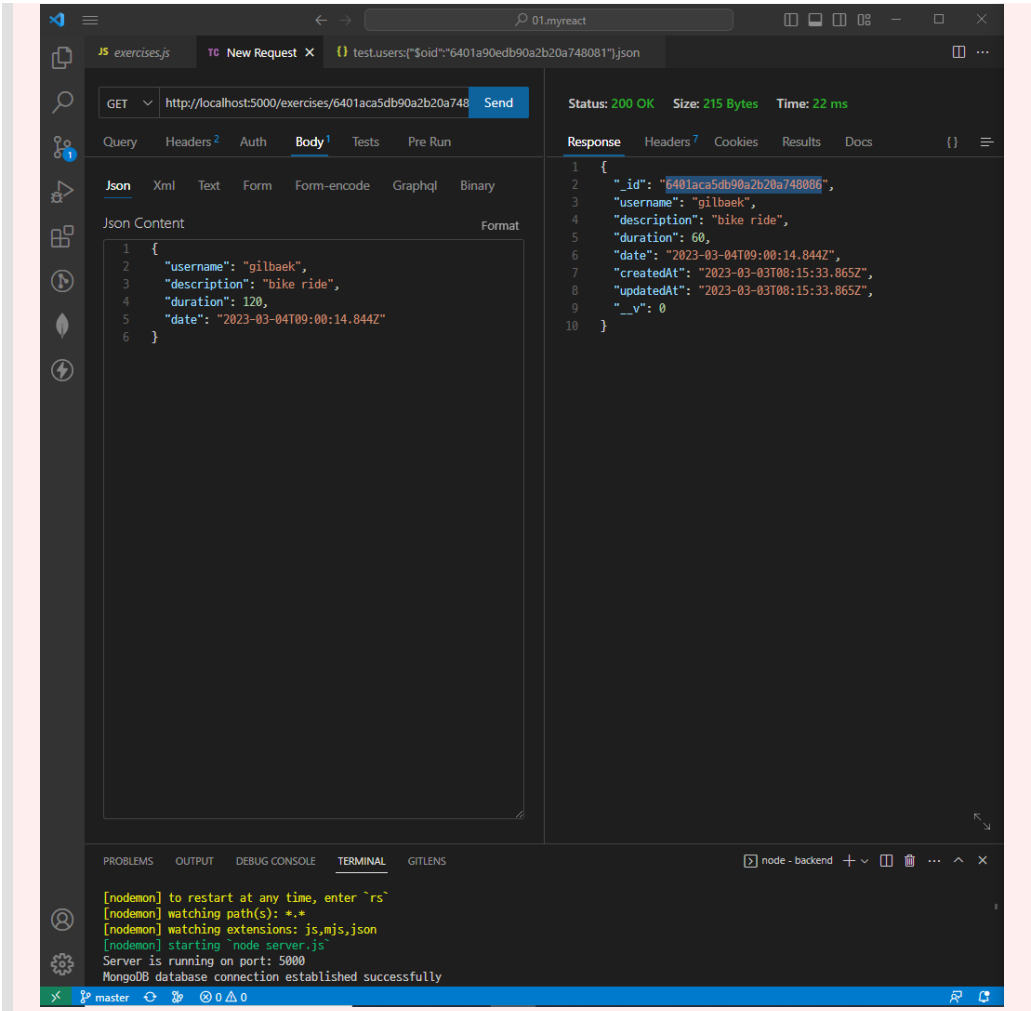
- `http://localhost:5000/exercises`





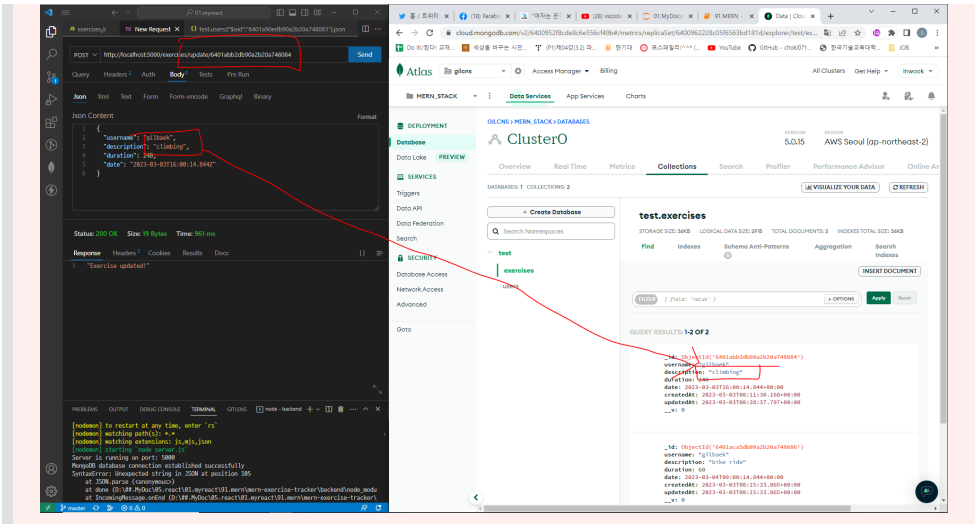
2.9.1 update - id로 조회(get)

- `http://localhost:5000/exercises/6401aca5db90a2b20a748086`
- mongodb 에서 id 복사 후 get



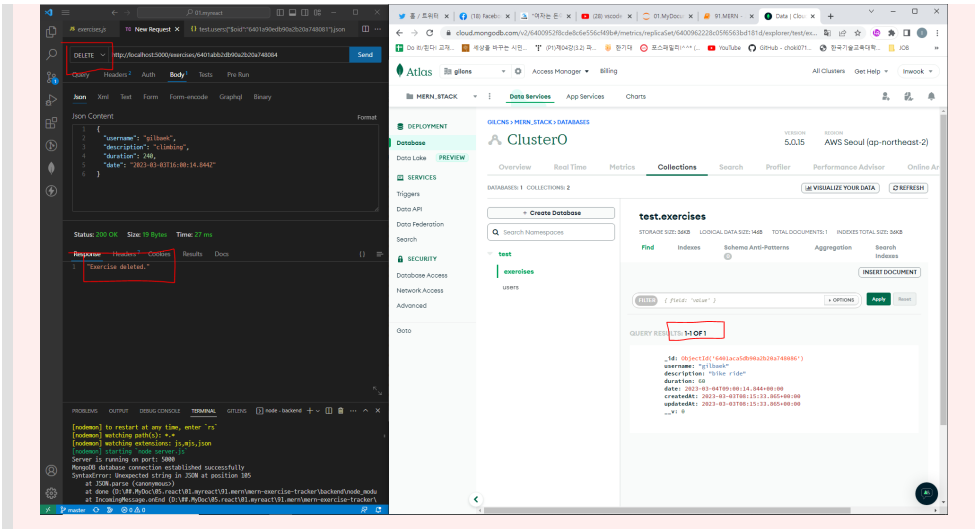
2.9.2 update

- `http://localhost:5000/exercises/update/6401aca5db90a2b20a748086`



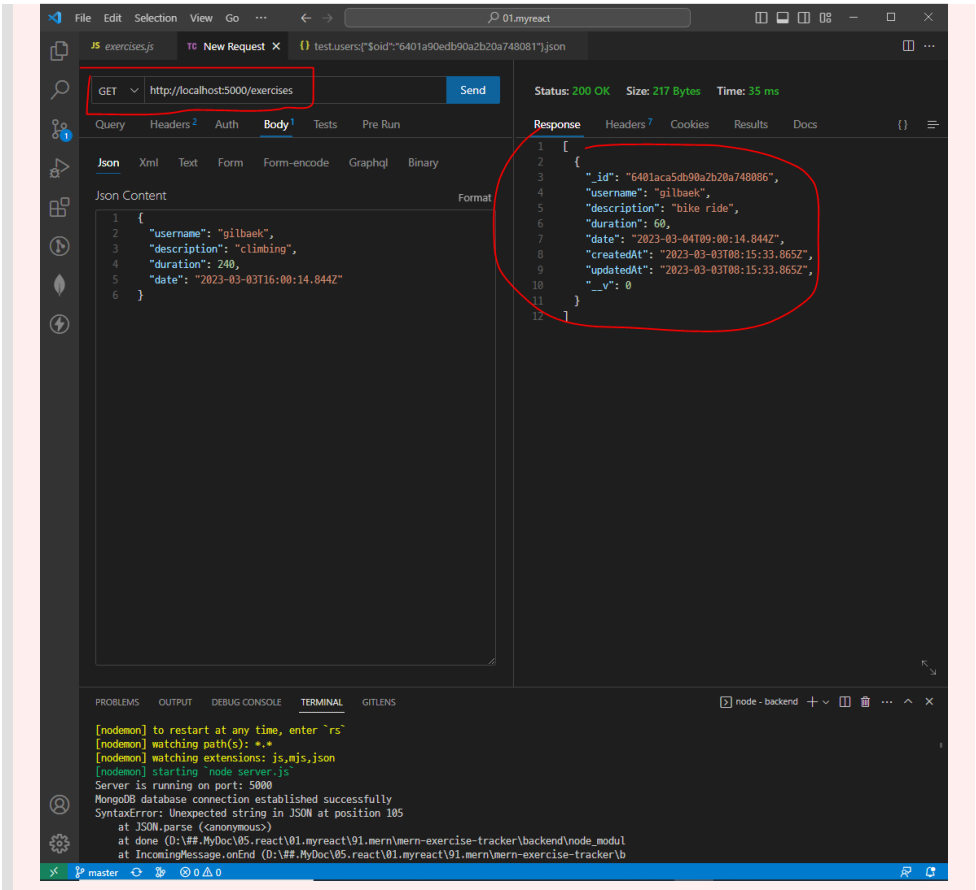
2.10.1 delete

- `http://localhost:5000/exercises/6401aca5db90a2b20a748086`



2.10.2 get

- `http://localhost:5000/exercises`



3. FrontEnd

1. index.html

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
```

```
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <meta name="theme-color" content="#000000" />

    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png"
/>

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <title>Exercise Tracker</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.
</noscript>
    <div id="root"></div>
  </body>
</html>
```

1. index.js

```
index.js
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

reportWebVitals();
```

1. App.js

```
App.js
import './App.css';

function App() {
  return (
    <div className="container">
      Hello World!!!
    </div>
  );
}

export default App;
```

1. 패키지 설치

react-bootsrap 설치

```
npm install react-bootstrap bootstrap
```

react-router 설치

- react-router-dom은 강의자료와는 버전차이 때문에 에러 발생시 해결방법
  1. https://velog.io/@seondal/React-Router-오류-6.x.x-버전에서-바뀐-사용법
  2. https://likedev.tistory.com/33 1.https://taehyeki.tistory.com/103

```
npm install react-router-dom
```

react-datepicker설치

```
npm install react-datepicker
```

axios설치

```
npm install axios
```

3.1 패키지 설치후 프로그램 수정 및 작성 - 1차

3.1.1 App.js

App.js

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-
router-dom';
import 'bootstrap/dist/css/bootstrap.min.css';

import Navbar from "./components/navbar.component"
import ExercisesList from "./components/exercises-list.component";
import EditExercise from "./components/edit-exercise.component";
import CreateExercise from "./components/create-
exercise.component";
import CreateUser from "./components/create-user.component";

function App() {
  return (
    <div className="container">
      <Router>
        <Navbar />
        <br/>
        <Routes>
          <Route path="/" exact element={<ExercisesList />} />
          <Route path="/edit/:id" element={<EditExercise />} />
          <Route path="/create" element={<CreateExercise />} />
          <Route path="/user" element={<CreateUser />} />
        </Routes>
      </Router>
    </div>
  );

}

export default App;
```

3.2 Comonents 작성

- src/components/navbar.component.js
- src/components/exercises-list.component.js
- src/components/edit-exercise.component.js
- src/components/create-exercise.component.js
- src/components/create-user.component.js

1. navbar.component.js

```
navbar.component.js
import React, { Component } from 'react';
import { Link } from 'react-router-dom';

export default class Navbar extends Component {

  render() {
    return (

      <nav className="navbar navbar-dark bg-dark navbar-expand-
lg">
        <Link to="/" className="navbar-brand">ExcerTracker</Link>
        <div className="collpase navbar-collapse">
          <ul className="navbar-nav mr-auto">
            <li className="navbar-item">
            </li>
            <li className="navbar-item">
            <Link to="/create" className="nav-link">Create
Exercise Log</Link>
            </li>
            <li className="navbar-item">
            <Link to="/user" className="nav-link">Create
User</Link>
            </li>
          </ul>
        </div>
      </nav>
    );
  }
}
```

1. exercises-list.component.js

```
import React, { Component} from "react";

export default class ExercisesList extends Component {
  render() {
    return(
      <div>
        <h4>Exercises List Component</h4>
      </div>
    )
  }
}
```

1. edit-exercise.component.js

```
import React, { Component} from "react";

export default class EditExercise extends Component {
  render() {
    return(
      <div>
        <h4>Edit Exercise Component</h4>
      </div>
    )
  }
}
```

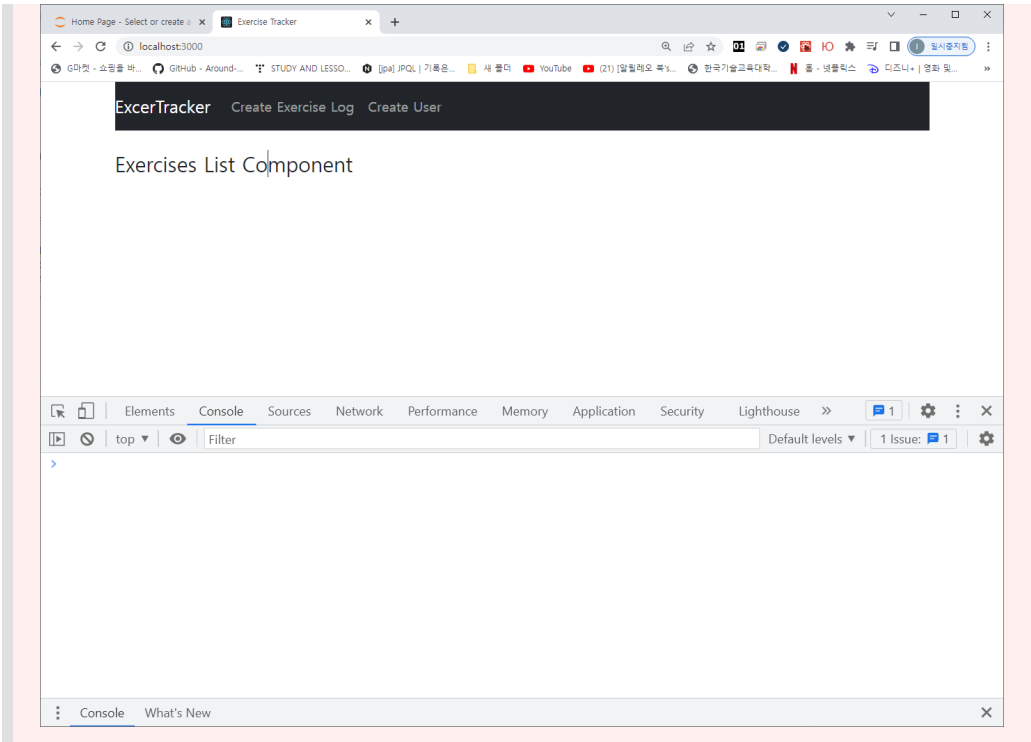
1. create-exercise.component.js

```
import React, { Component} from "react";

export default class CreateExercise extends Component {
  render() {
    return(
      <div>
        <h4>Create Exercise Component</h4>
      </div>
    )
  }
}
```

1. create-user.component.js

```
import React, { Component} from "react";

export default class CreateUsers extends Component {
  render() {
    return(
      <div>
        <h4>Create Users Component</h4>
      </div>
    )
  }
}
```

3.3 1차 수정 결과



3.4 2차 수정결과 : create-exercise.component.js

```
create-exercise.component.js
import React, { Component } from 'react';
// import axios from 'axios';
import DatePicker from 'react-datepicker';
import "react-datepicker/dist/react-datepicker.css";

export default class CreateExercise extends Component {
  constructor(props) {
    super(props);

    this.onChangeUsername = this.onChangeUsername.bind(this);
    this.onChangeDescription =
this.onChangeDescription.bind(this);
    this.onChangeDuration = this.onChangeDuration.bind(this);
    this.onChangeDate = this.onChangeDate.bind(this);
    this.onSubmit = this.onSubmit.bind(this);

    this.state = {
      username: '',
      description: '',
      duration: 0,
      date: new Date(),
      users: []
    }
  }

  componentDidMount() {
    this.setState({
      users: ['test user'],
      username: 'test user'
    })
  }
```

```
onChangeUsername(e) {
  this.setState({
    username: e.target.value
  })
}

onChangeDescription(e) {
  this.setState({
    description: e.target.value
  })
}

onChangeDuration(e) {
  this.setState({
    duration: e.target.value
  })
}

onChangeDate(date) {
  this.setState({
    date: date
  })
}

onSubmit(e) {

  e.preventDefault();

  const exercise = {
    username: this.state.username,
    description: this.state.description,
    duration: this.state.duration,
    date: this.state.date
  }


  console.log(exercise);

  // axios.post('http://localhost:5000/exercises/add', exercise)
  //   .then(res => console.log(res.data));

  window.location = '/';
}

render() {
  return (
  <div>
    <h3>Create New Exercise Log</h3>
    <form onSubmit={this.onSubmit}>
      <div className="form-group">
        <label>Username: </label>
        <select ref="userInput"
            required
            className="form-control"
            value={this.state.username}
            onChange={this.onChangeUsername}>
```

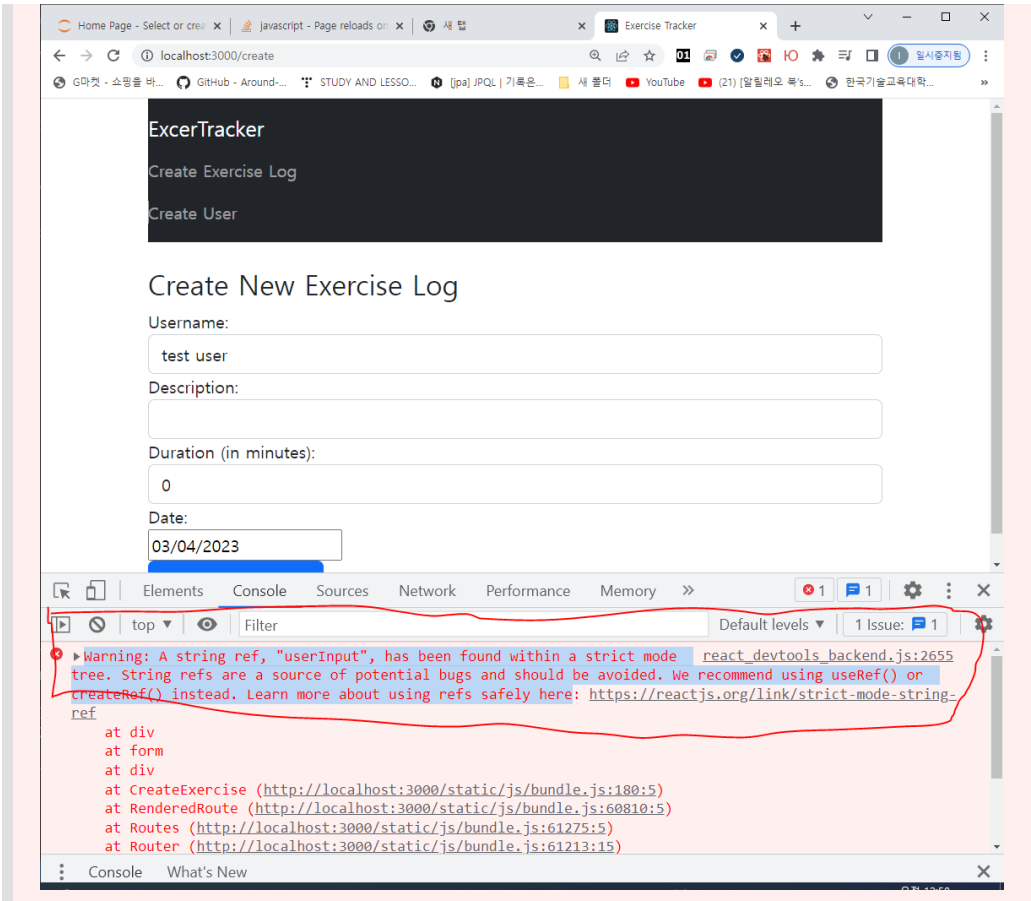
```
                {
                  this.state.users.map(function(user) {
                    return <option
                      key={user}
                      value={user}>{user}
                      </option>;
                  })
                }
              </select>
            </div>
            <div className="form-group">
              <label>Description: </label>
              <input  type="text"
                  required
                  className="form-control"
                  value={this.state.description}
                  onChange={this.onChangeDescription}
                  />
            </div>
            <div className="form-group">
              <label>Duration (in minutes): </label>
              <input
                  type="text"
                  className="form-control"
                  value={this.state.duration}
                  onChange={this.onChangeDuration}
                  />
            </div>
            <div className="form-group">
              <label>Date: </label>
              <div>
                <DatePicker
                  selected={this.state.date}
                  onChange={this.onChangeDate}
                />
              </div>
            </div>

            <div className="form-group">
              <input type="submit" value="Create Exercise Log"
  className="btn btn-primary" />
            </div>
          </form>
        </div>
        )
      }
  }
```

아래처럼 경고메시지가 출력될때 `index.js`에서 `<React.StrictMode>`를 주석처리 할 것

```
<React.StrictMode>
```

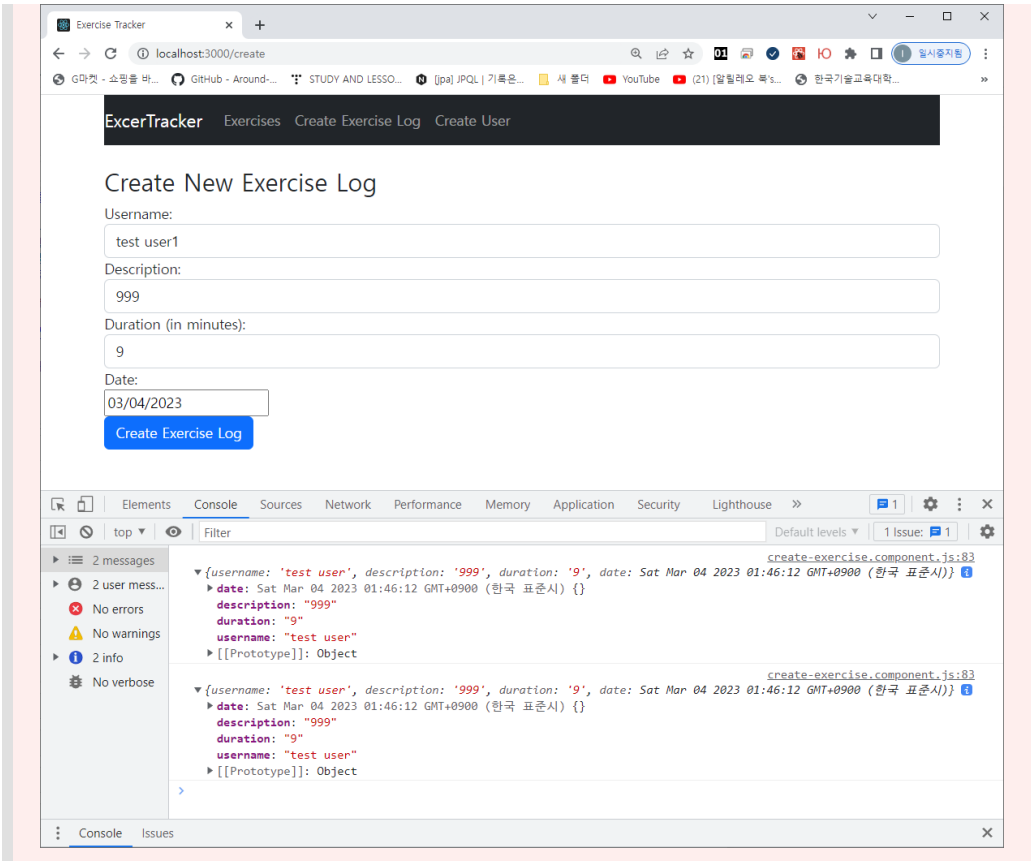
- StrictMode는 애플리케이션 내의 잠재적인 문제파악 도구이다.
- Fragment와 같이 UI를 렌더링하지 않으며, 자손들에 대한 부가적인 검사와 경고를 활성화한다.

- 주의 : Strict 모드는 개발 모드에서만 활성화되기 때문에, 프로덕션 빌드에는 영향을 끼치지 않는다.
- 참고사이트 : https://ko.reactjs.org/docs/strict-mode.html



3.4 .1 onSubmit 테스트

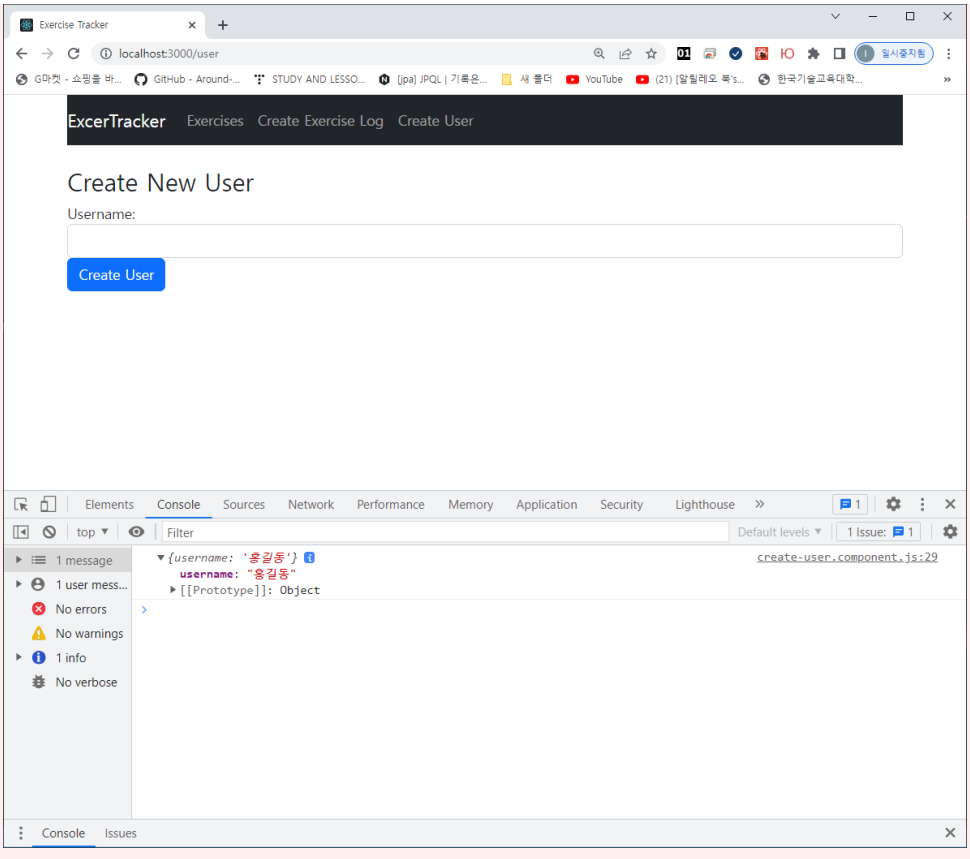
- window.location = '/'; 주석처리후 실행
- 실행결과 개발자도구창 확인



3.5. create-user.component.js

```
create-user.component.js
import React, { Component} from "react";

export default class CreateUsers extends Component {

  constructor(props) {
    super(props);

    this.onChangeUsername = this.onChangeUsername.bind(this);
    this.onSubmit = this.onSubmit.bind(this);

    this.state = {
      username: ''
    }
  }

  onChangeUsername(e) {
    this.setState({
      username: e.target.value
    })
  }

  onSubmit(e) {
    e.preventDefault();

    const user = {
      username: this.state.username
    }
```

```
    console.log(user);

    this.setState({
      username: ''
    })
  }

  render() {
    return(
      <div>
        <h3>Create New User</h3>
        <form onSubmit={this.onSubmit}>
          <div className="form-group">
            <label>Username: </label>
            <input  type="text"
                required
                className="form-control"
                value={this.state.username}
                onChange={this.onChangeUsername}
                />
          </div>
          <div className="form-group">
            <input type="submit" value="Create User"
className="btn btn-primary" />
          </div>
        </form>
      </div>
    )
  }
}
```



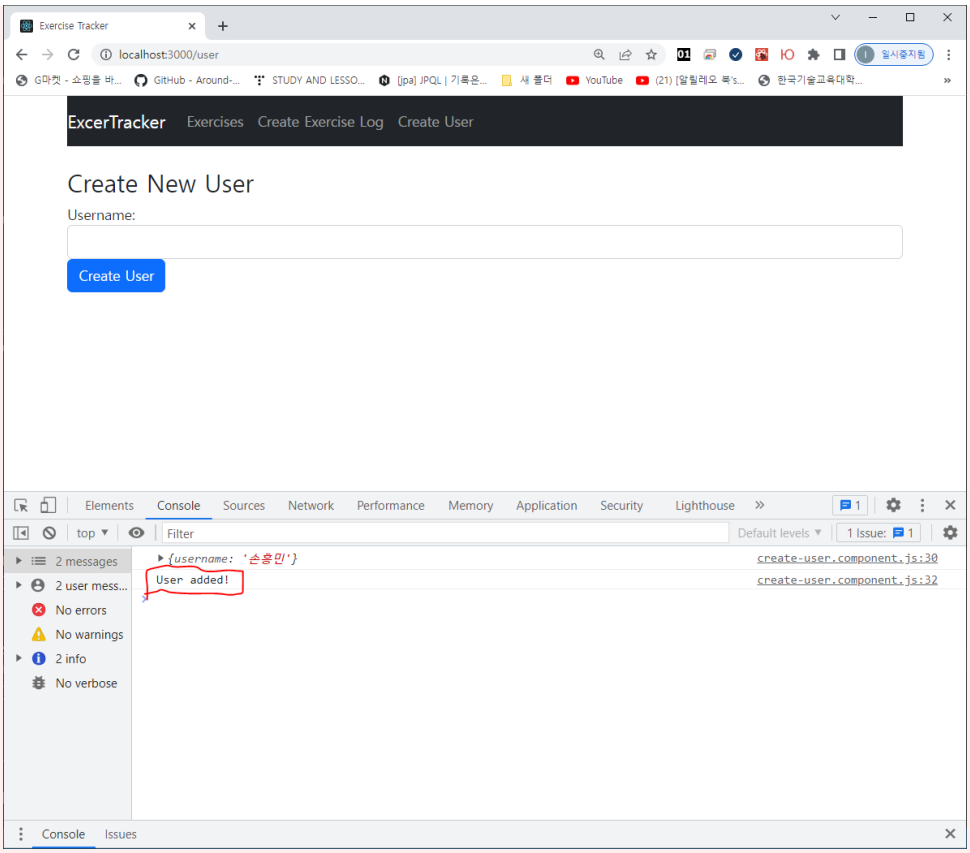
4. FrontEnd와 BackEnd 연동

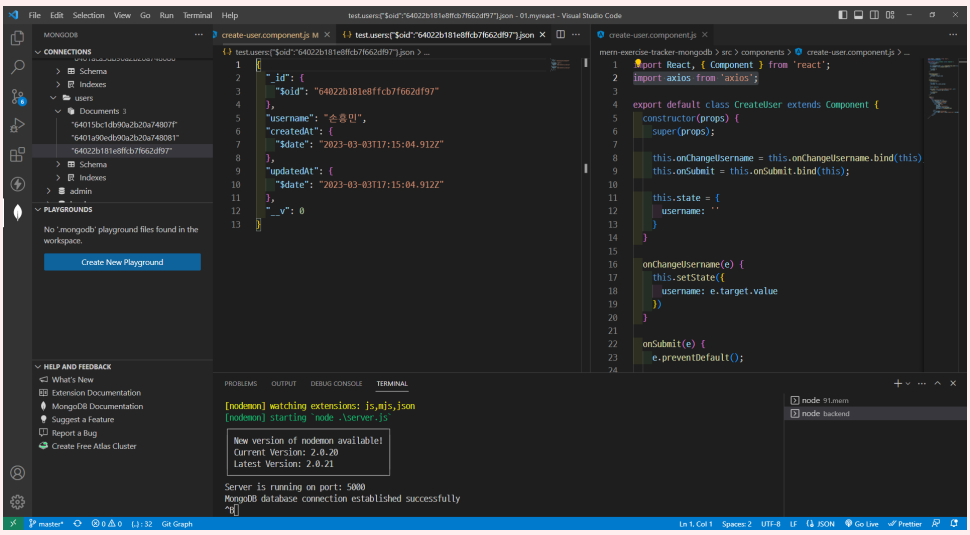
4.1 axios 반영하기

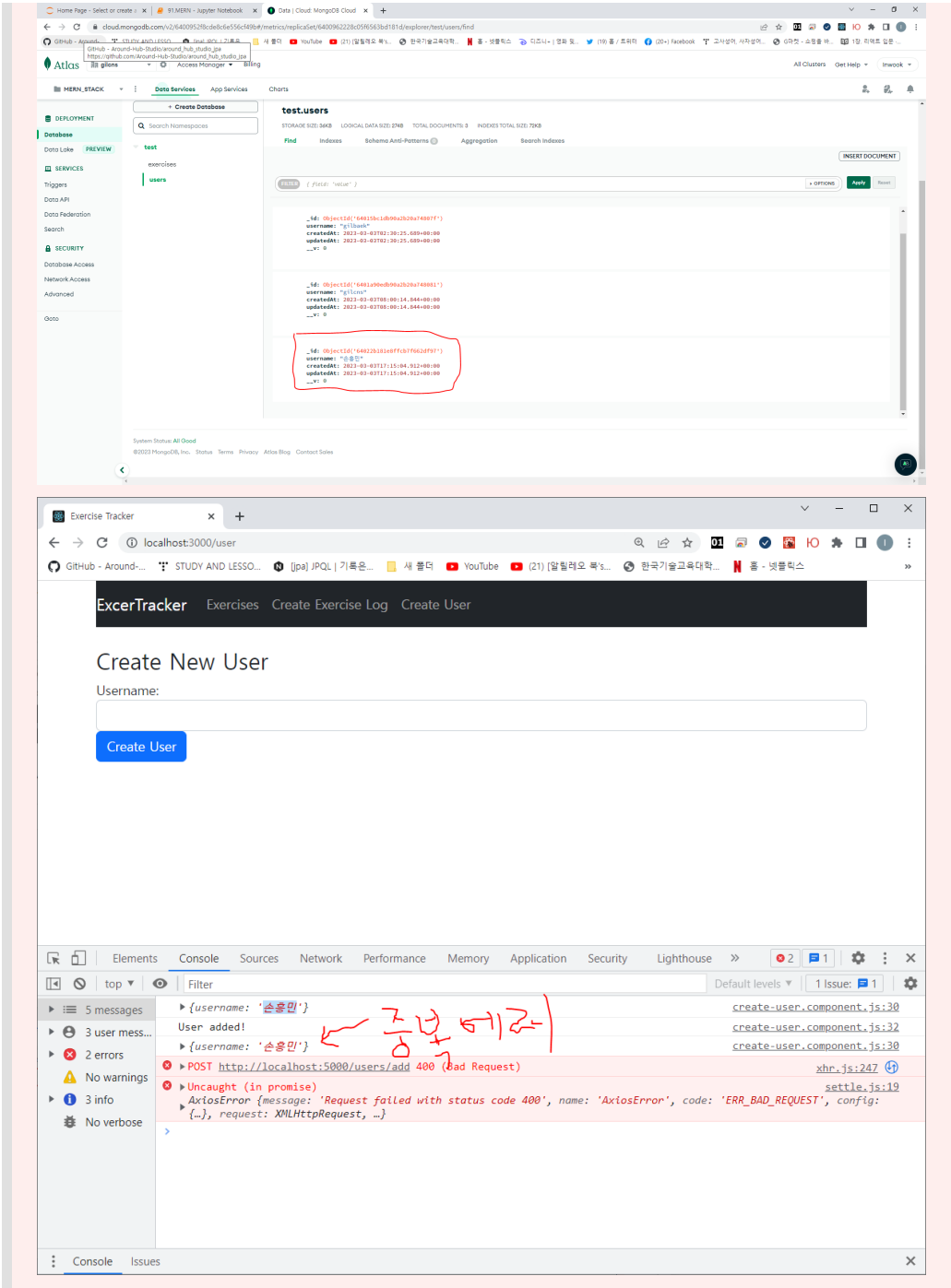
4.1.1 create-user.component.js

```
create-user.component.js
import React, { Component} from "react";
import axios from 'axios';

export default class CreateUsers extends Component {

  constructor(props) {
    super(props);

    this.onChangeUsername = this.onChangeUsername.bind(this);
    this.onSubmit = this.onSubmit.bind(this);

    this.state = {
      username: ''
    }
  }

  onChangeUsername(e) {
    this.setState({
      username: e.target.value
    })
  }

  onSubmit(e) {
    e.preventDefault();

    const user = {
      username: this.state.username
    }

    console.log(user);

    axios.post('http://localhost:5000/users/add', user)
      .then(res => console.log(res.data));

    this.setState({
      username: ''
    })
  }

  render() {
    return(
      <div>
        <h3>Create New User</h3>
        <form onSubmit={this.onSubmit}>
          <div className="form-group">
            <label>Username: </label>
```

```
                <input  type="text"
                    required
                    className="form-control"
                    value={this.state.username}
                    onChange={this.onChangeUsername}
                    />
                </div>
                <div className="form-group">
                  <input type="submit" value="Create User"
className="btn btn-primary" />
                </div>
              </form>
            </div>
        )
      }
}
```







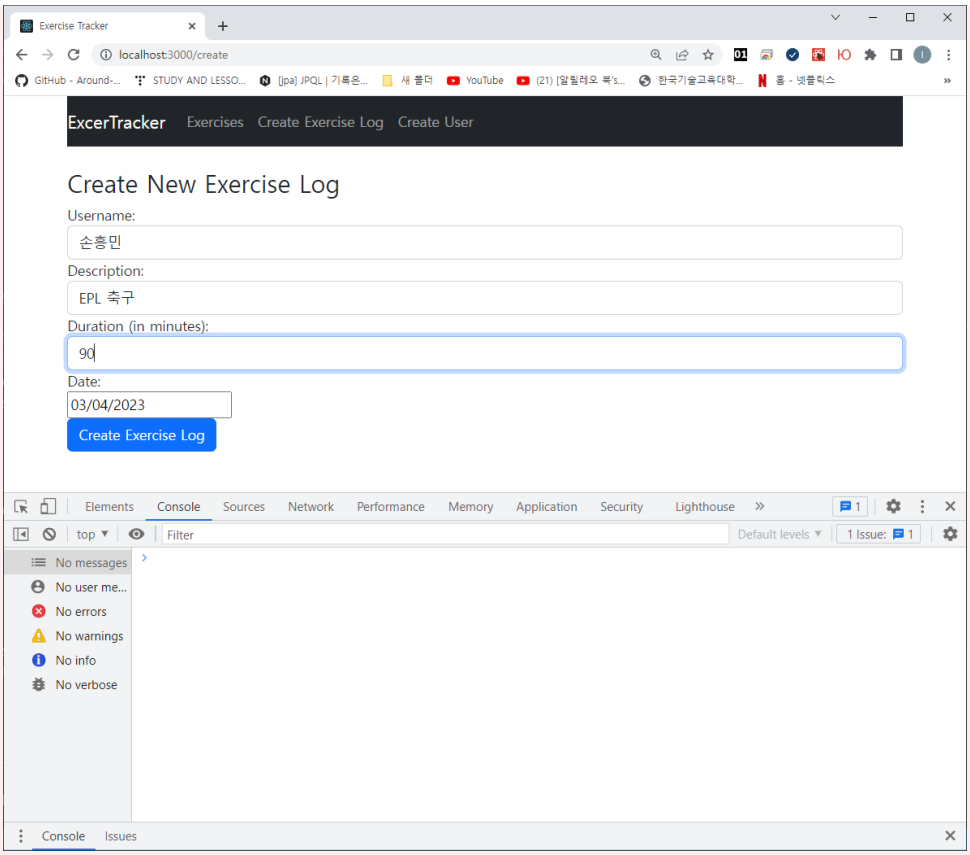
4.1.2 create-exercise.component.js

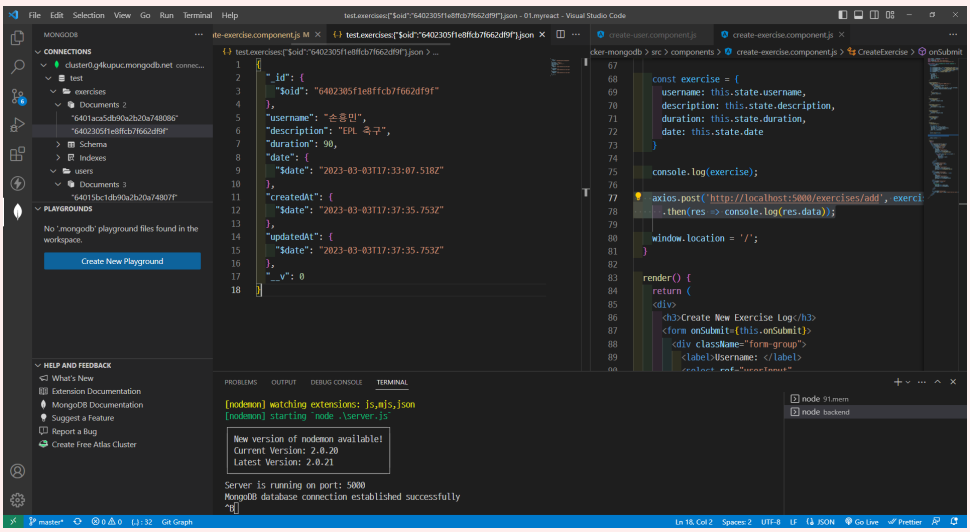
```
create-exercise.component.js
import React, { Component } from 'react';
import axios from 'axios';
import DatePicker from 'react-datepicker';
import "react-datepicker/dist/react-datepicker.css";

export default class CreateExercise extends Component {
  constructor(props) {
    super(props);

    this.onChangeUsername = this.onChangeUsername.bind(this);
    this.onChangeDescription =
this.onChangeDescription.bind(this);
    this.onChangeDuration = this.onChangeDuration.bind(this);
    this.onChangeDate = this.onChangeDate.bind(this);
```

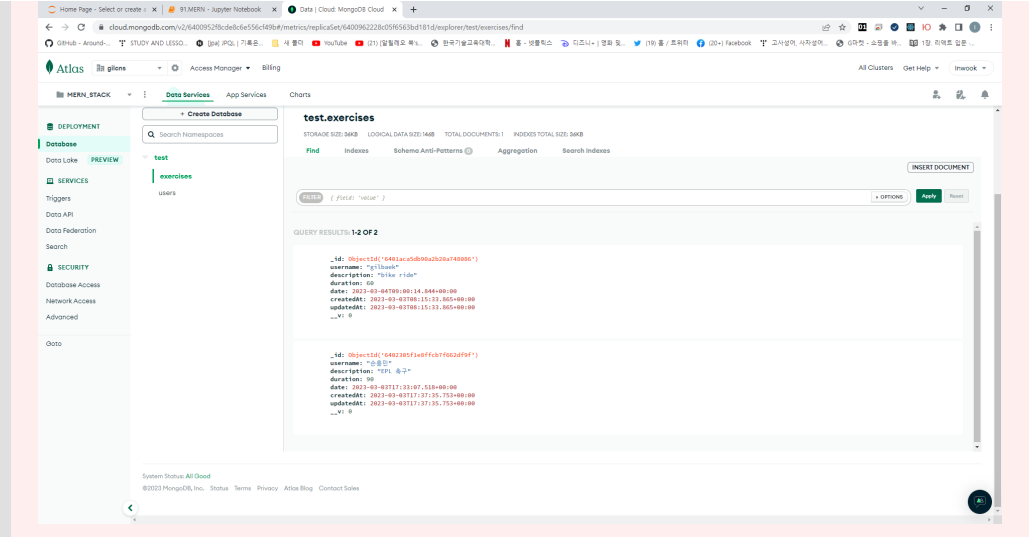
```
    this.onSubmit = this.onSubmit.bind(this);

    this.state = {
      username: '',
      description: '',
      duration: 0,
      date: new Date(),
      users: []
    }
  }

  componentDidMount() {
    axios.get('http://localhost:5000/users/')
      .then(response => {
        if (response.data.length > 0) {
          this.setState({
            users: response.data.map(user => user.username),
            username: response.data[0].username
          })
        }
      })
      .catch((error) => {
        console.log(error);
      })
  }

  onChangeUsername(e) {
    this.setState({
      username: e.target.value
    })
  }

  onChangeDescription(e) {
    this.setState({
      description: e.target.value
    })
  }

  onChangeDuration(e) {
    this.setState({
      duration: e.target.value
    })
  }

  onChangeDate(date) {
    this.setState({
      date: date
    })
  }

  onSubmit(e) {

    e.preventDefault();

    const exercise = {
      username: this.state.username,
      description: this.state.description,
```

```
      duration: this.state.duration,
      date: this.state.date
    }


    console.log(exercise);

    axios.post('http://localhost:5000/exercises/add', exercise)
      .then(res => console.log(res.data));

    window.location = '/';
  }

  render() {
    return (
    <div>
      <h3>Create New Exercise Log</h3>
      <form onSubmit={this.onSubmit}>
        <div className="form-group">
          <label>Username: </label>
          <select ref="userInput"
              required
              className="form-control"
              value={this.state.username}
              onChange={this.onChangeUsername}>
              {
                this.state.users.map(function(user) {
                  return <option
                    key={user}
                    value={user}>{user}
                    </option>;
                })
              }
          </select>
        </div>
        <div className="form-group">
          <label>Description: </label>
          <input  type="text"
              required
              className="form-control"
              value={this.state.description}
              onChange={this.onChangeDescription}
              />
        </div>
        <div className="form-group">
          <label>Duration (in minutes): </label>
          <input
              type="text"
              className="form-control"
              value={this.state.duration}
              onChange={this.onChangeDuration}
              />
        </div>
        <div className="form-group">
          <label>Date: </label>
          <div>
            <DatePicker
```

```
              selected={this.state.date}
              onChange={this.onChangeDate}
            />
          </div>
        </div>

        <div className="form-group">
          <input type="submit" value="Create Exercise Log"
className="btn btn-primary" />
        </div>
      </form>
    </div>
    )
  }
}
```








```
// props.match.params은 react-router-dom v6.x 버전에서는
// userParams() hook을 사용해야 한다.
// let { id } = userParams() 하지만 해결방법을 찾지 못했다.
// 어렴풋이나마 hook은 class 내부가 아니라 function에서 사용하는
// 방법만 검색했을 뿐 class안에서의 componentDidMount()안에서
// 사용방법을 찾지 못했다. ㅠㅠ
```

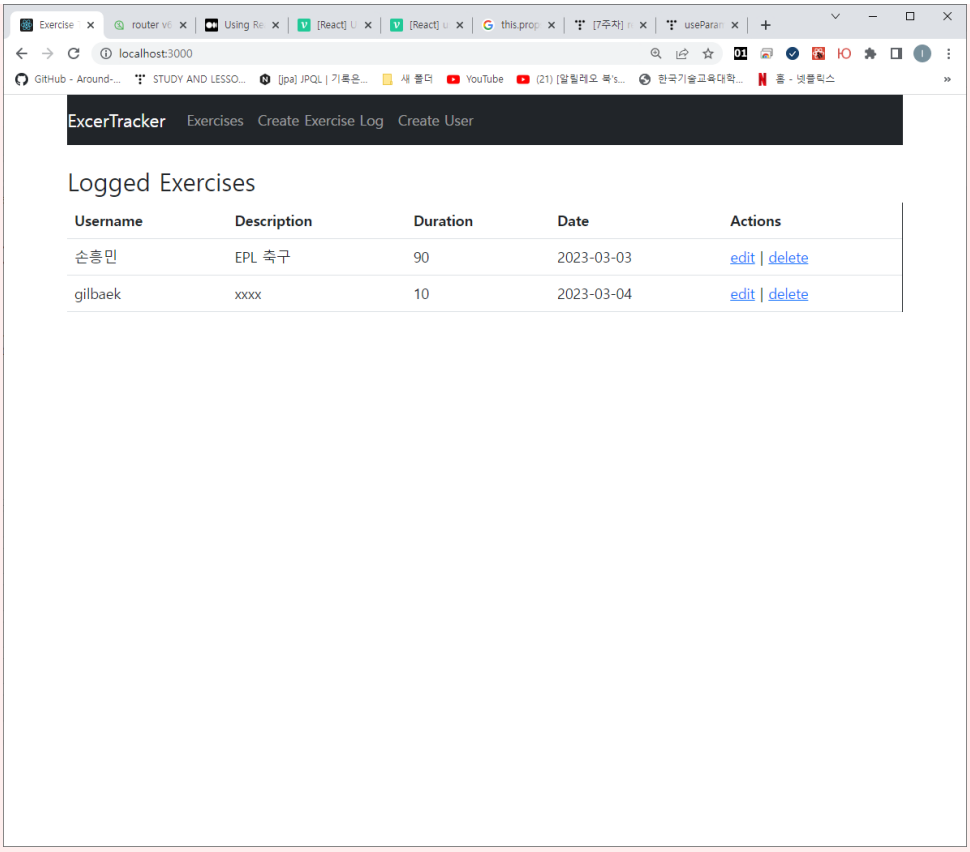
4.1.3 운동목록보기 exercises-list.component.js

```
exercises-list.component.js
import React, { Component } from 'react';
import axios from 'axios';
import DatePicker from 'react-datepicker';
import "react-datepicker/dist/react-datepicker.css";

export default class EditExercise extends Component {
  constructor(props) {
    super(props);

    this.onChangeUsername = this.onChangeUsername.bind(this);
    this.onChangeDescription =
this.onChangeDescription.bind(this);
    this.onChangeDuration = this.onChangeDuration.bind(this);
    this.onChangeDate = this.onChangeDate.bind(this);
    this.onSubmit = this.onSubmit.bind(this);

    this.state = {
      username: '',
      description: '',
      duration: 0,
      date: new Date(),
      users: [],
      id: ''
    }
  }

  componentDidMount() {

    // props.match.params은 react-router-dom v6.x 버전에서는
```

```
    // userParams() hook을 사용해야 한다.
    // let { id } = userParams() 하지만 해결방법을 찾지 못했다.
    // 어렴풋이나마 hook은 class 내부가 아니라 function에서 사용하는
    // 방법만 검색했을 뿐 class안에서의 componentDidMount()안에서
    // 사용방법을 찾지 못했다. ㅠㅠ


axios.get('http://localhost:5000/exercises/'+this.props.match.params.id)
      .then(response => {
        this.setState({
          username: response.data.username,
          description: response.data.description,
          duration: response.data.duration,
          date: new Date(response.data.date)
        })
      })
      .catch(function (error) {
        console.log(error);
      })

    axios.get('http://localhost:5000/users/')
      .then(response => {
        if (response.data.length > 0) {
          this.setState({
            users: response.data.map(user => user.username),
          })
        }
      })
      .catch((error) => {
        console.log(error);
      })

  }

  onChangeUsername(e) {
    this.setState({
      username: e.target.value
    })
  }

  onChangeDescription(e) {
    this.setState({
      description: e.target.value
    })
  }

  onChangeDuration(e) {
    this.setState({
      duration: e.target.value
    })
  }

  onChangeDate(date) {
    this.setState({
      date: date
    })
  }
```

```
  onSubmit(e) {
    e.preventDefault();

    const exercise = {
      username: this.state.username,
      description: this.state.description,
      duration: this.state.duration,
      date: this.state.date
    }

    console.log(exercise);

    axios.post('http://localhost:5000/exercises/update/' +
this.props.match.params.id, exercise)
      .then(res => console.log(res.data));

    window.location = '/';
  }

  render() {
    return (
    <div>
      <h3>Edit Exercise Log</h3>
      <form onSubmit={this.onSubmit}>
        <div className="form-group">
          <label>Username: </label>
          <select ref="userInput"
              required
              className="form-control"
              value={this.state.username}
              onChange={this.onChangeUsername}>
              {
                this.state.users.map(function(user) {
                  return <option
                    key={user}
                    value={user}>{user}
                    </option>;
                })
              }
          </select>
        </div>
        <div className="form-group">
          <label>Description: </label>
          <input  type="text"
              required
              className="form-control"
              value={this.state.description}
              onChange={this.onChangeDescription}
              />
        </div>
        <div className="form-group">
          <label>Duration (in minutes): </label>
          <input
              type="text"
              className="form-control"
              value={this.state.duration}
```

```
              onChange={this.onChangeDuration}
              />
          </div>
          <div className="form-group">
            <label>Date: </label>
            <div>
              <DatePicker
                selected={this.state.date}
                onChange={this.onChangeDate}
              />
            </div>
          </div>

          <div className="form-group">
            <input type="submit" value="Edit Exercise Log"
className="btn btn-primary" />
          </div>
        </form>
      </div>
      )
    }
}
```



4.1.4 운동내역 수정하기 edit-exercise.component.js

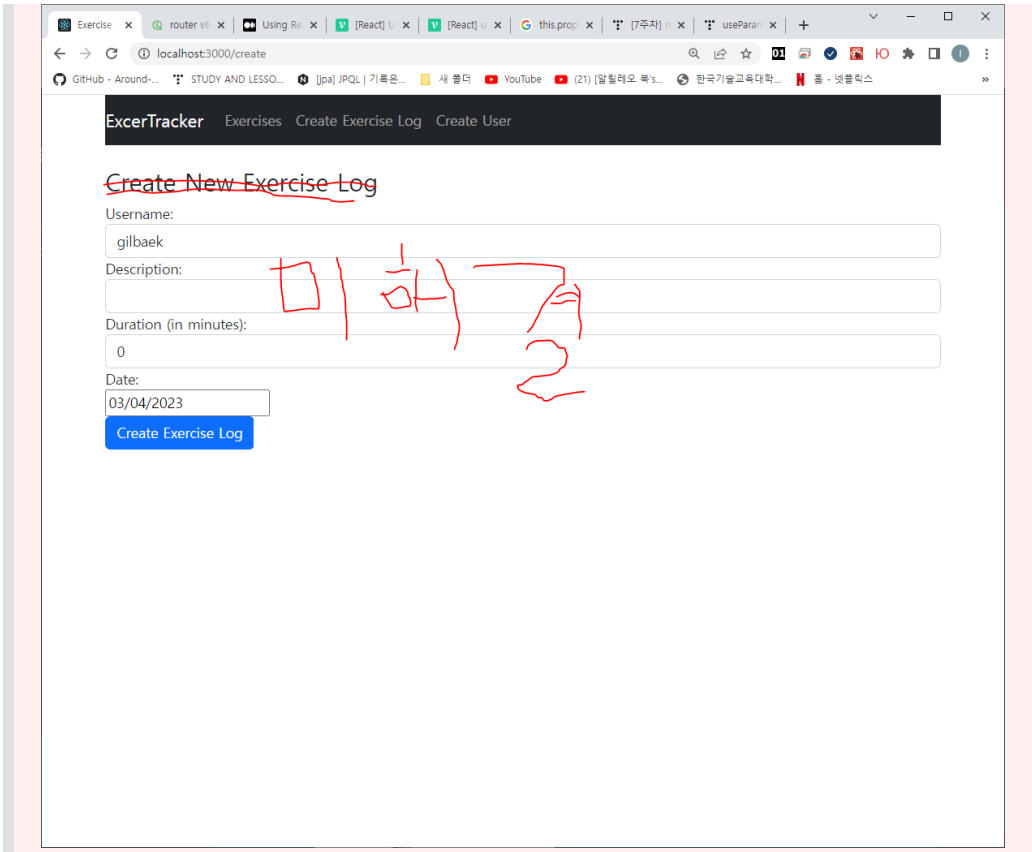
- `react-router-dom v6에서는 this.props.match.params.id 을 사용할 수 없고 let { id ] = userParams()를 사용해야 한다.`
- `이 issue를 아직 해결하지 못함!!! ㅠㅠ`

edit-exercise.component.js

```
import React, { Component } from 'react';
import axios from 'axios';
import DatePicker from 'react-datepicker';
import "react-datepicker/dist/react-datepicker.css";

export default class EditExercise extends Component {
  constructor(props) {
    super(props);

    this.onChangeUsername = this.onChangeUsername.bind(this);
    this.onChangeDescription =
this.onChangeDescription.bind(this);
    this.onChangeDuration = this.onChangeDuration.bind(this);
    this.onChangeDate = this.onChangeDate.bind(this);
    this.onSubmit = this.onSubmit.bind(this);

    this.state = {
      username: '',
      description: '',
      duration: 0,
      date: new Date(),
      users: [],
    }
  }

  componentDidMount() {

    // props.match.params은 react-router-dom v6.x 버전에서는
    // userParams() hook을 사용해야 한다.
    // let { id } = userParams() 하지만 해결방법을 찾지 못했다.
    // 어렴풋이나마 hook은 class 내부가 아니라 function에서 사용하는
    // 방법만 검색했을 뿐 class안에서의 componentDidMount()안에서
    // 사용방법을 찾지 못했다. ㅠㅠ


axios.get('http://localhost:5000/exercises/'+this.props.match.params.id)
      .then(response => {
        this.setState({
          username: response.data.username,
          description: response.data.description,
          duration: response.data.duration,
          date: new Date(response.data.date)
        })
      })
      .catch(function (error) {
        console.log(error);
      })

    axios.get('http://localhost:5000/users/')
      .then(response => {
        if (response.data.length > 0) {
          this.setState({
            users: response.data.map(user => user.username),
          })
        }
      })
```

```
      .catch((error) => {
        console.log(error);
      })

  }

  onChangeUsername(e) {
    this.setState({
      username: e.target.value
    })
  }

  onChangeDescription(e) {
    this.setState({
      description: e.target.value
    })
  }

  onChangeDuration(e) {
    this.setState({
      duration: e.target.value
    })
  }

  onChangeDate(date) {
    this.setState({
      date: date
    })
  }

  onSubmit(e) {
    e.preventDefault();

    const exercise = {
      username: this.state.username,
      description: this.state.description,
      duration: this.state.duration,
      date: this.state.date
    }

    console.log(exercise);

    axios.post('http://localhost:5000/exercises/update/' +
this.props.match.params.id, exercise)
      .then(res => console.log(res.data));

    window.location = '/';
  }

  render() {
    return (
    <div>
      <h3>Edit Exercise Log</h3>
      <form onSubmit={this.onSubmit}>
        <div className="form-group">
          <label>Username: </label>
          <select ref="userInput"
```

```
                required
                className="form-control"
                value={this.state.username}
                onChange={this.onChangeUsername}>
                {
                  this.state.users.map(function(user) {
                    return <option
                      key={user}
                      value={user}>{user}
                      </option>;
                  })
                }
            </select>
          </div>
          <div className="form-group">
            <label>Description: </label>
            <input  type="text"
                required
                className="form-control"
                value={this.state.description}
                onChange={this.onChangeDescription}
                />
          </div>
          <div className="form-group">
            <label>Duration (in minutes): </label>
            <input
                type="text"
                className="form-control"
                value={this.state.duration}
                onChange={this.onChangeDuration}
                />
          </div>
          <div className="form-group">
            <label>Date: </label>
            <div>
              <DatePicker
                selected={this.state.date}
                onChange={this.onChangeDate}
              />
            </div>
          </div>

          <div className="form-group">
            <input type="submit" value="Edit Exercise Log"
className="btn btn-primary" />
          </div>
        </form>
      </div>
      )
    }
}
```



4.1.5 운동내용 삭제하기