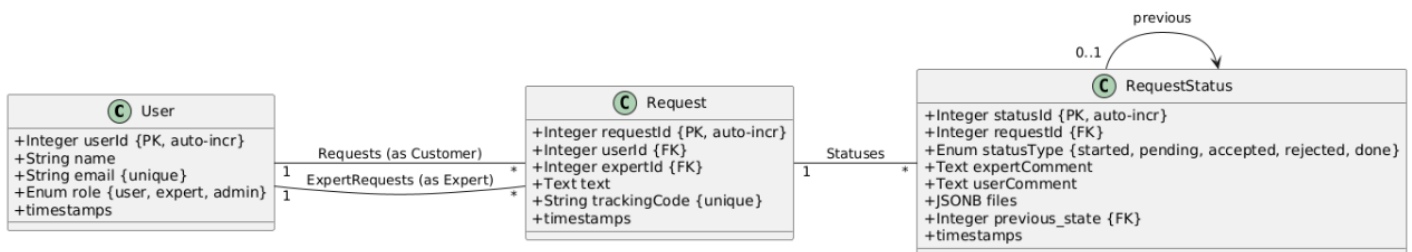


سی باتی د رات خاس ی حارط (ER-Diagram)

؟؟ ?????? ?? ?????? ?? ??? ?????? ??????????
?????? ??? ????



1. ?????? ?????????? ?????????? (ER-Diagram)

- ?????? ?????? ?????????????? (Entities): `User` | `Request` | `RequestStatus`
- ?????? ??????: ?? `User` | ?????????? ??? `Request` | ?????? ?????? ?? `Request` | ??? `RequestStatus`
- ?????? ?????????? ????? (PK) ? ?????? (FK) ?? ?? ??????

2. ?????? ?????????? Sequelize ?????? PostgreSQL

- ??? ??????????:

```
npm install sequelize pg pg-hstore
```

- ?????? ?????? ?????????? `src/configs/database.js`:

```
import { Sequelize } from 'sequelize';
const sequelize = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASS,
  {
    host: process.env.DB_HOST,
    dialect: 'postgres',
    logging: false,
```

```

    }
  );
  export default sequelize;

```

3. **????? ???** **User**
- ????? ????????? ???? ??? **userId**? **name**? **email** ? **role**

```

import { DataTypes } from 'sequelize';
import sequelize from '../configs/database.js';
const User = sequelize.define('User', {
  userId: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  },
  name: {
    type: DataTypes.STRING(100),
    allowNull: false,
  },
  email: {
    type: DataTypes.STRING(150),
    allowNull: false,
    unique: true,
  },
  role: {
    type: DataTypes.ENUM('user', 'expert', 'admin'),
    defaultValue: 'user',
  },
}, {
  tableName: 'users',
  timestamps: true,
});

export default User;

```

4. **????? ???** **Request**
- ????????? **requestId**? **userId**? **expertId**? **text** ? **trackingCode**

```

import { DataTypes } from 'sequelize';

```

```

import sequelize from '../configs/database.js';
import User from './user.js';
import { v4 as uuidv4 } from 'uuid';
const Request = sequelize.define('Request', {
  requestId: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  },
  userId: {
    type: DataTypes.INTEGER,
    references: { model: User, key: 'userId' },
    allowNull: true,
  },
  expertId: {
    type: DataTypes.INTEGER,
    references: { model: User, key: 'userId' },
    allowNull: true,
  },
  text: {
    type: DataTypes.TEXT,
    allowNull: false,
  },
  trackingCode: {
    type: DataTypes.STRING(20),
    allowNull: false,
    unique: true,
    defaultValue: () => uuidv4().slice(0, 20),
  },
}, {
  tableName: 'requests',
  timestamps: true,
});

export default Request;

```

5. ????? ??? RequestStatus

- ENUM ?????????? ????? ?? requestId ? ?????? previous_state

```

import { DataTypes } from 'sequelize';
import sequelize from '../configs/database.js';
import Request from './request.js';

const RequestStatus = sequelize.define('RequestStatus', {
  statusId: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true,
  },
  requestId: {
    type: DataTypes.INTEGER,
    references: { model: Request, key: 'requestId' },
    allowNull: false,
  },
  statusType: {
    type: DataTypes.ENUM('started', 'pending', 'accepted', 'rejected', 'done'),
    defaultValue: 'started',
  },
  expertComment: {
    type: DataTypes.TEXT,
    defaultValue: '',
  },
  userComment: {
    type: DataTypes.TEXT,
    defaultValue: '',
  },
  files: {
    type: DataTypes.JSONB,
    allowNull: true,
  },
  previous_state: {
    type: DataTypes.INTEGER,
    references: { model: 'request_statuses', key: 'statusId' },
    allowNull: true,
  },
}, {
  tableName: 'request_statuses',
  timestamps: true,
  hooks: {

```

```

        beforeCreate: async (status) => {
            const last = await RequestStatus.findOne({
                where: { requestId: status.requestId },
                order: [['statusId', 'DESC']],
            });
            if (last) status.previous_state = last.statusId;
        }
    }
});

export default RequestStatus;

```

6. ?????? ?????????? (Associations)

- ?? ????? |src/models/index.js| ?? ??????? ?? ????:

```

import User from './user.js';
import Request from './request.js';
import RequestStatus from './requestStatus.js';

// User ↔ Request
User.hasMany(Request, { foreignKey: 'userId', as: 'Requests' });
Request.belongsTo(User, { foreignKey: 'userId', as: 'Customer' });
User.hasMany(Request, { foreignKey: 'expertId', as: 'ExpertRequests' });
Request.belongsTo(User, { foreignKey: 'expertId', as: 'Expert' });

// Request ↔ RequestStatus
Request.hasMany(RequestStatus, { foreignKey: 'requestId', as: 'Statuses' });
RequestStatus.belongsTo(Request, { foreignKey: 'requestId', as: 'Request' });

export { User, Request, RequestStatus };

```

7. ??????????? ?????? ?????? (State Pattern)

- ????? ????? |src/models/state/| ? ?????? ?????????? ??????:

```

import { handleReqRejected } from '../../utils/status.utils.js';

class AcceptedState {
    constructor(context) { this.context = context; }

    async transitionTo(statusType) {
        if (!statusType) throw new Error('Invalid statusType');
        if (statusType === 'rejected') {
            return await handleReqRejected(this.context);
        }

        throw new Error(`Cannot transition from accepted to ${statusType}`);
    }
}

```

```
    }  
    }  
    export default AcceptedState;
```

- ?? ????? Request ??????? ??????:

```
import AcceptedState from './state/acceptedState.js';  
import PendingState from './state/pendingState.js';  
    // ...  
    class RequestService {  
        constructor(request) {  
            this.request = request;  
            this.state = this._getStateInstance(request.currentStatus);  
        }  
        _getStateInstance( type) {  
            switch( type) {  
            case 'accepted': return new AcceptedState(this.request);  
            // ...  
            }  
        }  
        async changeStatus( toType) {  
            return this.state.transitionTo( toType);  
        }  
    }
```

Revision #5

Created 14 May 2025 06:53:02 by a.tavana

Updated 14 May 2025 07:02:51 by a.tavana