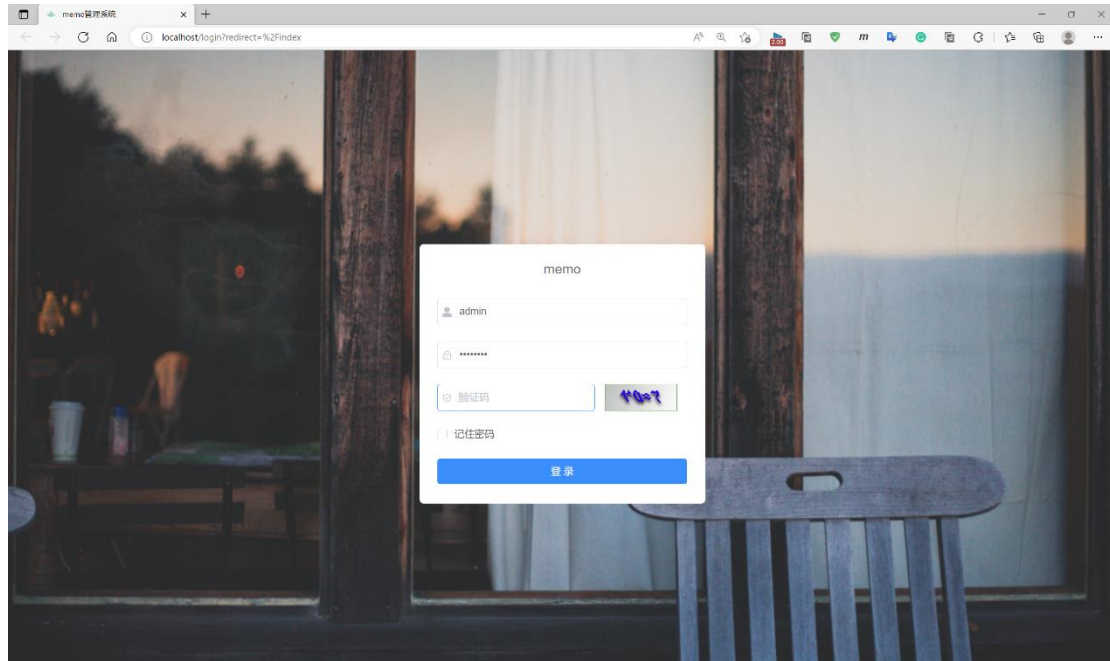


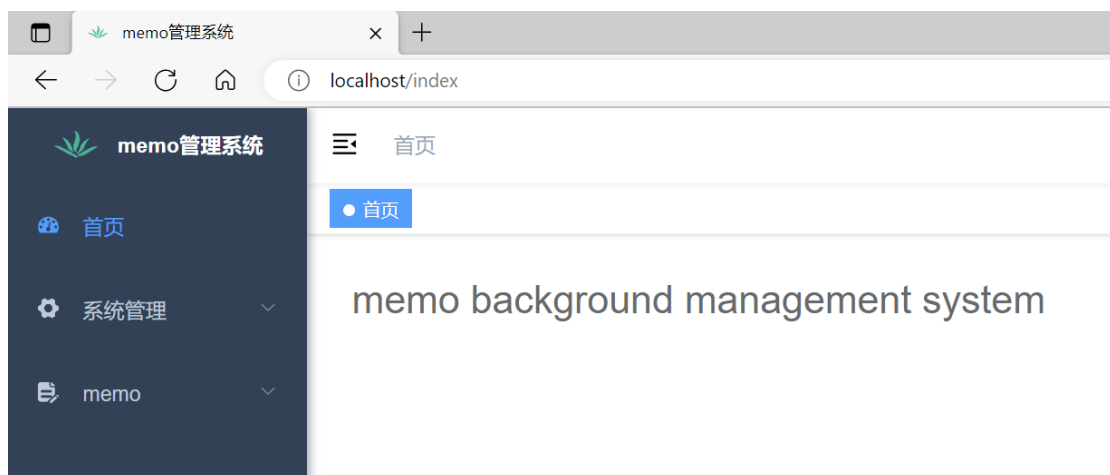
# Todo list

This document is for the use of the todolist\_memo system.

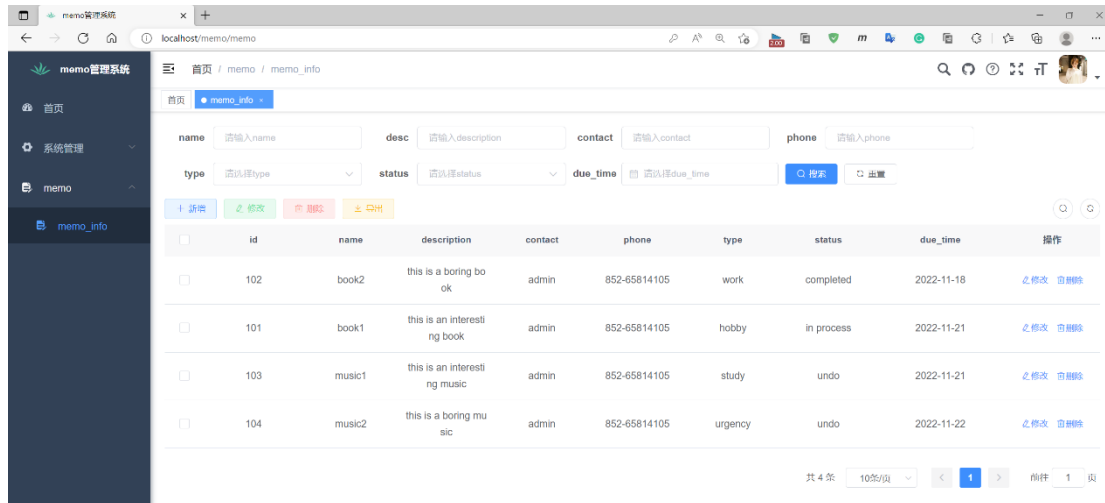
## How to use



Enter the verification code to access the login page.

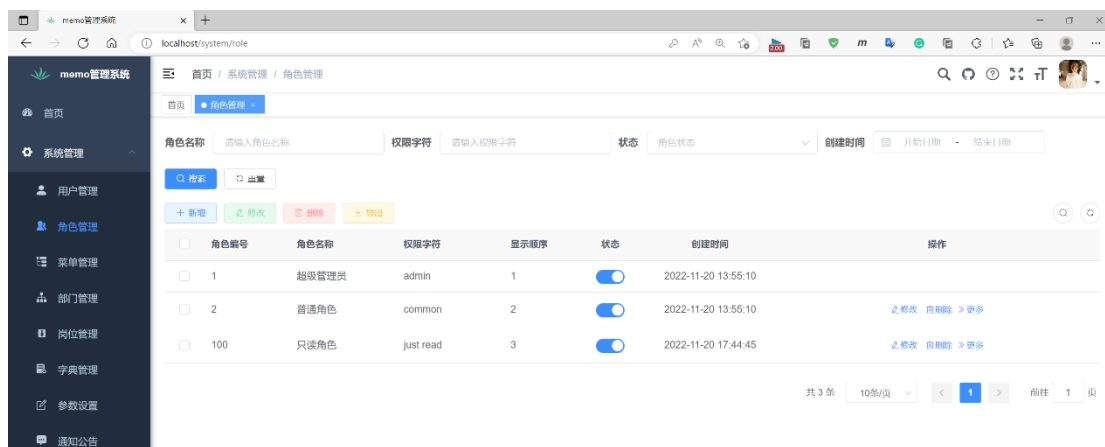


Login Page



### Memo\_list Page

All to-do list functions are implemented in this page, and the display of due\_time is sorted by date. (order by due\_time in TodoMemoMapper.xml)



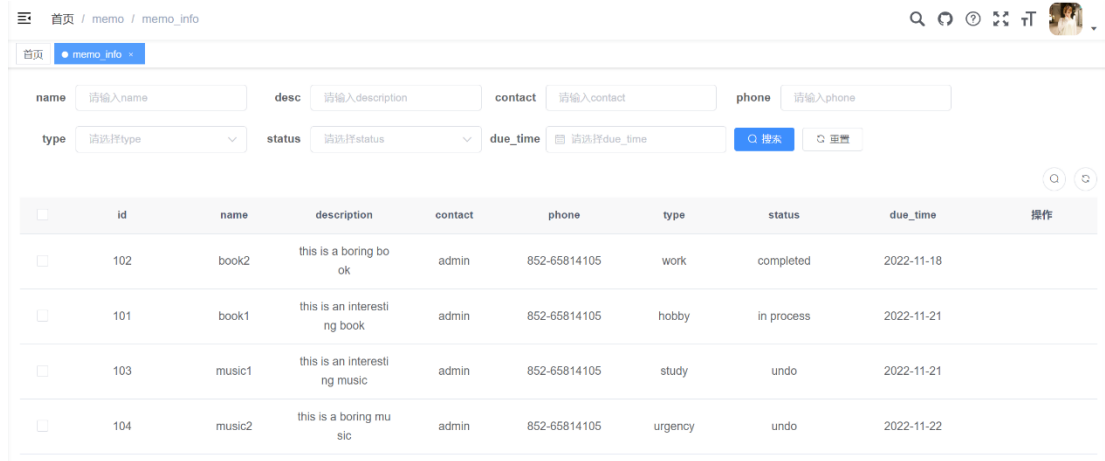
### Role Management Page

You can create different types of roles that have different permissions.



### User Management Page

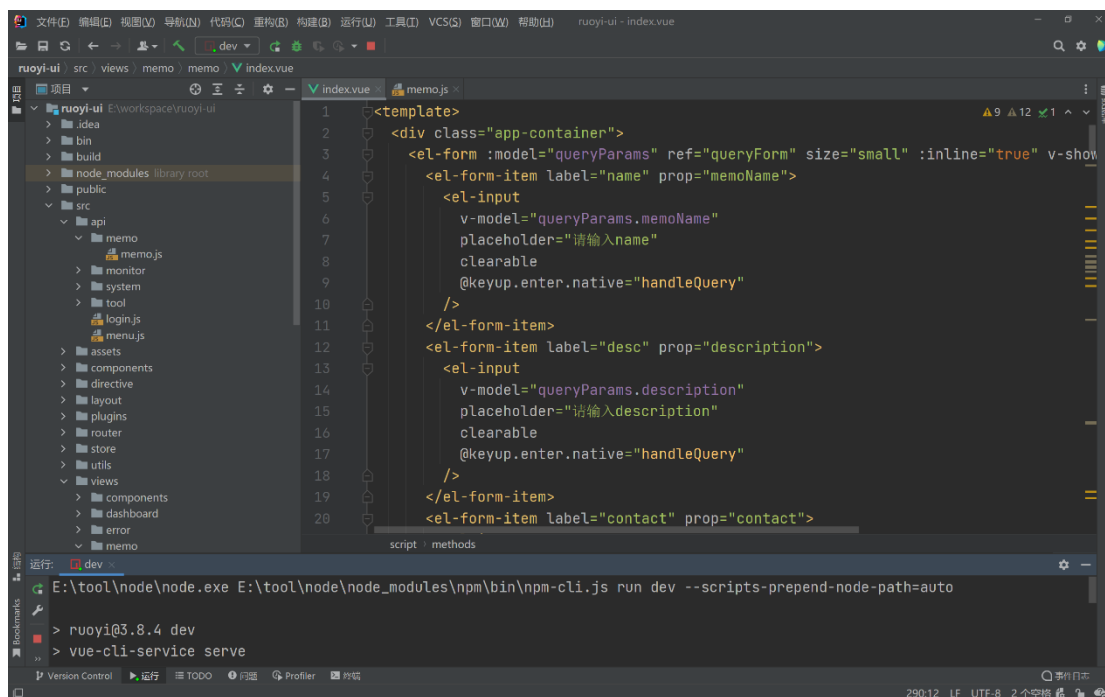
You can add different users that belong to the specified role, The read-only role can only view the memo and cannot modify the to-do list.



The Read-only Role Page

# How build

## Front end part

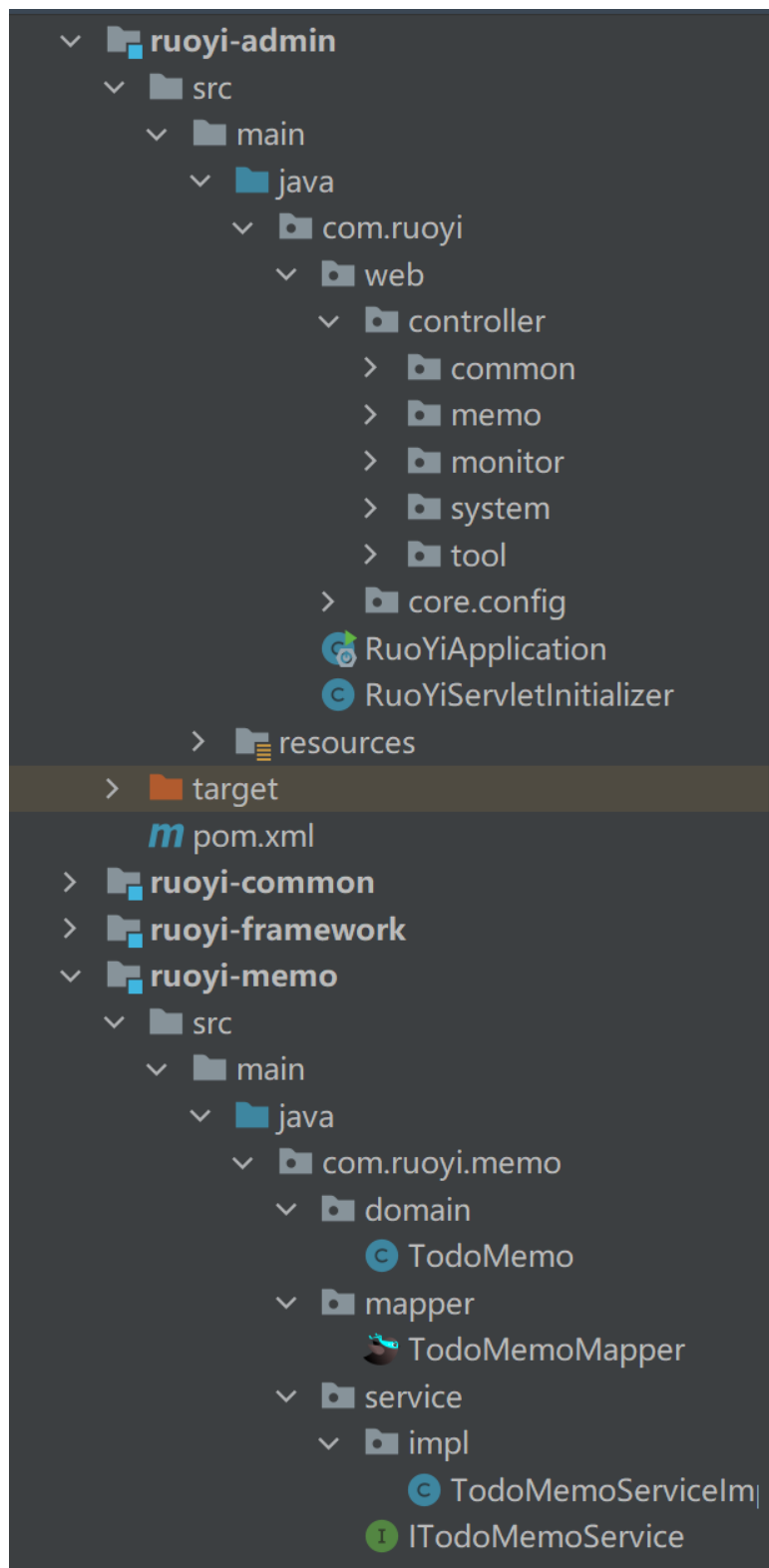


File directory and page preparation.

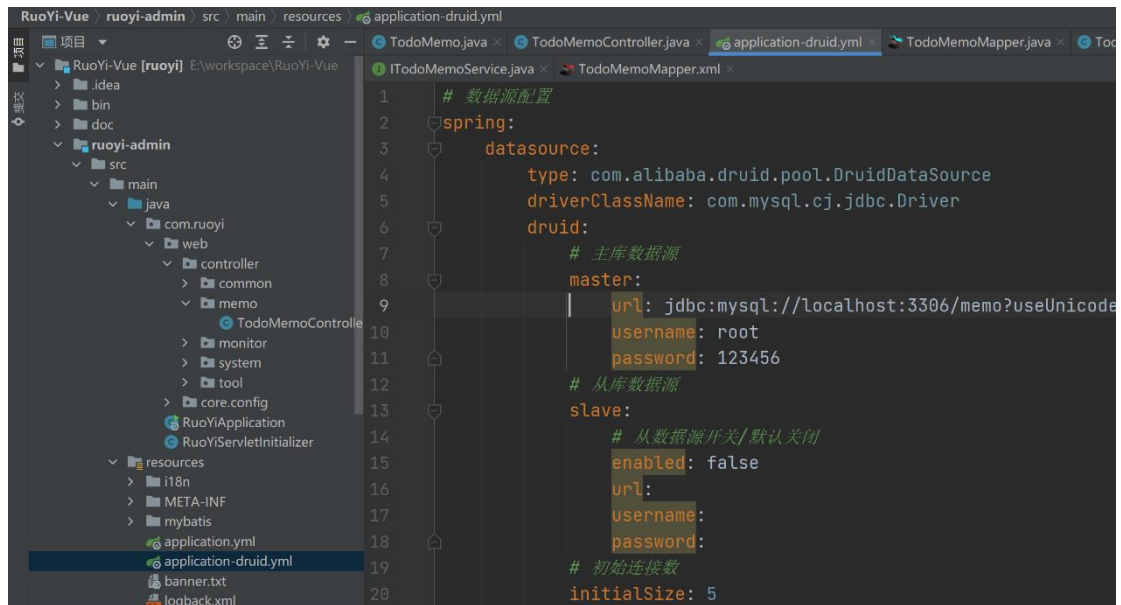
```
index.vue x memo.js x
1 import request from '@utils/request'
2
3 // 查询memo_info列表
4 export function listMemo(query) {
5   return request({
6     url: '/memo/memo/list',
7     method: 'get',
8     params: query
9   })
10 }
11
12 // 查询memo_info详细
13 export function getMemo(memoId) {
14   return request({
15     url: '/memo/memo/' + memoId,
16     method: 'get'
17   })
18 }
19
20 // 新增memo_info
21 export function addMemo(data) {
22   return request({
23     url: '/memo/memo',
24     method: 'post',
```

Mapping of the request path.

## Back end part



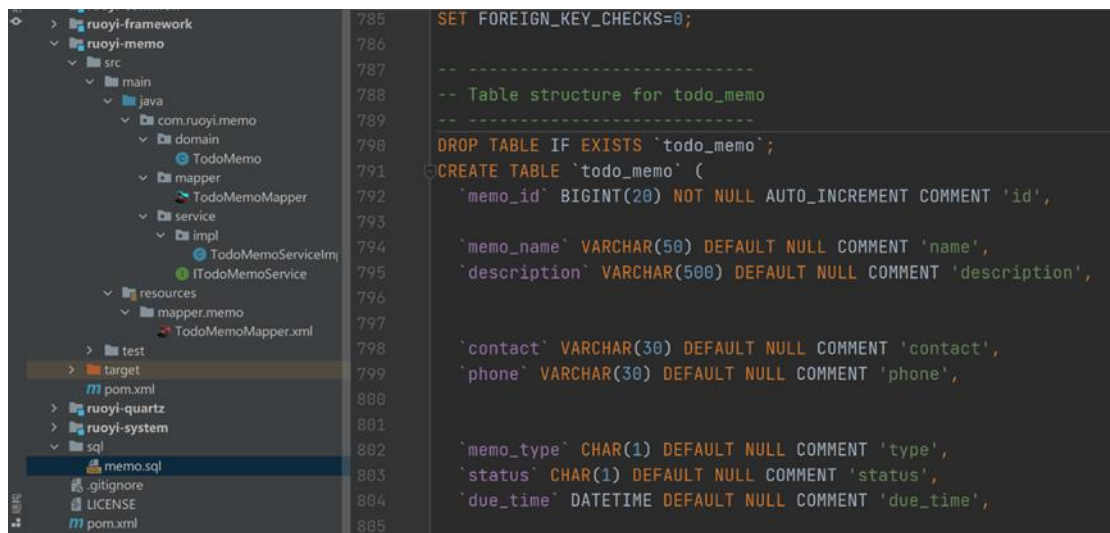
The main API implementation directory, this project relies on the ruoyi-framework for development. This framework helps us to implement the system management module, which includes user management, role management and other modules.



```
1  # 数据源配置
2  spring:
3    datasource:
4      type: com.alibaba.druid.pool.DruidDataSource
5      driverClassName: com.mysql.cj.jdbc.Driver
6      druid:
7        # 主库数据源
8        master:
9          url: jdbc:mysql://localhost:3306/memo?useUnicode=
10         username: root
11         password: 123456
12        # 从库数据源
13        slave:
14          # 从数据源开关/默认关闭
15          enabled: false
16          url:
17          username:
18          password:
19        # 初始连接数
20        initialSize: 5
```

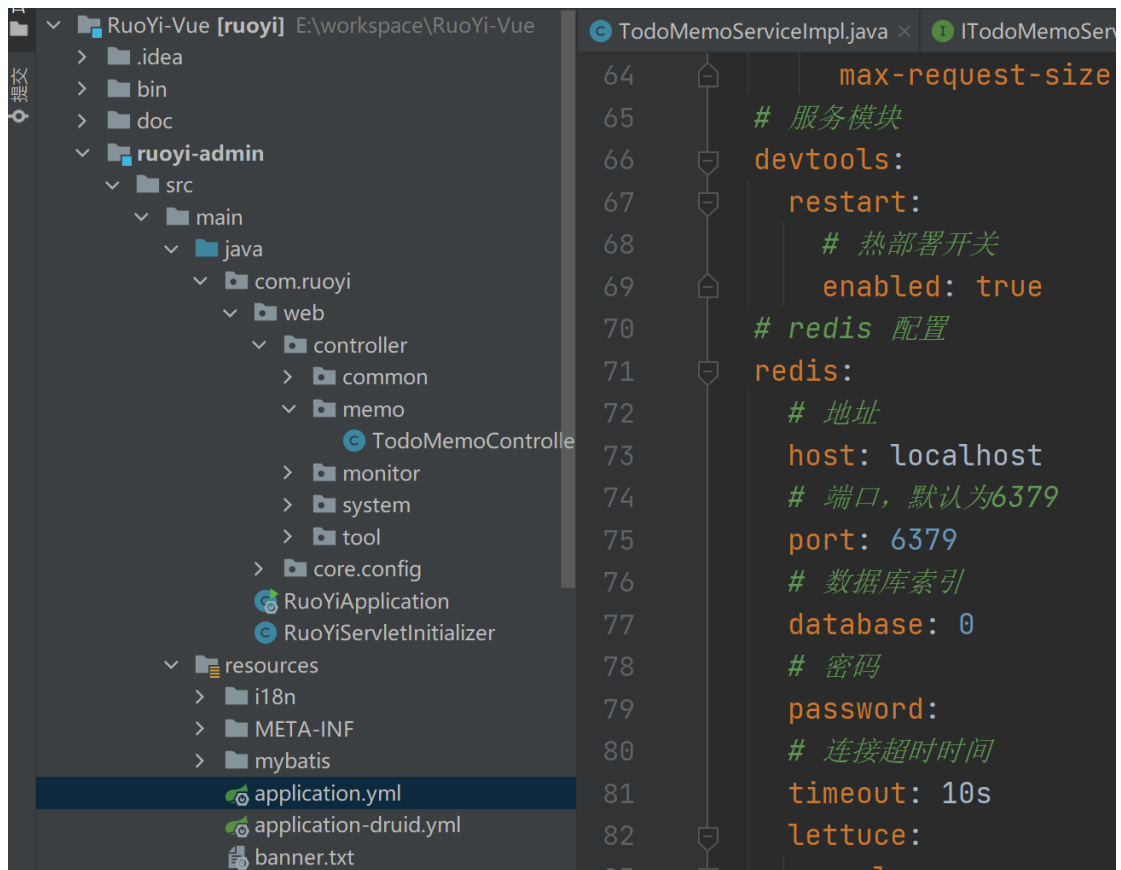
### Database Configuration

The url links to the specific database that was created, username, password set by yourself

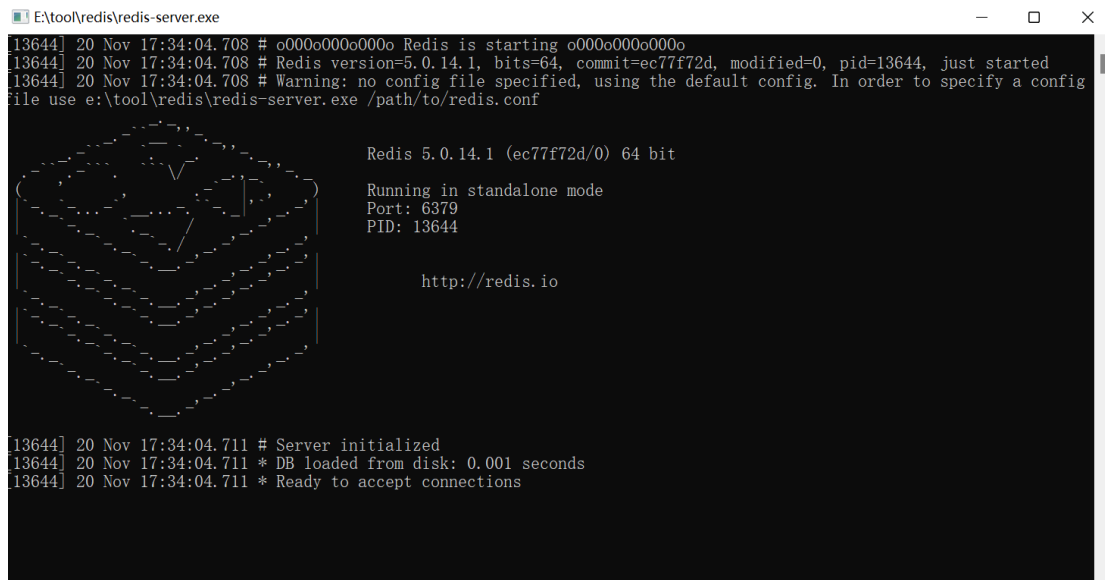


```
785 SET FOREIGN_KEY_CHECKS=0;
786
787 -----
788 -- Table structure for todo_memo
789 -----
790 DROP TABLE IF EXISTS `todo_memo`;
791 CREATE TABLE `todo_memo` (
792   `memo_id` BIGINT(20) NOT NULL AUTO_INCREMENT COMMENT 'id',
793   `memo_name` VARCHAR(50) DEFAULT NULL COMMENT 'name',
794   `description` VARCHAR(500) DEFAULT NULL COMMENT 'description',
795   `contact` VARCHAR(30) DEFAULT NULL COMMENT 'contact',
796   `phone` VARCHAR(30) DEFAULT NULL COMMENT 'phone',
797   `memo_type` CHAR(1) DEFAULT NULL COMMENT 'type',
798   `status` CHAR(1) DEFAULT NULL COMMENT 'status',
799   `due_time` DATETIME DEFAULT NULL COMMENT 'due_time',
800
801
802
803
804
805
```

SQL statement to create the database.



To implement the CAPTCHA functionality, we used the redis database as a cache.



Start the redis service.

```

<select id="selectTodoMemoList" parameterType="TodoMemo" resultMap="To
    <include refid="selectTodoMemoVo"/>
    <where>
        <if test="memoName != null and memoName != ''"> and memo_name
        <if test="description != null and description != ''"> and des
        <if test="contact != null and contact != ''"> and contact lik
        <if test="phone != null and phone != ''"> and phone like cond
        <if test="memoType != null and memoType != ''"> and memo_type
        <if test="status != null and status != ''"> and status = #{st
        <if test="dueTime != null "> and due_time = #{dueTime}</if>
    </where>
    order by due_time
</select>

<select id="selectTodoMemoByMemoId" parameterType="Long" resultMap="To
    <include refid="selectTodoMemoVo"/>
    where memo_id = #{memoId}
    order by due_time
</select>

```

The memo sorting was realized by the order of the due\_time.