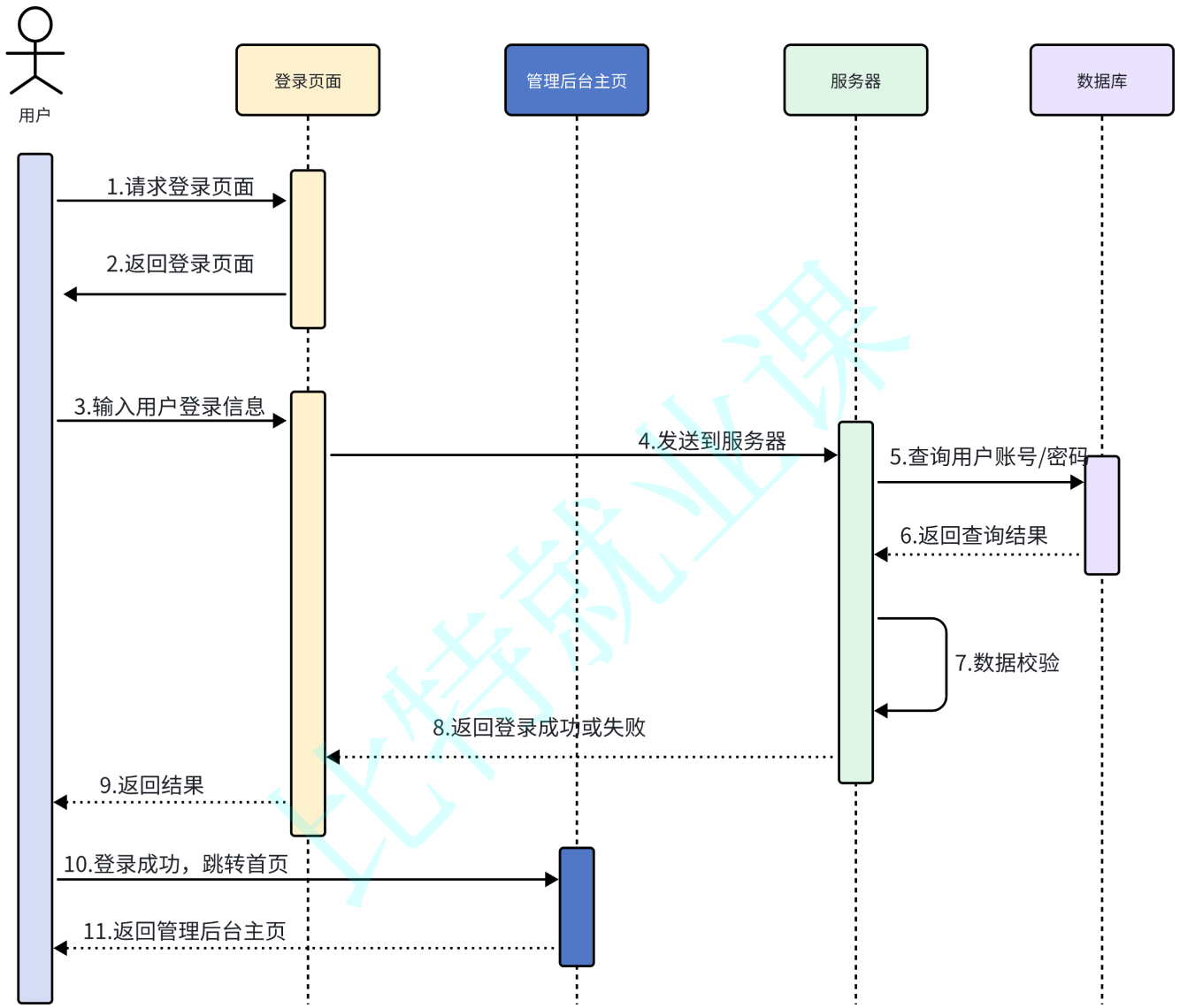
管理员登录-表结构设计

需求再分析



表结构设计：

在设计时要注意：

- 满足当前需求
- 避免冗余设计
- 考虑今后的扩展

```sql
DROP TABLE IF EXISTS `tb_sys_user`;

CREATE TABLE `tb_sys_user` (
  `user_id` bigint(20) unsigned NOT NULL COMMENT '用户id',
  `user_account` varchar(32) DEFAULT NULL COMMENT '用户账号',
  `password` varchar(100) DEFAULT NULL COMMENT '用户密码',
  `nick_name` varchar(32) DEFAULT NULL COMMENT '昵称',
  `create_by` bigint(8) NOT NULL COMMENT '创建用户',
  `create_time` datetime NOT NULL COMMENT '创建时间',
  `update_by` bigint(8) DEFAULT NULL COMMENT '更新用户',
  `update_time` datetime DEFAULT NULL COMMENT '更新时间',
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `user_account` (`user_account`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='管理端用户表'
```

实体类创建

实体类

```java
package com.bite.system.domain;

import com.baomidou.mybatisplus.annotation.TableName;

import java.time.LocalDateTime;

@TableName("tb_sys_user")
public class SysUser {

    private Long userId;

    private String userAccount;

    private String password;

    private Long createBy;

    private LocalDateTime createTime;

    private Long updateBy;

    private LocalDateTime updateTime;

    public Long getUserId() {
        return userId;
```

```
26      }
27
28      public void setUserId(Long userId) {
29          this.userId = userId;
30      }
31
32      public String getUserAccount() {
33          return userAccount;
34      }
35
36      public void setUserAccount(String userAccount) {
37          this.userAccount = userAccount;
38      }
39
40      public String getPassword() {
41          return password;
42      }
43
44      public void setPassword(String password) {
45          this.password = password;
46      }
47
48      public Long getCreateBy() {
49          return createBy;
50      }
51
52      public void setCreateBy(Long createBy) {
53          this.createBy = createBy;
54      }
55
56      public LocalDateTime getCreateTime() {
57          return createTime;
58      }
59
60      public void setCreateTime(LocalDateTime createTime) {
61          this.createTime = createTime;
62      }
63
64      public Long getUpdateBy() {
65          return updateBy;
66      }
67
68      public void setUpdateBy(Long updateBy) {
69          this.updateBy = updateBy;
70      }
71
72      public LocalDateTime getUpdateTime() {
```

```
73          return updateTime;
74      }
75
76      public void setUpdateTime(LocalDateTime updateTime) {
77          this.updateTime = updateTime;
78      }
79  }
```

主键

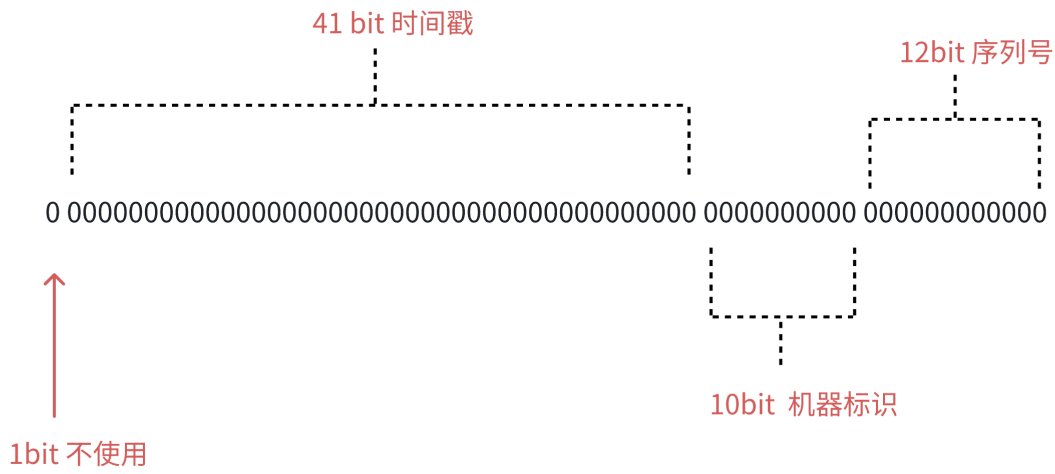
为什么不使用自增id

- **数据迁移和备份**：如果你需要将数据从一个数据库迁移到另一个数据库，或者备份和恢复数据，自增主键可能会导致问题。例如，如果你在新数据库中已经存在与旧数据库相同的自增 ID，那么插入操作可能会失败。
- **删除和插入操作**：如果表中存在大量删除和插入操作，自增主键可能会导致 ID 值的不连续。这可能会浪费存储空间，并可能导致某些应用程序或系统逻辑出现问题。
- **性能问题**：在高并发的写入操作中，自增主键可能会导致性能瓶颈。因为每次插入新记录时，数据库都需要找到下一个可用的自增 ID。这可能会增加写操作的延迟。
- **可预测性**：自增主键的值是可预测的，因为它们总是按照递增的顺序生成。这可能会带来安全风险，例如，攻击者可能会尝试预测未来的 ID 值来插入恶意数据。
- **分布式环境问题**：在分布式数据库系统中，自增主键可能会带来挑战。如何保证各个节点生成的自增id是唯一的，这将需要额外的机制来协调各个节点。

如何处理自增id问题：

使用uuid或者雪花算法生成的主键。

- **uuid**：UUID（Universally Unique Identifier，通用唯一识别码）是一种软件建构的标准，用于在分布式计算环境中为元素提供唯一的辨识信息。UUID共占128位，分为五段，它具有唯一性、全局性、不变性等特点。
- **雪花算法**：雪花算法（Snowflake）是一种分布式唯一ID生成算法，用于生成全局唯一的ID。它的设计目标是在分布式系统中生成ID，保证ID的唯一性、有序性和趋势递增。雪花算法的核心思想是将一个64位的ID分成多个部分，分别表示不同的信息。



为什么不使用uuid作为主键：

存储空间：UUID存储会占用更大的空间，这会增加数据库的存储需求，尤其是在大型数据库或高并发的系统中，这可能会成为性能瓶颈。

索引性能：由于UUID相对较大，使用UUID作为主键会导致索引变得更大，从而影响查询性能。特别是在执行范围查询或JOIN操作时，性能下降可能会更加明显。

可读性：UUID由一串字符组成，不易于人类阅读或记忆。这可能会影响到开发、运维和调试的便利性。

使用mybatis-plus支持雪花算法

```java
package com.bite.system.domain;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;

import java.time.LocalDateTime;

@TableName("tb_sys_user")
public class SysUser {

    @TableId(value = "USER_ID", type = IdType.ASSIGN_ID)
    private Long userId;

    private String userAccount;

    private String password;

    private Long createBy;

```

```
21      private LocalDateTime createTime;
22
23      private Long updateBy;
24
25      private LocalDateTime updateTime;
26
27      public Long getUserId() {
28          return userId;
29      }
30
31      public void setUserId(Long userId) {
32          this.userId = userId;
33      }
34
35      public String getUserAccount() {
36          return userAccount;
37      }
38
39      public void setUserAccount(String userAccount) {
40          this.userAccount = userAccount;
41      }
42
43      public String getPassword() {
44          return password;
45      }
46
47      public void setPassword(String password) {
48          this.password = password;
49      }
50
51      public Long getCreateBy() {
52          return createBy;
53      }
54
55      public void setCreateBy(Long createBy) {
56          this.createBy = createBy;
57      }
58
59      public LocalDateTime getCreateTime() {
60          return createTime;
61      }
62
63      public void setCreateTime(LocalDateTime createTime) {
64          this.createTime = createTime;
65      }
66
67      public Long getUpdateBy() {
```

```
68          return updateBy;
69      }
70
71      public void setUpdateBy(Long updateBy) {
72          this.updateBy = updateBy;
73      }
74
75      public LocalDateTime getUpdateTime() {
76          return updateTime;
77      }
78
79      public void setUpdateTime(LocalDateTime updateTime) {
80          this.updateTime = updateTime;
81      }
82  }
```

代码优化

提取公共类

- 创建oj-common-core

先创建oj-common工程，再创建oj-common-core。

这个公共的类，将在多个微服务中被使用，所以最合适的位置应该是common包中。所以我们将这个类放在com.bite.common.core.domain.entity中。后续这个类将作为所有实体类的父类。

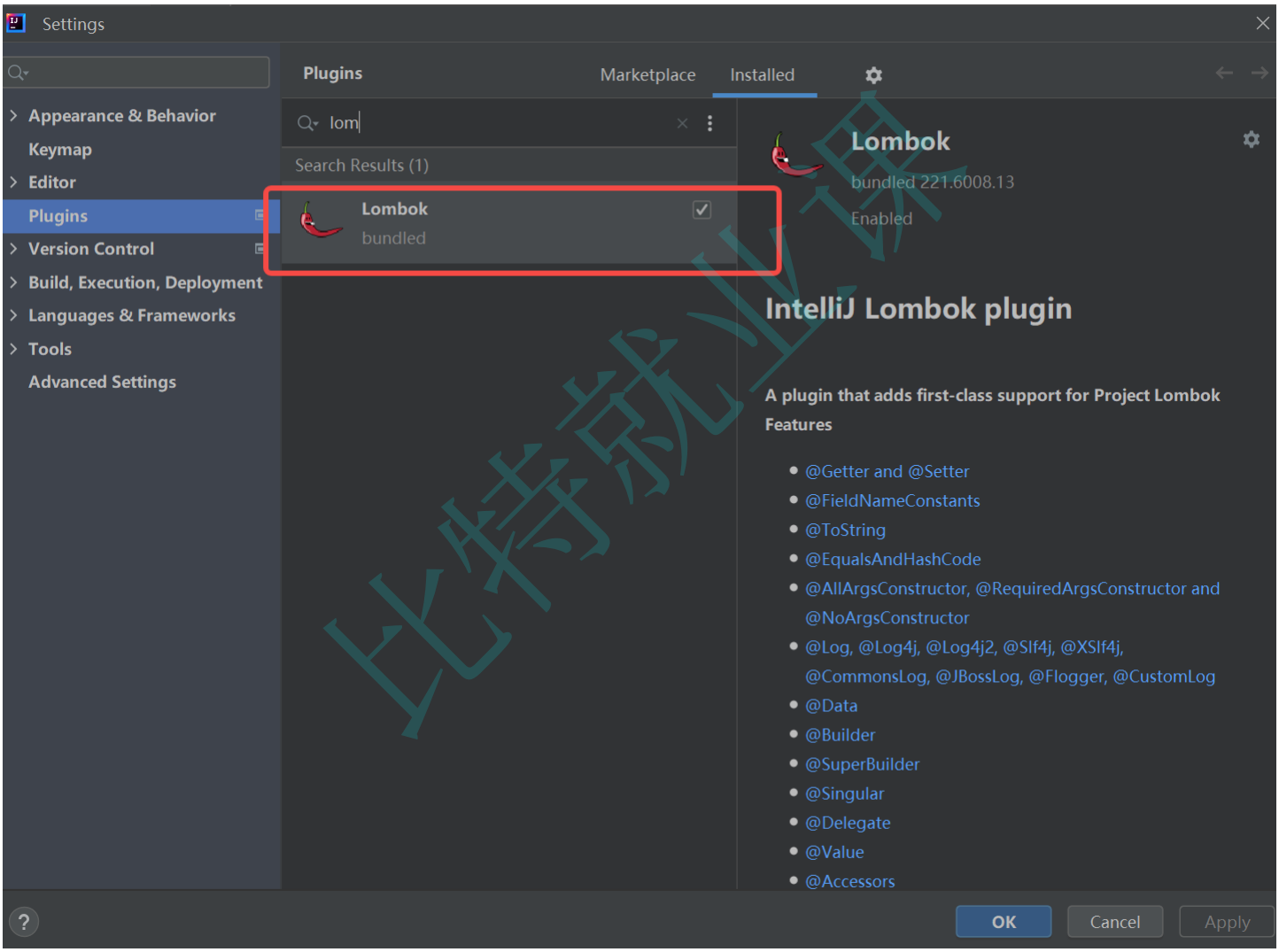
```
1  @Data
2  public class BaseEntity implements Serializable {
3
4      private static final long serialVersionUID = 1L;
5
6      /**
7       * 创建者
8       */
9      private Long createBy;
10
11     /**
12      * 创建时间
13      */
14     private LocalDateTime createTime;
15
16     /**
17      * 更新者
18      */
19     private Long updateBy;
```

```
20
21      /**
22       * 更新时间
23       */
24      private LocalDateTime updateTime;
25  }
```

引入lombok

lombok安装

同学们可以检查下lombok插件是否已经安装。



- **lombok依赖导入**

```
1  <dependency>
2      <groupId>org.projectlombok</groupId>
3      <artifactId>lombok</artifactId>
4  </dependency>
```