

```
-- 用户认证相关表
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  username VARCHAR(50) UNIQUE NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  display_name VARCHAR(100),
  phone VARCHAR(20),
  avatar_url VARCHAR(255),
  department_id INTEGER,
  role_id INTEGER,
  status VARCHAR(20) DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'locked')),
  last_login_at TIMESTAMP,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  created_by INTEGER
);
COMMENT ON TABLE users IS '系统用户表';
COMMENT ON COLUMN users.username IS '用户名';
COMMENT ON COLUMN users.email IS '邮箱';
COMMENT ON COLUMN users.status IS '状态: active-激活, inactive-未激活, locked-锁定';
CREATE TABLE roles (
  id SERIAL PRIMARY KEY,
  name VARCHAR(50) UNIQUE NOT NULL,
  description TEXT,
  permissions JSONB,
  data_scope VARCHAR(20) DEFAULT 'all' CHECK (data_scope IN ('all', 'department', 'self')),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE roles IS '角色表';
COMMENT ON COLUMN roles.permissions IS '权限配置 (JSON 格式)';
COMMENT ON COLUMN roles.data_scope IS '数据权限范围: all-全部, department-本部门, self-仅自己';
CREATE TABLE departments (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  code VARCHAR(50) UNIQUE NOT NULL,
  parent_id INTEGER REFERENCES departments(id),
  manager_id INTEGER REFERENCES users(id),
  description TEXT,
  sort_order INTEGER DEFAULT 0,
  status VARCHAR(20) DEFAULT 'active' CHECK (status IN ('active', 'inactive')),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE departments IS '部门表';
-- 客户管理相关表
CREATE TABLE customers (
  id SERIAL PRIMARY KEY,
  customer_code VARCHAR(50) UNIQUE NOT NULL,
  name VARCHAR(100) NOT NULL,
  type VARCHAR(20) DEFAULT 'individual' CHECK (type IN ('individual', 'enterprise')),
  contact_person VARCHAR(100),
  phone VARCHAR(20),
```

```

    email VARCHAR(100),
    company VARCHAR(100),
    industry VARCHAR(50),
    source VARCHAR(50),
    risk_level VARCHAR(20) DEFAULT 'medium' CHECK (risk_level IN ('low', 'medium', 'high')),
    value_level VARCHAR(20) DEFAULT 'medium' CHECK (value_level IN ('low', 'medium', 'high', 'vip')),
    total_assets DECIMAL(15,2),
    investment_preferences JSONB,
    status VARCHAR(20) DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'potential')),
    created_by INTEGER REFERENCES users(id),
    assigned_to INTEGER REFERENCES users(id),
    data_level VARCHAR(20) DEFAULT 'public' CHECK (data_level IN ('public', 'internal', 'confidential', 'secret')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE customers IS '客户表';
COMMENT ON COLUMN customers.customer_code IS '客户编号';
COMMENT ON COLUMN customers.type IS '客户类型: individual-个人, enterprise-企业';
COMMENT ON COLUMN customers.investment_preferences IS '投资偏好 (JSON 格式)';
COMMENT ON COLUMN customers.data_level IS '数据密级: public-公开, internal-内部, confidential-机密, secret-秘密';
CREATE TABLE customer_tags (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    color VARCHAR(20),
    description TEXT,
    category VARCHAR(50),
    sort_order INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE customer_tags IS '客户标签定义表';
CREATE TABLE customer_tag_relations (
    id SERIAL PRIMARY KEY,
    customer_id INTEGER NOT NULL REFERENCES customers(id) ON DELETE CASCADE,
    tag_id INTEGER NOT NULL REFERENCES customer_tags(id) ON DELETE CASCADE,
    created_by INTEGER REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(customer_id, tag_id)
);
COMMENT ON TABLE customer_tag_relations IS '客户标签关联表';
CREATE TABLE customer_histories (
    id SERIAL PRIMARY KEY,
    customer_id INTEGER NOT NULL REFERENCES customers(id) ON DELETE CASCADE,
    type VARCHAR(50) NOT NULL,
    content TEXT,
    metadata JSONB,
    created_by INTEGER REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE customer_histories IS '客户历史记录表';
COMMENT ON COLUMN customer_histories.type IS '记录类型: call-电话, meeting-会议, email-邮件, note-备注, update-更新';
-- 量化分析相关表

```

```

CREATE TABLE analysis_models (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  type VARCHAR(50) NOT NULL,
  description TEXT,
  parameters JSONB,
  version VARCHAR(20),
  status VARCHAR(20) DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'testing')),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE analysis_models IS '分析模型表';
COMMENT ON COLUMN analysis_models.type IS '模型类型: value-价值分析, risk-风险评估, demand-需求预测';
CREATE TABLE customer_analyses (
  id SERIAL PRIMARY KEY,
  customer_id INTEGER NOT NULL REFERENCES customers(id) ON DELETE CASCADE,
  model_id INTEGER NOT NULL REFERENCES analysis_models(id),
  analysis_type VARCHAR(50) NOT NULL,
  input_data JSONB,
  output_data JSONB,
  score DECIMAL(5,2),
  confidence DECIMAL(5,4),
  insights TEXT,
  recommendations JSONB,
  status VARCHAR(20) DEFAULT 'completed' CHECK (status IN ('pending', 'processing', 'completed', 'failed')),
  analyzed_by INTEGER REFERENCES users(id),
  analyzed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE customer_analyses IS '客户分析结果表';
COMMENT ON COLUMN customer_analyses.analysis_type IS '分析类型: value-价值分析, risk-风险评估, demand-需求预测';
CREATE TABLE analysis_reports (
  id SERIAL PRIMARY KEY,
  customer_id INTEGER NOT NULL REFERENCES customers(id) ON DELETE CASCADE,
  title VARCHAR(200) NOT NULL,
  report_type VARCHAR(50) NOT NULL,
  content JSONB,
  summary TEXT,
  generated_by INTEGER REFERENCES users(id),
  generated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  expires_at TIMESTAMP,
  status VARCHAR(20) DEFAULT 'draft' CHECK (status IN ('draft', 'published', 'archived')),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE analysis_reports IS '分析报告表';
-- 服务流程相关表
CREATE TABLE projects (
  id SERIAL PRIMARY KEY,
  project_code VARCHAR(50) UNIQUE NOT NULL,
  customer_id INTEGER NOT NULL REFERENCES customers(id),
  name VARCHAR(200) NOT NULL,

```

```

        description TEXT,
        type VARCHAR(50),
        status VARCHAR(20) DEFAULT 'planning' CHECK (status IN ('planning', 'in_progress', 'paused', 'completed', 'cancelled')),
        priority VARCHAR(20) DEFAULT 'medium' CHECK (priority IN ('low', 'medium', 'high', 'urgent')),
        start_date DATE,
        end_date DATE,
        actual_start_date DATE,
        actual_end_date DATE,
        budget DECIMAL(15,2),
        actual_cost DECIMAL(15,2),
        manager_id INTEGER REFERENCES users(id),
        progress INTEGER DEFAULT 0 CHECK (progress >= 0 AND progress <= 100),
        created_by INTEGER REFERENCES users(id),
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    );
    COMMENT ON TABLE projects IS ' 咨询项目表';
    CREATE TABLE project_phases (
        id SERIAL PRIMARY KEY,
        project_id INTEGER NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
        name VARCHAR(100) NOT NULL,
        description TEXT,
        phase_order INTEGER NOT NULL,
        status VARCHAR(20) DEFAULT 'pending' CHECK (status IN ('pending', 'in_progress', 'completed', 'blocked')),
        start_date DATE,
        end_date DATE,
        actual_start_date DATE,
        actual_end_date DATE,
        assigned_to INTEGER REFERENCES users(id),
        progress INTEGER DEFAULT 0 CHECK (progress >= 0 AND progress <= 100),
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    );
    COMMENT ON TABLE project_phases IS ' 项目阶段表';
    CREATE TABLE service_records (
        id SERIAL PRIMARY KEY,
        customer_id INTEGER NOT NULL REFERENCES customers(id) ON DELETE CASCADE,
        project_id INTEGER REFERENCES projects(id),
        service_type VARCHAR(50) NOT NULL,
        content TEXT,
        duration_minutes INTEGER,
        service_date TIMESTAMP,
        next_followup_date DATE,
        satisfaction_score INTEGER CHECK (satisfaction_score >= 1 AND satisfaction_score <= 5),
        feedback TEXT,
        created_by INTEGER REFERENCES users(id),
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    );
    COMMENT ON TABLE service_records IS ' 服务记录表';
    CREATE TABLE reminders (
        id SERIAL PRIMARY KEY,
        title VARCHAR(200) NOT NULL,

```

```

description TEXT,
type VARCHAR(50) NOT NULL,
related_type VARCHAR(50),
related_id INTEGER,
due_date TIMESTAMP NOT NULL,
priority VARCHAR(20) DEFAULT 'medium' CHECK (priority IN ('low', 'medium', 'high')),
status VARCHAR(20) DEFAULT 'pending' CHECK (status IN ('pending', 'completed', 'dismissed')),
assigned_to INTEGER REFERENCES users(id),
created_by INTEGER REFERENCES users(id),
completed_at TIMESTAMP,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE reminders IS '提醒表';
COMMENT ON COLUMN reminders.related_type IS '关联类型: project-项目, customer-客户, phase-阶段';
-- 系统设置相关表
CREATE TABLE system_settings (
id SERIAL PRIMARY KEY,
setting_key VARCHAR(100) UNIQUE NOT NULL,
setting_value JSONB,
setting_type VARCHAR(50) DEFAULT 'string',
description TEXT,
category VARCHAR(50),
is_public BOOLEAN DEFAULT false,
updated_by INTEGER REFERENCES users(id),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE system_settings IS '系统设置表';
CREATE TABLE notification_templates (
id SERIAL PRIMARY KEY,
name VARCHAR(100) NOT NULL,
type VARCHAR(50) NOT NULL,
subject_template TEXT,
content_template TEXT,
variables JSONB,
status VARCHAR(20) DEFAULT 'active' CHECK (status IN ('active', 'inactive')),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE notification_templates IS '通知模板表';
-- 安全日志相关表
CREATE TABLE operation_logs (
id SERIAL PRIMARY KEY,
user_id INTEGER REFERENCES users(id),
operation_type VARCHAR(50) NOT NULL,
resource_type VARCHAR(50),
resource_id INTEGER,
description TEXT,
old_data JSONB,
new_data JSONB,
ip_address INET,
user_agent TEXT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

```

```

);
COMMENT ON TABLE operation_logs IS ' 操作日志表';
CREATE TABLE login_logs (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(id),
  login_type VARCHAR(20) DEFAULT 'password' CHECK (login_type IN ('password', 'sso', 'to-
ken')),
  ip_address INET,
  user_agent TEXT,
  success BOOLEAN DEFAULT true,
  failure_reason TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE login_logs IS ' 登录日志表';
CREATE TABLE data_access_logs (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(id),
  data_type VARCHAR(50) NOT NULL,
  data_id INTEGER NOT NULL,
  operation VARCHAR(20) NOT NULL CHECK (operation IN ('create', 'read', 'update', 'delete', 'ex-
port')),
  ip_address INET,
  user_agent TEXT,
  accessed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
COMMENT ON TABLE data_access_logs IS ' 数据访问日志表';
-- 创建索引
CREATE INDEX idx_users_username ON users(username);
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_department_id ON users(department_id);
CREATE INDEX idx_users_role_id ON users(role_id);
CREATE INDEX idx_customers_customer_code ON customers(customer_code);
CREATE INDEX idx_customers_name ON customers(name);
CREATE INDEX idx_customers_type ON customers(type);
CREATE INDEX idx_customers_risk_level ON customers(risk_level);
CREATE INDEX idx_customers_value_level ON customers(value_level);
CREATE INDEX idx_customers_assigned_to ON customers(assigned_to);
CREATE INDEX idx_customers_data_level ON customers(data_level);
CREATE INDEX idx_customer_tag_relations_customer_id ON customer_tag_relations(customer_id);
CREATE INDEX idx_customer_tag_relations_tag_id ON customer_tag_relations(tag_id);
CREATE INDEX idx_customer_histories_customer_id ON customer_histories(customer_id);
CREATE INDEX idx_customer_histories_type ON customer_histories(type);
CREATE INDEX idx_customer_histories_created_at ON customer_histories(created_at);
CREATE INDEX idx_customer_analyses_customer_id ON customer_analyses(customer_id);
CREATE INDEX idx_customer_analyses_model_id ON customer_analyses(model_id);
CREATE INDEX idx_customer_analyses_analysis_type ON customer_analyses(analysis_type);
CREATE INDEX idx_projects_project_code ON projects(project_code);
CREATE INDEX idx_projects_customer_id ON projects(customer_id);
CREATE INDEX idx_projects_status ON projects(status);
CREATE INDEX idx_projects_manager_id ON projects(manager_id);
CREATE INDEX idx_project_phases_project_id ON project_phases(project_id);
CREATE INDEX idx_project_phases_status ON project_phases(status);
CREATE INDEX idx_service_records_customer_id ON service_records(customer_id);
CREATE INDEX idx_service_records_project_id ON service_records(project_id);
CREATE INDEX idx_reminders_assigned_to ON reminders(assigned_to);

```

```
CREATE INDEX idx_reminders_due_date ON reminders(due_date);
CREATE INDEX idx_reminders_status ON reminders(status);
CREATE INDEX idx_operation_logs_user_id ON operation_logs(user_id);
CREATE INDEX idx_operation_logs_operation_type ON operation_logs(operation_type);
CREATE INDEX idx_operation_logs_created_at ON operation_logs(created_at);
CREATE INDEX idx_login_logs_user_id ON login_logs(user_id);
CREATE INDEX idx_login_logs_created_at ON login_logs(created_at);
CREATE INDEX idx_data_access_logs_user_id ON data_access_logs(user_id);
CREATE INDEX idx_data_access_logs_data_type ON data_access_logs(data_type);
CREATE INDEX idx_data_access_logs_accessed_at ON data_access_logs(accessed_at);
-- 添加外键约束
ALTER TABLE users ADD CONSTRAINT fk_users_department_id
    FOREIGN KEY (department_id) REFERENCES departments(id);
ALTER TABLE users ADD CONSTRAINT fk_users_role_id
    FOREIGN KEY (role_id) REFERENCES roles(id);
ALTER TABLE customers ADD CONSTRAINT fk_customers_created_by
    FOREIGN KEY (created_by) REFERENCES users(id);
ALTER TABLE customers ADD CONSTRAINT fk_customers_assigned_to
    FOREIGN KEY (assigned_to) REFERENCES users(id);
-- 创建迁移版本表
CREATE TABLE IF NOT EXISTS schema_migrations (
    version VARCHAR(255) PRIMARY KEY,
    applied_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
-- 迁移脚本：初始化数据库结构
DO $$
BEGIN
    -- 检查是否已经执行过初始化
    IF NOT EXISTS (SELECT 1 FROM schema_migrations WHERE version = '20240101000000_init') THEN
        -- 所有建表语句已在 all_tables.sql 中定义
        -- 这里只记录迁移版本
        INSERT INTO schema_migrations (version) VALUES ('20240101000000_init');
    END IF;
END $$;
-- 迁移脚本：添加数据权限控制字段
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM schema_migrations WHERE version = '20240101000001_add_data_permissions') THEN
        -- 检查并添加数据权限相关字段
        IF NOT EXISTS (SELECT 1 FROM information_schema.columns
            WHERE table_name = 'roles' AND column_name = 'data_scope') THEN
            ALTER TABLE roles ADD COLUMN data_scope VARCHAR(20) DEFAULT 'all'
            CHECK (data_scope IN ('all', 'department', 'self'));
        END IF;
        INSERT INTO schema_migrations (version) VALUES ('20240101000001_add_data_permissions');
    END IF;
END $$;
-- 迁移脚本：添加量化分析模型参数
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM schema_migrations WHERE version = '20240101000002_add_analysis_parameters') THEN
        -- 检查并添加分析模型参数字段
        IF NOT EXISTS (SELECT 1 FROM information_schema.columns
            WHERE table_name = 'analysis_models' AND column_name = 'parameters') THEN
            ALTER TABLE analysis_models ADD COLUMN parameters JSONB;
        END IF;
    END IF;
END $$;
```

```

        END IF;
        INSERT INTO schema_migrations (version) VALUES ('20240101000002_add_analysis_parameters');
    END IF;
END $$;
-- 创建更新时间的触发器函数
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ language 'plpgsql';
-- 为需要自动更新时间的表创建触发器
DO $$
DECLARE
    table_name text;
BEGIN
    FOR table_name IN
        SELECT tablename FROM pg_tables
        WHERE schemaname = 'public'
        AND tablename IN ('users', 'roles', 'departments', 'customers', 'customer_tags',
            'analysis_models', 'analysis_reports', 'projects', 'project_phases',
            'reminders', 'system_settings', 'notification_templates')
    LOOP
        EXECUTE format('
            DROP TRIGGER IF EXISTS update_%s_updated_at ON %s;
            CREATE TRIGGER update_%s_updated_at
            BEFORE UPDATE ON %s
            FOR EACH ROW
            EXECUTE FUNCTION update_updated_at_column();
        ', table_name, table_name, table_name, table_name);
    END LOOP;
END $$;
-- 初始化角色数据
INSERT INTO roles (name, description, permissions, data_scope) VALUES
('超级管理员', '系统最高权限管理员', '{"all": true}', 'all'),
('部门经理', '部门管理权限', '{"customer": ["read", "write", "delete"], "project": ["read", "write"], "analysis": ["read", "write"]}', 'department'),
('投资顾问', '客户服务和咨询权限', '{"customer": ["read", "write"], "project": ["read", "write"], "analysis": ["read"]}', 'self'),
('数据分析师', '数据分析权限', '{"customer": ["read"], "analysis": ["read", "write"]}', 'department')
ON CONFLICT (name) DO NOTHING;
-- 初始化部门数据
INSERT INTO departments (name, code, parent_id, description) VALUES
('总部', 'HQ', NULL, '公司总部'),
('投资咨询部', 'INVESTMENT', 1, '投资咨询业务部门'),
('数据分析部', 'ANALYTICS', 1, '数据分析和模型开发部门'),
('客户服务部', 'SERVICE', 1, '客户服务和关系维护部门')
ON CONFLICT (code) DO NOTHING;
-- 初始化用户数据
INSERT INTO users (username, email, password_hash, display_name, phone, department_id, role_id, status) VALUES
('admin', 'admin@quantcrm.com', '$2b$10$ExampleHashForDemoOnly', '系统管理员', '13800138000', 1, 1, 'active'),

```



```
( 'zhangwei', 'zhangwei@quantcrm.com', '$2b$10$ExampleHashForDemoOnly', '张伟', '13800138001', 2, 2, 'active'),
('limei', 'limei@quantcrm.com', '$2b$10$ExampleHashForDemoOnly', '李梅', '13800138002', 3, 4, 'active'),
('wanggang', 'wanggang@quantcrm.com', '$2b$10$ExampleHashForDemoOnly', '王刚', '4', 3, 'active')
ON CONFLICT (username) DO NOTHING;
-- 更新部门经理信息
UPDATE departments SET manager_id = 2 WHERE id = 2;
UPDATE departments SET manager_id = 3 WHERE id = 3;
UPDATE departments SET manager_id = 4 WHERE id = 4;
-- 初始化客户标签
INSERT INTO customer_tags (name, color, description, category) VALUES
('高净值客户', '#FF6B6B', '资产规模超过 1000 万', '价值等级'),
('风险偏好型', '#4ECDC4', '偏好高风险高收益投资', '风险偏好'),
('稳健型', '#45B7D1', '偏好低风险稳定收益', '风险偏好'),
('企业客户', '#96CEB4', '企业机构客户', '客户类型'),
('潜在 VIP', '#FFEEA7', '有潜力成为 VIP 客户', '发展潜力')
ON CONFLICT (name) DO NOTHING;
-- 初始化客户数据
INSERT INTO customers (customer_code, name, type, contact_person, phone, email, company, industry, risk_level, value_level, total_assets, investment_preferences, assigned_to, data_level) VALUES
('CUST001', '张三', 'individual', '张三', '13900139001', 'zhangsan@example.com', NULL, '金融', 'medium', 'high', 5000000.00, '{"preferred_products":["股票","基金"],"risk_tolerance":"medium","investment_horizon":"long_term"}', 2, 'confidential'),
('CUST002', '李四科技', 'enterprise', '李四', '13900139002', 'lisi@tech.com', '李四科技有限公司', '科技', 'low', 'vip', 20000000.00, '{"preferred_products":["债券","理财产品"],"risk_tolerance":"low","investment_horizon":"medium_term"}', 2, 'secret'),
('CUST003', '王五', 'individual', '王五', '13900139003', 'wangwu@example.com', NULL, '制造业', 'high', 'medium', 2000000.00, '{"preferred_products":["期货","外汇"],"risk_tolerance":"high","investment_horizon":"short_term"}', 4, 'internal'),
('CUST004', '赵六投资', 'enterprise', '赵六', '13900139004', 'zhaoliu@invest.com', '赵六投资有限公司', '投资', 'medium', 'high', 15000000.00, '{"preferred_products":["私募","股权"],"risk_tolerance":"medium","investment_horizon":"long_term"}', 2, 'confidential')
ON CONFLICT (customer_code) DO NOTHING;
-- 初始化客户标签关联
INSERT INTO customer_tag_relations (customer_id, tag_id, created_by) VALUES
(1, 1, 1), (1, 2, 1),
(2, 1, 1), (2, 3, 1), (2, 4, 1),
(3, 2, 1),
(4, 1, 1), (4, 4, 1), (4, 5, 1)
ON CONFLICT (customer_id, tag_id) DO NOTHING;
-- 初始化分析模型
INSERT INTO analysis_models (name, type, description, parameters, version, status) VALUES
('客户价值评估模型', 'value', '基于客户资产、交易行为和潜力的综合价值评估', '{"weight_assets": 0.4, "weight_transactions": 0.6, "weight_risk": 0.2, "weight_demand": 0.2, "weight_growth": 0.1, "weight_innovation": 0.1, "weight_social": 0.1, "weight_environmental": 0.1, "weight_governance": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.1, "weight_integrity": 0.1, "weight_honesty": 0.1, "weight_courage": 0.1, "weight_compassion": 0.1, "weight_humility": 0.1, "weight_gratitude": 0.1, "weight_patience": 0.1, "weight_persistence": 0.1, "weight_determination": 0.1, "weight_resilience": 0.1, "weight_adaptability": 0.1, "weight_inclusiveness": 0.1, "weight_sustainability": 0.1, "weight_responsible": 0.1, "weight_transparency": 0.1, "weight_accountability": 0.
```

```

(1, 1, 'value', 85.50, 0.92, '客户资产稳定增长, 投资经验丰富', ['推荐私募产品', '定期资产配置调整']),
(1, 2, 'risk', 65.30, 0.88, '中等风险偏好, 能承受一定市场波动', ['建议控制杠杆比例', '分散投资']),
(2, 1, 'value', 92.10, 0.95, '企业客户, 资金规模大, 稳定性高', ['定制化投资方案', '专属投资顾问']),
(2, 2, 'risk', 35.20, 0.90, '低风险偏好, 注重资金安全', ['推荐稳健型产品', '保本理财'])
ON CONFLICT DO NOTHING;
-- 初始化咨询项目
INSERT INTO projects (project_code, customer_id, name, description, type, status, priority, start_date, end_date, manager_id, progress) VALUES
('PROJ001', 1, '张三资产配置优化', '基于市场变化的资产重新配置', 'asset_allocation', 'in_progress', 'high', '2024-01-15', '2024-03-15', 2, 60),
('PROJ002', 2, '李四科技企业理财', '企业闲置资金理财方案设计', 'corporate_finance', 'planning', 'medium', '2024-02-01', '2024-04-01', 2, 20),
('PROJ003', 4, '赵六投资海外配置', '海外资产配置和风险管理', 'international', 'in_progress', 'high', '2024-01-20', '2024-05-20', 2, 40)
ON CONFLICT (project_code) DO NOTHING;
-- 初始化项目阶段
INSERT INTO project_phases (project_id, name, description, phase_order, status, start_date, end_date, assigned_to, progress) VALUES
(1, '需求分析', '深入了解客户需求和风险偏好', 1, 'completed', '2024-01-15', '2024-01-25', 2, 100),
(1, '方案设计', '制定个性化资产配置方案', 2, 'in_progress', '2024-01-26', '2024-02-15', 2, 80),
(1, '方案实施', '执行资产配置调整', 3, 'pending', '2024-02-16', '2024-03-01', 2, 0),
(1, '跟踪优化', '定期跟踪和方案优化', 4, 'pending', '2024-03-02', '2024-03-15', 2, 0)
ON CONFLICT DO NOTHING;
-- 初始化服务记录
INSERT INTO service_records (customer_id, project_id, service_type, content, duration_minutes, service_date, next_followup_date, satisfaction_score) VALUES
(1, 1, 'meeting', '讨论当前市场形势和资产配置调整需求', 60, '2024-01-15 14:00:00', '2024-02-01', 5),
(1, 1, 'call', '确认资产配置方案细节', 30, '2024-01-28 10:30:00', '2024-02-15', 4),
(2, NULL, 'consultation', '企业理财需求初步沟通', 45, '2024-01-20 15:00:00', '2024-02-05', 5)
ON CONFLICT DO NOTHING;
-- 初始化系统设置
INSERT INTO system_settings (setting_key, setting_value, setting_type, description, category, is_public) VALUES
('platform.name', '量化咨询 CRM', 'string', '平台名称', 'general', true),
('platform.logo', '/assets/logo.png', 'string', '平台 Logo', 'general', true),
('analysis.risk_threshold', '0.7', 'number', '风险阈值', 'analysis', false),
('notification.email_enabled', 'true', 'boolean', '邮件通知开关', 'notification', false),
('security.password_expiry_days', '90', 'number', '密码有效期(天)', 'security', false)
ON CONFLICT (setting_key) DO NOTHING;
-- 初始化提醒
INSERT INTO reminders (title, description, type, related_type, related_id, due_date, priority, assigned_to) VALUES
('张三方案设计截止', '完成张三资产配置方案设计', 'deadline', 'project', 1, '2024-02-15 18:00:00', 'high', 2),
('李四初次沟通跟进', '跟进李四科技企业理财需求', 'followup', 'customer', 2, '2024-02-05 09:00:00', 'medium', 2),
('赵六海外市场分析', '准备海外市场分析报告', 'task', 'project', 3, '2024-02-10 17:00:00', 'medium', 2)
ON CONFLICT DO NOTHING;
package com.quantcrm.system.model;
import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.Set;
@Entity
@Table(name = "system_user")
public class User {

```

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@Column(unique = true, nullable = false)
private String username;
@Column(nullable = false)
private String password;
@Column(nullable = false)
private String realName;
@Column(unique = true)
private String email;
@Column(unique = true)
private String phone;
@ManyToOne
@JoinColumn(name = "department_id")
private Department department;
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(
    name = "user_roles",
    joinColumns = @JoinColumn(name = "user_id"),
    inverseJoinColumns = @JoinColumn(name = "role_id")
)
private Set<Role> roles;
@Column(nullable = false)
private Boolean enabled = true;
@Column(nullable = false)
private LocalDateTime createTime;
private LocalDateTime lastLoginTime;
@Column(nullable = false)
private String createBy;
private String updateBy;
private LocalDateTime updateTime;
// Getters and Setters
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }
public String getUsername() { return username; }
public void setUsername(String username) { this.username = username; }
public String getPassword() { return password; }
public void setPassword(String password) { this.password = password; }
public String getRealName() { return realName; }
public void setRealName(String realName) { this.realName = realName; }
public String getEmail() { return email; }
public void setEmail(String email) { this.email = email; }
public String getPhone() { return phone; }
public void setPhone(String phone) { this.phone = phone; }
public Department getDepartment() { return department; }
public void setDepartment(Department department) { this.department = department; }
public Set<Role> getRoles() { return roles; }
public void setRoles(Set<Role> roles) { this.roles = roles; }
public Boolean getEnabled() { return enabled; }
public void setEnabled(Boolean enabled) { this.enabled = enabled; }
public LocalDateTime getCreateTime() { return createTime; }
public void setCreateTime(LocalDateTime createTime) { this.createTime = createTime; }
public LocalDateTime getLastLoginTime() { return lastLoginTime; }
public void setLastLoginTime(LocalDateTime lastLoginTime) { this.lastLoginTime = last-
```

```
LoginTime; }
    public String getCreateBy() { return createBy; }
    public void setCreateBy(String createBy) { this.createBy = createBy; }
    public String getUpdateBy() { return updateBy; }
    public void setUpdateBy(String updateBy) { this.updateBy = updateBy; }
    public LocalDateTime getUpdateTime() { return updateTime; }
    public void setUpdateTime(LocalDateTime updateTime) { this.updateTime = updateTime; }
}
package com.quantcrm.system.auth.model;
import jakarta.validation.constraints.NotBlank;
public class LoginRequest {
    @NotBlank
    private String username;
    @NotBlank
    private String password;
    public LoginRequest() {}
    public LoginRequest(String username, String password) {
        this.username = username;
        this.password = password;
    }
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}
package com.quantcrm.system.auth.model;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Size;
public class RegisterRequest {
    @NotBlank
    @Size(min = 3, max = 50)
    private String username;
    @NotBlank
    @Size(max = 100)
    @Email
    private String email;
    @NotBlank
    @Size(min = 6, max = 100)
    private String password;
    @Size(max = 20)
    private String phone;
    public RegisterRequest() {}
    public RegisterRequest(String username, String email, String password) {
        this.username = username;
        this.email = email;
        this.password = password;
    }
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
    public String getPhone() { return phone; }
```

```
        public void setPhone(String phone) { this.phone = phone; }
    }
    package com.quantcrm.system.auth.model;
    import jakarta.validation.constraints.Email;
    import jakarta.validation.constraints.NotBlank;
    public class ForgotPasswordRequest {
        @NotBlank
        @Email
        private String email;
        public ForgotPasswordRequest() {}
        public ForgotPasswordRequest(String email) {
            this.email = email;
        }
        public String getEmail() { return email; }
        public void setEmail(String email) { this.email = email; }
    }
    package com.quantcrm.system.auth.model;
    import jakarta.validation.constraints.NotBlank;
    import jakarta.validation.constraints.Size;
    public class ResetPasswordRequest {
        @NotBlank
        private String token;
        @NotBlank
        @Size(min = 6, max = 100)
        private String newPassword;
        public ResetPasswordRequest() {}
        public ResetPasswordRequest(String token, String newPassword) {
            this.token = token;
            this.newPassword = newPassword;
        }
        public String getToken() { return token; }
        public void setToken(String token) { this.token = token; }
        public String getNewPassword() { return newPassword; }
        public void setNewPassword(String newPassword) { this.newPassword = newPassword; }
    }
    package com.quantcrm.system.auth.model;
    public class AuthResponse {
        private String token;
        private String type = "Bearer";
        private Long id;
        private String username;
        private String email;
        public AuthResponse(String token, Long id, String username, String email) {
            this.token = token;
            this.id = id;
            this.username = username;
            this.email = email;
        }
        public String getToken() { return token; }
        public void setToken(String token) { this.token = token; }
        public String getType() { return type; }
        public void setType(String type) { this.type = type; }
        public Long getId() { return id; }
        public void setId(Long id) { this.id = id; }
        public String getUsername() { return username; }
```

```

    public void setUsername(String username) { this.username = username; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
}
package com.quantcrm.system.auth.model;
public class ApiResponse {
    private boolean success;
    private String message;
    private Object data;
    public ApiResponse(boolean success, String message) {
        this.success = success;
        this.message = message;
    }
    public ApiResponse(boolean success, String message, Object data) {
        this.success = success;
        this.message = message;
        this.data = data;
    }
    public boolean isSuccess() { return success; }
    public void setSuccess(boolean success) { this.success = success; }
    public String getMessage() { return message; }
    public void setMessage(String message) { this.message = message; }
    public Object getData() { return data; }
    public void setData(Object data) { this.data = data; }
}
package com.quantcrm.system.repository;
import com.quantcrm.system.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import java.util.List;
import java.util.Optional;
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    Optional<User> findByUsername(String username);
    Boolean existsByUsername(String username);
    Boolean existsByEmail(String email);
    Boolean existsByPhone(String phone);
    @Query("SELECT u FROM User u WHERE u.department.id = :departmentId AND u.enabled = true")
    List<User> findByDepartmentId(@Param("departmentId") Long departmentId);
    @Query("SELECT u FROM User u WHERE u.enabled = :enabled")
    List<User> findByEnabled(@Param("enabled") Boolean enabled);
    @Query("SELECT u FROM User u JOIN u.roles r WHERE r.id = :roleId")
    List<User> findByRoleId(@Param("roleId") Long roleId);
}
package com.quantcrm.system.auth.service;
import com.quantcrm.system.auth.model.*;
import org.springframework.security.core.Authentication;
public interface AuthService {
    AuthResponse authenticateUser(LoginRequest loginRequest);
    ApiResponse registerUser(RegisterRequest registerRequest);
    ApiResponse forgotPassword(ForgotPasswordRequest forgotPasswordRequest);
    ApiResponse resetPassword(ResetPasswordRequest resetPasswordRequest);
    User getCurrentUser(Authentication authentication);
}

```

```
}
package com.quantcrm.system.auth.service;
import com.quantcrm.system.auth.model.*;
import com.quantcrm.system.auth.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import java.time.LocalDateTime;
import java.util.Optional;
import java.util.UUID;
@Service
public class AuthServiceImpl implements AuthService {
    @Autowired
    private AuthenticationManager authenticationManager;
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Autowired
    private JwtTokenProvider tokenProvider;
    @Override
    public AuthResponse authenticateUser(LoginRequest loginRequest) {
        Authentication authentication = authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(
                loginRequest.getUsername(),
                loginRequest.getPassword()
            )
        );
        SecurityContextHolder.getContext().setAuthentication(authentication);
        String jwt = tokenProvider.generateToken(authentication);
        User user = (User) authentication.getPrincipal();
        user.setLastLogin(LocalDateTime.now());
        userRepository.save(user);
        return new AuthResponse(jwt, user.getId(), user.getUsername(), user.getEmail());
    }
    @Override
    public ApiResponse registerUser(RegisterRequest registerRequest) {
        if (userRepository.existsByUsername(registerRequest.getUsername())) {
            return new ApiResponse(false, "用户名已存在");
        }
        if (userRepository.existsByEmail(registerRequest.getEmail())) {
            return new ApiResponse(false, "邮箱已被注册");
        }
        User user = new User(
            registerRequest.getUsername(),
            registerRequest.getEmail(),
            passwordEncoder.encode(registerRequest.getPassword())
        );
        user.setPhone(registerRequest.getPhone());
        user.setVerificationCode(UUID.randomUUID().toString());
        userRepository.save(user);
    }
}
```

```

        return new ApiResponse(true, " 用户注册成功, 请等待管理员审核");
    }
    @Override
    public ApiResponse forgotPassword(ForgotPasswordRequest forgotPasswordRequest) {
        Optional<User> userOptional = userRepository.findByEmail(forgotPasswordRequest.getEmail());
        if (userOptional.isEmpty()) {
            return new ApiResponse(false, " 该邮箱未注册");
        }
        User user = userOptional.get();
        String resetToken = UUID.randomUUID().toString();
        user.setResetToken(resetToken);
        user.setResetTokenExpiry(LocalDate.now().plusHours(24));
        userRepository.save(user);
        return new ApiResponse(true, " 密码重置邮件已发送", resetToken);
    }
    @Override
    public ApiResponse resetPassword(ResetPasswordRequest resetPasswordRequest) {
        Optional<User> userOptional = userRepository.findByResetToken(resetPasswordRequest.getToken());
        if (userOptional.isEmpty()) {
            return new ApiResponse(false, " 无效的重置令牌");
        }
        User user = userOptional.get();
        if (user.getResetTokenExpiry().isBefore(LocalDate.now())) {
            return new ApiResponse(false, " 重置令牌已过期");
        }
        user.setPassword(passwordEncoder.encode(resetPasswordRequest.getNewPassword()));
        user.setResetToken(null);
        user.setResetTokenExpiry(null);
        userRepository.save(user);
        return new ApiResponse(true, " 密码重置成功");
    }
    @Override
    public User getCurrentUser(Authentication authentication) {
        return (User) authentication.getPrincipal();
    }
}
package com.quantcrm.system.auth.service;
import io.jsonwebtoken.*;
import io.jsonwebtoken.security.Keys;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Component;
import javax.crypto.SecretKey;
import java.util.Date;
@Component
public class JwtTokenProvider {
    @Value("${app.jwt.secret}")
    private String jwtSecret;
    @Value("${app.jwt.expiration}")
    private int jwtExpiration;
    private SecretKey getSigningKey() {
        return Keys.hmacShaKeyFor(jwtSecret.getBytes());
    }
    public String generateToken(Authentication authentication) {
        UserPrincipal userPrincipal = (UserPrincipal) authentication.getPrincipal();

```



```

        Date now = new Date();
        Date expiryDate = new Date(now.getTime() + jwtExpiration);
        return Jwts.builder()
            .setSubject(userPrincipal.getUsername())
            .setIssuedAt(now)
            .setExpiration(expiryDate)
            .signWith(getSigningKey(), SignatureAlgorithm.HS512)
            .compact();
    }

    public String getUsernameFromToken(String token) {
        Claims claims = Jwts.parserBuilder()
            .setSigningKey(getSigningKey())
            .build()
            .parseClaimsJws(token)
            .getBody();
        return claims.getSubject();
    }

    public boolean validateToken(String token) {
        try {
            Jwts.parserBuilder()
                .setSigningKey(getSigningKey())
                .build()
                .parseClaimsJws(token);
            return true;
        } catch (JwtException | IllegalArgumentException e) {
            return false;
        }
    }
}

package com.quantcrm.system.auth.service;
import com.quantcrm.system.auth.model.User;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import java.util.Collection;
import java.util.Collections;
public class UserPrincipal implements UserDetails {
    private User user;
    public UserPrincipal(User user) {
        this.user = user;
    }
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return Collections.singletonList(new SimpleGrantedAuthority("ROLE_USER"));
    }
    @Override
    public String getPassword() {
        return user.getPassword();
    }
    @Override
    public String getUsername() {
        return user.getUsername();
    }
    @Override
    public boolean isAccountNonExpired() {

```

```

        return true;
    }
    @Override
    public boolean isAccountNonLocked() {
        return user.getIsActive();
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
    public boolean isEnabled() {
        return user.getIsActive();
    }
    public Long getId() {
        return user.getId();
    }
    public String getEmail() {
        return user.getEmail();
    }
}

package com.quantcrm.system.auth.service;
import com.quantcrm.system.auth.model.User;
import com.quantcrm.system.auth.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class CustomUserDetailsService implements UserDetailsService {
    @Autowired
    private UserRepository userRepository;
    @Override
    @Transactional
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepository.findByUsername(username)
            .orElseThrow(() -> new UsernameNotFoundException(" 用户未找到: " + username));
        return new UserPrincipal(user);
    }
    @Transactional
    public UserDetails loadUserById(Long id) {
        User user = userRepository.findById(id)
            .orElseThrow(() -> new UsernameNotFoundException(" 用户未找到, ID: " + id));
        return new UserPrincipal(user);
    }
}

package com.quantcrm.system.auth.controller;
import com.quantcrm.system.auth.model.*;
import com.quantcrm.system.auth.service.AuthService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;

```

```

import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.*;
@RestController
@RequestMapping("/api/auth")
public class AuthController {
    @Autowired
    private AuthService authService;
    @PostMapping("/login")
    public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {
        AuthResponse authResponse = authService.authenticateUser(loginRequest);
        return ResponseEntity.ok(new ApiResponse(true, " 登录成功", authResponse));
    }
    @PostMapping("/register")
    public ResponseEntity<?> registerUser(@Valid @RequestBody RegisterRequest registerRequest) {
        ApiResponse response = authService.registerUser(registerRequest);
        return ResponseEntity.ok(response);
    }
    @PostMapping("/forgot-password")
    public ResponseEntity<?> forgotPassword(@Valid @RequestBody ForgotPasswordRequest forgotPasswordRequest) {
        ApiResponse response = authService.forgotPassword(forgotPasswordRequest);
        return ResponseEntity.ok(response);
    }
    @PostMapping("/reset-password")
    public ResponseEntity<?> resetPassword(@Valid @RequestBody ResetPasswordRequest resetPasswordRequest) {
        ApiResponse response = authService.resetPassword(resetPasswordRequest);
        return ResponseEntity.ok(response);
    }
    @GetMapping("/me")
    public ResponseEntity<?> getCurrentUser(Authentication authentication) {
        User user = authService.getCurrentUser(authentication);
        return ResponseEntity.ok(new ApiResponse(true, " 获取用户信息成功", user));
    }
}

package com.quantcrm.system.auth.util;
import com.quantcrm.system.auth.service.CustomUserDetailsService;
import com.quantcrm.system.auth.service.JwtTokenProvider;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;
import java.io.IOException;
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    @Autowired
    private JwtTokenProvider tokenProvider;

```

```

@Autowired
private CustomUserDetailsService customUserDetailsService;
@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
                                FilterChain filterChain) throws ServletException, IOException {
    try {
        String jwt = getJwtFromRequest(request);
        if (StringUtils.hasText(jwt) && tokenProvider.validateToken(jwt)) {
            String username = tokenProvider.getUsernameFromToken(jwt);
            UserDetails userDetails = customUserDetailsService.loadUserByUsername(username);
            UsernamePasswordAuthenticationToken authentication =
                new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
            authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
            SecurityContextHolder.getContext().setAuthentication(authentication);
        }
    } catch (Exception ex) {
        logger.error(" 无法设置用户认证", ex);
    }
    filterChain.doFilter(request, response);
}

private String getJwtFromRequest(HttpServletRequest request) {
    String bearerToken = request.getHeader("Authorization");
    if (StringUtils.hasText(bearerToken) && bearerToken.startsWith("Bearer ")) {
        return bearerToken.substring(7);
    }
    return null;
}
}

package com.quantcrm.system.auth.config;
import com.quantcrm.system.auth.service.CustomUserDetailsService;
import com.quantcrm.system.auth.util.JwtAuthenticationFilter;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;
import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
@Configuration
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true)
public class SecurityConfig {
    @Autowired
    private CustomUserDetailsService customUserDetailsService;
    @Bean
    public JwtAuthenticationFilter jwtAuthenticationFilter() {
        return new JwtAuthenticationFilter();
    }
}

```

```

@Bean
public DaoAuthenticationProvider authenticationProvider() {
    DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
    authProvider.setUserDetailsService(customUserDetailsService);
    authProvider.setPasswordEncoder(passwordEncoder());
    return authProvider;
}
@Bean
public AuthenticationManager authenticationManager(AuthenticationConfiguration auth-
Config) throws Exception {
    return authConfig.getAuthenticationManager();
}
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.cors().and().csrf().disable()
        .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
        .authorizeHttpRequests(authz -> authz
            .requestMatchers("/api/auth/**").permitAll()
            .anyRequest().authenticated()
        );
    http.authenticationProvider(authenticationProvider());
    http.addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFil-
ter.class);
    return http.build();
}
}
package com.quantcrm.system.dashboard;
import java.time.LocalDateTime;
import java.util.List;
public class DashboardData {
    private Integer totalCustomers;
    private Integer activeProjects;
    private Integer pendingReminders;
    private Double averageCustomerValue;
    private List<KeyMetric> keyMetrics;
    private List<RecentActivity> recentActivities;
    private List<PendingTask> pendingTasks;
    public DashboardData() {}
    public DashboardData(Integer totalCustomers, Integer activeProjects, Integer pendingRe-
mindings,
        Double averageCustomerValue, List<KeyMetric> keyMetrics,
        List<RecentActivity> recentActivities, List<PendingTask> pendingTasks) {
        this.totalCustomers = totalCustomers;
        this.activeProjects = activeProjects;
        this.pendingReminders = pendingReminders;
        this.averageCustomerValue = averageCustomerValue;
        this.keyMetrics = keyMetrics;
        this.recentActivities = recentActivities;
        this.pendingTasks = pendingTasks;
    }
    public static class KeyMetric {
        private String name;

```

```

    private String value;
    private String trend;
    public KeyMetric() {}
    public KeyMetric(String name, String value, String trend) {
        this.name = name;
        this.value = value;
        this.trend = trend;
    }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getValue() { return value; }
    public void setValue(String value) { this.value = value; }
    public String getTrend() { return trend; }
    public void setTrend(String trend) { this.trend = trend; }
}

public static class RecentActivity {
    private String type;
    private String description;
    private LocalDateTime timestamp;
    private String customerName;
    public RecentActivity() {}
    public RecentActivity(String type, String description, LocalDateTime timestamp, String customerName) {
        this.type = type;
        this.description = description;
        this.timestamp = timestamp;
        this.customerName = customerName;
    }
    public String getType() { return type; }
    public void setType(String type) { this.type = type; }
    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }
    public LocalDateTime getTimestamp() { return timestamp; }
    public void setTimestamp(LocalDateTime timestamp) { this.timestamp = timestamp; }
    public String getCustomerName() { return customerName; }
    public void setCustomerName(String customerName) { this.customerName = customerName; }
}

public static class PendingTask {
    private String title;
    private String priority;
    private LocalDateTime dueDate;
    private String assignedTo;
    public PendingTask() {}
    public PendingTask(String title, String priority, LocalDateTime dueDate, String assignedTo) {
        this.title = title;
        this.priority = priority;
        this.dueDate = dueDate;
        this.assignedTo = assignedTo;
    }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getPriority() { return priority; }
    public void setPriority(String priority) { this.priority = priority; }
    public LocalDateTime getDueDate() { return dueDate; }

```

```

        public void setDueDate(LocalDateTime dueDate) { this.dueDate = dueDate; }
        public String getAssignedTo() { return assignedTo; }
        public void setAssignedTo(String assignedTo) { this.assignedTo = assignedTo; }
    }
    public Integer getTotalCustomers() { return totalCustomers; }
    public void setTotalCustomers(Integer totalCustomers) { this.totalCustomers = totalCustomers; }
    public Integer getActiveProjects() { return activeProjects; }
    public void setActiveProjects(Integer activeProjects) { this.activeProjects = activeProjects; }
    public Integer getPendingReminders() { return pendingReminders; }
    public void setPendingReminders(Integer pendingReminders) { this.pendingReminders = pendingReminders; }
    public Double getAverageCustomerValue() { return averageCustomerValue; }
    public void setAverageCustomerValue(Double averageCustomerValue) { this.averageCustomerValue = averageCustomerValue; }
    public List<KeyMetric> getKeyMetrics() { return keyMetrics; }
    public void setKeyMetrics(List<KeyMetric> keyMetrics) { this.keyMetrics = keyMetrics; }
    public List<RecentActivity> getRecentActivities() { return recentActivities; }
    public void setRecentActivities(List<RecentActivity> recentActivities) { this.recentActivities = recentActivities; }
    public List<PendingTask> getPendingTasks() { return pendingTasks; }
    public void setPendingTasks(List<PendingTask> pendingTasks) { this.pendingTasks = pendingTasks; }
}

package com.quantcrm.system.dashboard;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.time.LocalDateTime;
import java.util.Arrays;
import java.util.List;
@Service
public class DashboardService {
    @Autowired
    private CustomerDataService customerDataService;
    @Autowired
    private ProjectTrackingService projectTrackingService;
    @Autowired
    private ReminderService reminderService;
    public DashboardData getDashboardData() {
        Integer totalCustomers = customerDataService.getTotalCustomerCount();
        Integer activeProjects = projectTrackingService.getActiveProjectCount();
        Integer pendingReminders = reminderService.getPendingReminderCount();
        Double averageCustomerValue = customerDataService.getAverageCustomerValue();
        List<DashboardData.KeyMetric> keyMetrics = getKeyMetrics();
        List<DashboardData.RecentActivity> recentActivities = getRecentActivities();
        List<DashboardData.PendingTask> pendingTasks = getPendingTasks();
        return new DashboardData(totalCustomers, activeProjects, pendingReminders,
            averageCustomerValue, keyMetrics, recentActivities, pendingTasks);
    }
    private List<DashboardData.KeyMetric> getKeyMetrics() {
        return Arrays.asList(
            new DashboardData.KeyMetric(" 高价值客户", "28", "+5%"),
            new DashboardData.KeyMetric(" 转化率", "42%", "+2.3%"),
            new DashboardData.KeyMetric(" 平均响应时间", "2.1 小时", "-0.3 小时"),
            new DashboardData.KeyMetric(" 客户满意度", "94%", "+1.2%")
        );
    }
}

```

```

    );
}
private List<DashboardData.RecentActivity> getRecentActivities() {
    return Arrays.asList(
        new DashboardData.RecentActivity(" 咨询", " 新客户咨询量化投资策略",
            LocalDateTime.now().minusHours(2), " 张伟"),
        new DashboardData.RecentActivity(" 跟进", " 风险等级评估完成",
            LocalDateTime.now().minusHours(5), " 李芳"),
        new DashboardData.RecentActivity(" 签约", " 投资组合管理服务签约",
            LocalDateTime.now().minusDays(1), " 王刚"),
        new DashboardData.RecentActivity(" 分析", " 客户价值分析报告生成",
            LocalDateTime.now().minusDays(1), " 赵敏")
    );
}
private List<DashboardData.PendingTask> getPendingTasks() {
    return Arrays.asList(
        new DashboardData.PendingTask(" 客户风险评估报告", " 高",
            LocalDateTime.now().plusDays(1), " 张顾问"),
        new DashboardData.PendingTask(" 投资组合调整建议", " 中",
            LocalDateTime.now().plusDays(2), " 李分析师"),
        new DashboardData.PendingTask(" 月度业绩报告", " 中",
            LocalDateTime.now().plusDays(3), " 王经理"),
        new DashboardData.PendingTask(" 客户回访安排", " 低",
            LocalDateTime.now().plusDays(5), " 赵客服")
    );
}
}
package com.quantcrm.system.dashboard;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import org.springframework.http.ResponseEntity;
@RestController
@RequestMapping("/api/dashboard")
public class DashboardController {
    @Autowired
    private DashboardService dashboardService;
    @GetMapping
    public ResponseEntity<DashboardData> getDashboardData() {
        try {
            DashboardData dashboardData = dashboardService.getDashboardData();
            return ResponseEntity.ok(dashboardData);
        } catch (Exception e) {
            return ResponseEntity.internalServerError().build();
        }
    }
}
package com.quantcrm.system.dashboard;
import org.springframework.stereotype.Service;
@Service
public class CustomerDataService {
    public Integer getTotalCustomerCount() {
        return 156;
    }
    public Double getAverageCustomerValue() {
        return 285000.0;
    }
}

```



```
    }  
}  
package com.quantcrm.system.dashboard;  
import org.springframework.stereotype.Service;  
@Service  
public class ProjectTrackingService {  
    public Integer getActiveProjectCount() {  
        return 42;  
    }  
}  
package com.quantcrm.system.service;  
import com.quantcrm.system.model.ReminderRule;  
import com.quantcrm.system.repository.ReminderRuleRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.Pageable;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
import java.util.List;  
import java.util.Optional;  
@Service  
public class ReminderService {  
    @Autowired  
    private ReminderRuleRepository reminderRuleRepository;  
    public Page<ReminderRule> getAllReminderRules(Pageable pageable) {  
        return reminderRuleRepository.findAll(pageable);  
    }  
    public List<ReminderRule> getActiveReminderRules() {  
        return reminderRuleRepository.findByIsActiveTrue();  
    }  
    public Optional<ReminderRule> getReminderRuleById(Long id) {  
        return reminderRuleRepository.findById(id);  
    }  
    public List<ReminderRule> getReminderRulesByType(ReminderRule.ReminderType type) {  
        return reminderRuleRepository.findByType(type);  
    }  
    @Transactional  
    public ReminderRule createReminderRule(ReminderRule reminderRule) {  
        return reminderRuleRepository.save(reminderRule);  
    }  
    @Transactional  
    public ReminderRule updateReminderRule(Long id, ReminderRule reminderRuleDetails) {  
        ReminderRule reminderRule = reminderRuleRepository.findById(id)  
            .orElseThrow(() -> new RuntimeException("Reminder rule not found with id: " + id));  
        reminderRule.setName(reminderRuleDetails.getName());  
        reminderRule.setDescription(reminderRuleDetails.getDescription());  
        reminderRule.setType(reminderRuleDetails.getType());  
        reminderRule.setTriggerCondition(reminderRuleDetails.getTriggerCondition());  
        reminderRule.setReminderMessage(reminderRuleDetails.getReminderMessage());  
        reminderRule.setIsActive(reminderRuleDetails.getIsActive());  
        reminderRule.setRecipientType(reminderRuleDetails.getRecipientType());  
        return reminderRuleRepository.save(reminderRule);  
    }  
    @Transactional  
    public ReminderRule toggleReminderRuleStatus(Long id) {
```

```

        ReminderRule reminderRule = reminderRuleRepository.findById(id)
            .orElseThrow(() -> new RuntimeException("Reminder rule not found with id: " + id));
        reminderRule.setIsActive(!reminderRule.getIsActive());
        return reminderRuleRepository.save(reminderRule);
    }
    @Transactional
    public void deleteReminderRule(Long id) {
        ReminderRule reminderRule = reminderRuleRepository.findById(id)
            .orElseThrow(() -> new RuntimeException("Reminder rule not found with id: " + id));
        reminderRuleRepository.delete(reminderRule);
    }
}
package com.quantcrm.system.model;
import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.List;
@Entity
@Table(name = "customer")
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "name", nullable = false)
    private String name;
    @Column(name = "phone")
    private String phone;
    @Column(name = "email")
    private String email;
    @Column(name = "company")
    private String company;
    @Column(name = "position")
    private String position;
    @Column(name = "industry")
    private String industry;
    @Column(name = "source")
    private String source;
    @Column(name = "status")
    private String status;
    @Column(name = "level")
    private Integer level;
    @ElementCollection
    @CollectionTable(name = "customer_tags", joinColumns = @JoinColumn(name = "customer_id"))
    @Column(name = "tag")
    private List<String> tags;
    @Column(name = "preferences", columnDefinition = "TEXT")
    private String preferences;
    @Column(name = "consultation_history", columnDefinition = "TEXT")
    private String consultationHistory;
    @Column(name = "created_at")
    private LocalDateTime createdAt;
    @Column(name = "updated_at")
    private LocalDateTime updatedAt;
    @Column(name = "created_by")
    private String createdBy;
    @Column(name = "updated_by")

```

```

private String updatedBy;
@PrePersist
protected void onCreate() {
    createdAt = LocalDateTime.now();
    updatedAt = LocalDateTime.now();
}
@PreUpdate
protected void onUpdate() {
    updatedAt = LocalDateTime.now();
}
// Getters and Setters
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }
public String getName() { return name; }
public void setName(String name) { this.name = name; }
public String getPhone() { return phone; }
public void setPhone(String phone) { this.phone = phone; }
public String getEmail() { return email; }
public void setEmail(String email) { this.email = email; }
public String getCompany() { return company; }
public void setCompany(String company) { this.company = company; }
public String getPosition() { return position; }
public void setPosition(String position) { this.position = position; }
public String getIndustry() { return industry; }
public void setIndustry(String industry) { this.industry = industry; }
public String getSource() { return source; }
public void setSource(String source) { this.source = source; }
public String getStatus() { return status; }
public void setStatus(String status) { this.status = status; }
public Integer getLevel() { return level; }
public void setLevel(Integer level) { this.level = level; }
public List<String> getTags() { return tags; }
public void setTags(List<String> tags) { this.tags = tags; }
public String getPreferences() { return preferences; }
public void setPreferences(String preferences) { this.preferences = preferences; }
public String getConsultationHistory() { return consultationHistory; }
public void setConsultationHistory(String consultationHistory) { this.consultationHistory = consultationHistory; }
public LocalDateTime getCreatedAt() { return createdAt; }
public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }
public LocalDateTime getUpdatedAt() { return updatedAt; }
public void setUpdatedAt(LocalDateTime updatedAt) { this.updatedAt = updatedAt; }
public String getCreatedBy() { return createdBy; }
public void setCreatedBy(String createdBy) { this.createdBy = createdBy; }
public String getUpdatedBy() { return updatedBy; }
public void setUpdatedBy(String updatedBy) { this.updatedBy = updatedBy; }
}
package com.quantcrm.system.repository;
import com.quantcrm.system.model.Customer;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

```

```

import org.springframework.stereotype.Repository;
import java.util.List;
@Repository
public interface CustomerRepository extends JpaRepository<Customer, Long>, JpaRepository<Customer> {
    Page<Customer> findByNameContainingIgnoreCase(String name, Pageable pageable);
    Page<Customer> findByPhoneContaining(String phone, Pageable pageable);
    Page<Customer> findByEmailContainingIgnoreCase(String email, Pageable pageable);
    Page<Customer> findByCompanyContainingIgnoreCase(String company, Pageable pageable);
    Page<Customer> findByStatus(String status, Pageable pageable);
    Page<Customer> findByIndustry(String industry, Pageable pageable);
    @Query("SELECT DISTINCT c.industry FROM Customer c WHERE c.industry IS NOT NULL")
    List<String> findAllIndustries();
    @Query("SELECT DISTINCT c.status FROM Customer c WHERE c.status IS NOT NULL")
    List<String> findAllStatuses();
    @Query("SELECT DISTINCT c.source FROM Customer c WHERE c.source IS NOT NULL")
    List<String> findAllSources();
    @Query("SELECT c FROM Customer c WHERE :tag MEMBER OF c.tags")
    Page<Customer> findByTag(@Param("tag") String tag, Pageable pageable);
    boolean existsByEmail(String email);
    boolean existsByPhone(String phone);
}
package com.quantcrm.system.service;
import com.quantcrm.system.model.Customer;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.web.multipart.MultipartFile;
import java.util.List;
import java.util.Map;
public interface CustomerService {
    Page<Customer> getAllCustomers(Pageable pageable);
    Customer getCustomerById(Long id);
    Customer createCustomer(Customer customer);
    Customer updateCustomer(Long id, Customer customer);
    void deleteCustomer(Long id);
    Page<Customer> searchCustomers(String keyword, Pageable pageable);
    Page<Customer> filterCustomers(Map<String, Object> filters, Pageable pageable);
    List<String> getAllIndustries();
    List<String> getAllStatuses();
    List<String> getAllSources();
    List<Customer> importCustomers(MultipartFile file);
    Map<String, Object> validateImportData(MultipartFile file);
    Customer addTagToCustomer(Long customerId, String tag);
    Customer removeTagFromCustomer(Long customerId, String tag);
    List<String> getAllTags();
}
package com.quantcrm.system.service.impl;
import com.quantcrm.system.model.Customer;
import com.quantcrm.system.repository.CustomerRepository;
import com.quantcrm.system.service.CustomerService;
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;
import org.springframework.beans.factory.annotation.Autowired;

```

```

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import javax.persistence.criteria.Predicate;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.nio.charset.StandardCharsets;
import java.time.LocalDateTime;
import java.util.*;
import java.util.stream.Collectors;
@Service
public class CustomerServiceImpl implements CustomerService {
    @Autowired
    private CustomerRepository customerRepository;
    @Override
    public Page<Customer> getAllCustomers(Pageable pageable) {
        return customerRepository.findAll(pageable);
    }
    @Override
    public Customer getCustomerById(Long id) {
        return customerRepository.findById(id)
            .orElseThrow(() -> new RuntimeException("Customer not found with id: " + id));
    }
    @Override
    public Customer createCustomer(Customer customer) {
        if (customer.getEmail() != null && customerRepository.existsByEmail(customer.getEmail())) {
            throw new RuntimeException("Customer with email " + customer.getEmail() + " already exists");
        }
        if (customer.getPhone() != null && customerRepository.existsByPhone(customer.getPhone())) {
            throw new RuntimeException("Customer with phone " + customer.getPhone() + " already exists");
        }
        customer.setCreatedAt(LocalDateTime.now());
        customer.setUpdatedAt(LocalDateTime.now());
        return customerRepository.save(customer);
    }
    @Override
    public Customer updateCustomer(Long id, Customer customerDetails) {
        Customer customer = getCustomerById(id);
        if (customerDetails.getEmail() != null &&
            !customerDetails.getEmail().equals(customer.getEmail()) &&
            customerRepository.existsByEmail(customerDetails.getEmail())) {
            throw new RuntimeException("Customer with email " + customerDetails.getEmail() + " already exists");
        }
        if (customerDetails.getPhone() != null &&
            !customerDetails.getPhone().equals(customer.getPhone()) &&
            customerRepository.existsByPhone(customerDetails.getPhone())) {
            throw new RuntimeException("Customer with phone " + customerDetails.getPhone() + " already exists");
        }
        customer.setName(customerDetails.getName());
    }

```

```

        customer.setPhone(customerDetails.getPhone());
        customer.setEmail(customerDetails.getEmail());
        customer.setCompany(customerDetails.getCompany());
        customer.setPosition(customerDetails.getPosition());
        customer.setIndustry(customerDetails.getIndustry());
        customer.setSource(customerDetails.getSource());
        customer.setStatus(customerDetails.getStatus());
        customer.setLevel(customerDetails.getLevel());
        customer.setTags(customerDetails.getTags());
        customer.setPreferences(customerDetails.getPreferences());
        customer.setConsultationHistory(customerDetails.getConsultationHistory());
        customer.setUpdatedAt(LocalDateTime.now());
        customer.setUpdatedBy(customerDetails.getUpdatedBy());
        return customerRepository.save(customer);
    }

    @Override
    public void deleteCustomer(Long id) {
        Customer customer = getCustomerById(id);
        customerRepository.delete(customer);
    }

    @Override
    public Page<Customer> searchCustomers(String keyword, Pageable pageable) {
        if (keyword == null || keyword.trim().isEmpty()) {
            return customerRepository.findAll(pageable);
        }
        String searchTerm = "%" + keyword.toLowerCase() + "%";
        Specification<Customer> spec = (root, query, criteriaBuilder) -> {
            List<Predicate> predicates = new ArrayList<>();
            predicates.add(criteriaBuilder.like(criteriaBuilder.lower(root.get("name")), searchTerm));
            predicates.add(criteriaBuilder.like(root.get("phone"), searchTerm));
            predicates.add(criteriaBuilder.like(criteriaBuilder.lower(root.get("email")), searchTerm));
            predicates.add(criteriaBuilder.like(criteriaBuilder.lower(root.get("company")), searchTerm));
            return criteriaBuilder.or(predicates.toArray(new Predicate[0]));
        };
        return customerRepository.findAll(spec, pageable);
    }

    @Override
    public Page<Customer> filterCustomers(Map<String, Object> filters, Pageable pageable) {
        Specification<Customer> spec = (root, query, criteriaBuilder) -> {
            List<Predicate> predicates = new ArrayList<>();
            if (filters.containsKey("industry") && filters.get("industry") != null) {
                predicates.add(criteriaBuilder.equal(root.get("industry"), filters.get("industry")));
            }
            if (filters.containsKey("status") && filters.get("status") != null) {
                predicates.add(criteriaBuilder.equal(root.get("status"), filters.get("status")));
            }
            if (filters.containsKey("source") && filters.get("source") != null) {
                predicates.add(criteriaBuilder.equal(root.get("source"), filters.get("source")));
            }
            if (filters.containsKey("level") && filters.get("level") != null) {
                predicates.add(criteriaBuilder.equal(root.get("level"), filters.get("level")));
            }
            if (filters.containsKey("tags") && filters.get("tags") != null) {
                String tag = (String) filters.get("tags");
                predicates.add(criteriaBuilder.isMember(tag, root.get("tags")));
            }
        };
    }

```

```

    }
    return criteriaBuilder.and(predicates.toArray(new Predicate[0]));
};
return customerRepository.findAll(spec, pageable);
}
@Override
public List<String> getAllIndustries() {
    return customerRepository.findAllIndustries();
}
@Override
public List<String> getAllStatuses() {
    return customerRepository.findAllStatuses();
}
@Override
public List<String> getAllSources() {
    return customerRepository.findAllSources();
}
@Override
public List<Customer> importCustomers(MultipartFile file) {
    List<Customer> importedCustomers = new ArrayList<>();
    try (BufferedReader fileReader = new BufferedReader(
        new InputStreamReader(file.getInputStream(), StandardCharsets.UTF_8));
        CSVParser csvParser = new CSVParser(fileReader,
        CSVFormat.DEFAULT.withFirstRecordAsHeader().withIgnoreHeaderCase().withTrim())) {
        Iterable<CSVRecord> csvRecords = csvParser.getRecords();
        for (CSVRecord csvRecord : csvRecords) {
            Customer customer = new Customer();
            customer.setName(csvRecord.get("name"));
            customer.setPhone(csvRecord.get("phone"));
            customer.setEmail(csvRecord.get("email"));
            customer.setCompany(csvRecord.get("company"));
            customer.setPosition(csvRecord.get("position"));
            customer.setIndustry(csvRecord.get("industry"));
            customer.setSource(csvRecord.get("source"));
            customer.setStatus(csvRecord.get("status"));
            if (csvRecord.isSet("level")) {
                try {
                    customer.setLevel(Integer.parseInt(csvRecord.get("level")));
                } catch (NumberFormatException e) {
                    customer.setLevel(1);
                }
            }
            if (csvRecord.isSet("tags")) {
                String tags = csvRecord.get("tags");
                if (tags != null && !tags.isEmpty()) {
                    List<String> tagList = Arrays.stream(tags.split(","))
                        .map(String::trim)
                        .collect(Collectors.toList());
                    customer.setTags(tagList);
                }
            }
            customer.setCreatedAt(LocalDateTime.now());
            customer.setUpdatedAt(LocalDateTime.now());
            importedCustomers.add(customerRepository.save(customer));
        }
    }
}

```

```

    } catch (Exception e) {
        throw new RuntimeException("Failed to import customers: " + e.getMessage());
    }
    return importedCustomers;
}
@Override
public Map<String, Object> validateImportData(MultipartFile file) {
    Map<String, Object> validationResult = new HashMap<>();
    List<String> errors = new ArrayList<>();
    List<String> warnings = new ArrayList<>();
    if (file.isEmpty()) {
        errors.add("File is empty");
        validationResult.put("errors", errors);
        validationResult.put("warnings", warnings);
        return validationResult;
    }
    if (!Objects.requireNonNull(file.getOriginalFilename()).toLowerCase().endsWith(".csv")) {
        errors.add("Only CSV files are supported");
    }
    try (BufferedReader fileReader = new BufferedReader(
        new InputStreamReader(file.getInputStream(), StandardCharsets.UTF_8));
        CSVParser csvParser = new CSVParser(fileReader,
        CSVFormat.DEFAULT.withFirstRecordAsHeader().withIgnoreHeaderCase().withTrim())) {
        Set<String> headers = csvParser.getHeaderMap().keySet();
        Set<String> requiredHeaders = Set.of("name", "phone", "email");
        for (String requiredHeader : requiredHeaders) {
            if (!headers.contains(requiredHeader)) {
                errors.add("Missing required column: " + requiredHeader);
            }
        }
    }
    int recordCount = 0;
    for (CSVRecord record : csvParser) {
        recordCount++;
        if (record.get("name") == null || record.get("name").trim().isEmpty()) {
            errors.add("Record " + recordCount + ": Name is required");
        }
        if (record.get("email") != null && !record.get("email").trim().isEmpty()) {
            String email = record.get("email").trim();
            if (customerRepository.existsByEmail(email)) {
                warnings.add("Record " + recordCount + ": Email " + email + " already exists");
            }
        }
    }
    validationResult.put("totalRecords", recordCount);
    0.5: '0.5',
    1: '1'
    }}
    />
</Form.Item>
</Col>
</Row>
<Row gutter={16}>
<Col span={12}>
<Form.Item
    label="潜在需求评分"

```



```

        name="potentialDemandScore"
        tooltip=" 潜在需求分析的基准评分"
    >
        <InputNumber
            min={0}
            max={1}
            step={0.1}
            style={{ width: '100%' }}
        />
    </Form.Item>
</Col>
<Col span={12}>
    <Form.Item
        label=" 分析刷新闻隔 (秒)"
        name="analysisRefreshInterval"
        tooltip=" 量化分析数据自动刷新的时间间隔"
    >
        <InputNumber
            min={300}
            max={86400}
            step={300}
            style={{ width: '100%' }}
        />
    </Form.Item>
</Col>
</Row>
<Row gutter={16}>
    <Col span={12}>
        <Form.Item
            label=" 自动生成报告"
            name="autoGenerateReports"
            valuePropName="checked"
        >
            <Switch />
        </Form.Item>
    </Col>
    <Col span={12}>
        <Form.Item
            label=" 报告保留天数"
            name="reportRetentionDays"
            tooltip=" 自动生成的报告在系统中的保留天数"
        >
            <InputNumber
                min={30}
                max={1095}
                step={30}
                style={{ width: '100%' }}
            />
        </Form.Item>
    </Col>
</Row>
</Form>
</Card>
);
};

```

```

import React from 'react';
import { Card, Form, Switch, Input, Select, Button, Row, Col } from 'antd';
import { SaveOutlined } from '@ant-design/icons';
import type { NotificationSettings } from './settings_types';
const { TextArea } = Input;
const { Option } = Select;
interface NotificationSettingsProps {
  settings: NotificationSettings;
  loading: boolean;
  onSave: (data: NotificationSettings) => void;
}
export const NotificationSettingsForm: React.FC<NotificationSettingsProps> = ({
  settings,
  loading,
  onSave
}) => {
  const [form] = Form.useForm();
  const handleSave = async () => {
    try {
      const values = await form.validateFields();
      onSave(values);
    } catch (error) {
      console.error(' 表单验证失败:', error);
    }
  };
  return (
    <Card
      title=" 通知设置"
      className="rounded-lg shadow-md card-bg-color"
      extra={
        <Button
          type="primary"
          icon={<SaveOutlined />}
          loading={loading}
          onClick={handleSave}
          className="primary-color"
        >
          保存设置
        </Button>
      }
    >
      <Form
        form={form}
        layout="vertical"
        initialValues={settings}
        className="text-primary"
      >
        <Row gutter={16}>
          <Col span={8}>
            <Form.Item
              label=" 邮件通知"
              name="emailNotifications"
              valuePropName="checked"
            >
              <Switch />

```

```

    </Form.Item>
  </Col>
  <Col span={8}>
    <Form.Item
      label=" 短信通知"
      name="smsNotifications"
      valuePropName="checked"
    >
      <Switch />
    </Form.Item>
  </Col>
  <Col span={8}>
    <Form.Item
      label=" 推送通知"
      name="pushNotifications"
      valuePropName="checked"
    >
      <Switch />
    </Form.Item>
  </Col>
</Row>
<Form.Item
  label=" 邮件模板"
  name="emailTemplate"
  dependencies={['emailNotifications']}
>
  {( { getFieldValue } ) =>
    getFieldValue('emailNotifications') ? (
      <TextArea
        rows={4}
        placeholder=" 请输入邮件通知模板内容"
      />
    ) : null
  }
</Form.Item>
<Form.Item
  label=" 短信模板"
  name="smsTemplate"
  dependencies={['smsNotifications']}
>
  {( { getFieldValue } ) =>
    getFieldValue('smsNotifications') ? (
      <TextArea
        rows={2}
        placeholder=" 请输入短信通知模板内容"
        maxLength={70}
        showCount
      />
    ) : null
  }
</Form.Item>
<Form.Item
  label=" 提醒间隔 (天)"
  name="reminderIntervals"
  tooltip=" 设置提醒通知的发送间隔天数"

```

```

    >
    <Select mode="multiple" placeholder=" 选择提醒间隔">
      <Option value={1}>1 天 </Option>
      <Option value={3}>3 天 </Option>
      <Option value={7}>7 天 </Option>
      <Option value={15}>15 天 </Option>
      <Option value={30}>30 天 </Option>
    </Select>
  </Form.Item>
  <Form.Item
    label=" 启用关键警报"
    name="criticalAlerts"
    valuePropName="checked"
    tooltip=" 启用高风险客户和异常数据的实时警报"
  >
    <Switch />
  </Form.Item>
</Form>
</Card>
);
};
import React from 'react';
import { Tabs, Spin } from 'antd';
import { SettingOutlined, BarChartOutlined, NotificationOutlined } from '@ant-design/icons';
import { useSettings } from './settings_hooks';
import { PlatformSettingsForm } from './platform_settings';
import { BusinessSettingsForm } from './business_settings';
import { NotificationSettingsForm } from './notification_settings';
const { TabPane } = Tabs;
export const SettingsPage: React.FC = () => {
  const {
    settings,
    savePlatformSettings,
    saveBusinessSettings,
    saveNotificationSettings
  } = useSettings();
  if (settings.loading) {
    return (
      <div className="flex justify-center items-center h-64">
        <Spin size="large" />
      </div>
    );
  }
  return (
    <div className="p-6 page-bg-color min-h-screen">
      <div className="mb-6">
        <h1 className="text-2xl font-bold text-primary"> 系统设置 </h1>
        <p className="text-secondary"> 配置平台基础信息与业务参数 </p>
      </div>
      <Tabs
        defaultActiveKey="platform"
        type="card"
        className="settings-tabs"
      >
        <TabPane

```

```

        tab={
          <span>
            <SettingOutlined />
            平台信息设置
          </span>
        }
        key="platform"
      >
        <PlatformSettingsForm
          settings={settings.platform}
          loading={settings.saving}
          onSave={savePlatformSettings}
        />
      </TabPane>
      <TabPane
        tab={
          <span>
            <BarChartOutlined />
            业务参数配置
          </span>
        }
        key="business"
      >
        <BusinessSettingsForm
          settings={settings.business}
          loading={settings.saving}
          onSave={saveBusinessSettings}
        />
      </TabPane>
      <TabPane
        tab={
          <span>
            <NotificationOutlined />
            通知设置
          </span>
        }
        key="notification"
      >
        <NotificationSettingsForm
          settings={settings.notification}
          loading={settings.saving}
          onSave={saveNotificationSettings}
        />
      </TabPane>
    </Tabs>
  </div>
);
};
import type { PlatformSettings, BusinessParameters, NotificationSettings } from './settings_types';
export const validatePlatformSettings = (settings: PlatformSettings): string[] => {
  const errors: string[] = [];
  if (!settings.systemName || settings.systemName.trim().length === 0) {
    errors.push('系统名称不能为空');
  }
  if (settings.systemName && settings.systemName.length > 50) {

```

```

    errors.push('系统名称长度不能超过 50 个字符');
  }
  if (settings.watermark && (!settings.watermarkText || settings.watermarkText.trim().length === 0)) {
    errors.push('启用水印时, 水印文本不能为空');
  }
  return errors;
};
export const validateBusinessParameters = (params: BusinessParameters): string[] => {
  const errors: string[] = [];
  if (params.riskModelThreshold < 0 || params.riskModelThreshold > 1) {
    errors.push('风险模型阈值必须在 0-1 之间');
  }
  if (params.customerValueWeight < 0 || params.customerValueWeight > 1) {
    errors.push('客户价值权重必须在 0-1 之间');
  }
  if (params.potentialDemandScore < 0 || params.potentialDemandScore > 1) {
    errors.push('潜在需求评分必须在 0-1 之间');
  }
  if (params.analysisRefreshInterval < 300) {
    errors.push('分析刷新闻隔不能小于 300 秒');
  }
  if (params.reportRetentionDays < 1) {
    errors.push('报告保留天数不能小于 1 天');
  }
  return errors;
};
export const validateNotificationSettings = (settings: NotificationSettings): string[] => {
  const errors: string[] = [];
  if (settings.emailNotifications && (!settings.emailTemplate || settings.emailTemplate.trim().length === 0)) {
    errors.push('启用邮件通知时, 邮件模板不能为空');
  }
  if (settings.smsNotifications && (!settings.smsTemplate || settings.smsTemplate.trim().length === 0)) {
    errors.push('启用短信通知时, 短信模板不能为空');
  }
  if (settings.smsNotifications && settings.smsTemplate && settings.smsTemplate.length > 70) {
    errors.push('短信模板长度不能超过 70 个字符');
  }
  if (settings.reminderIntervals.length === 0) {
    errors.push('至少需要设置一个提醒间隔');
  }
  return errors;
};
export const exportSettingsToJson = (settings: {
  platform: PlatformSettings;
  business: BusinessParameters;
  notification: NotificationSettings;
}): string => {
  return JSON.stringify(settings, null, 2);
};
export const importSettingsFromJson = (jsonString: string): {
  platform: PlatformSettings;
  business: BusinessParameters;
  notification: NotificationSettings;
} | null => {
  try {

```

```

    const settings = JSON.parse(jsonString);
    return settings;
  } catch (error) {
    console.error(' 导入设置失败:', error);
    return null;
  }
};export interface UserProfile {
  id: string;
  username: string;
  email: string;
  phone: string;
  department: string;
  position: string;
  avatar?: string;
  createdAt: string;
  lastLoginAt: string;
}
export interface ProfileFormData {
  username: string;
  email: string;
  phone: string;
  department: string;
  position: string;
}
export interface PasswordFormData {
  currentPassword: string;
  newPassword: string;
  confirmPassword: string;
}
export interface PreferenceSettings {
  theme: 'light' | 'dark' | 'auto';
  language: string;
  dashboardLayout: 'compact' | 'comfortable';
  notificationEnabled: boolean;
  emailNotifications: boolean;
  autoSave: boolean;
}
export interface ProfileState {
  userProfile: UserProfile | null;
  preferences: PreferenceSettings;
  isLoading: boolean;
  isUpdating: boolean;
}
import { useState, useEffect, useCallback } from 'react';
import { message } from 'antd';
import { UserProfile, ProfileFormData, PasswordFormData, PreferenceSettings, ProfileState } from './pro-
file_types';
export const useProfile = () => {
  const [state, setState] = useState<ProfileState>({
    userProfile: null,
    preferences: {
      theme: 'light',
      language: 'zh-CN',
      dashboardLayout: 'comfortable',
      notificationEnabled: true,

```

```
    emailNotifications: true,
    autoSave: true
  },
  isLoading: false,
  isUpdating: false
});
const fetchUserProfile = useCallback(async () => {
  setState(prev => ({ ...prev, isLoading: true }));
  try {
    // 模拟 API 调用
    const mockProfile: UserProfile = {
      id: 'user-001',
      username: 'zhangsan',
      email: 'zhangsan@quant-consult.com',
      phone: '13800138000',
      department: ' 量化咨询部',
      position: ' 高级咨询顾问',
      avatar: undefined,
      createdAt: '2023-01-15T08:00:00Z',
      lastLoginAt: new Date().toISOString()
    };
    setState(prev => ({ ...prev, userProfile: mockProfile, isLoading: false }));
  } catch (error) {
    message.error(' 获取用户信息失败');
    setState(prev => ({ ...prev, isLoading: false }));
  }
}, []);
const updateProfile = useCallback(async (formData: ProfileFormData) => {
  setState(prev => ({ ...prev, isUpdating: true }));
  try {
    // 模拟 API 调用
    await new Promise(resolve => setTimeout(resolve, 1000));
    setState(prev => ({
      ...prev,
      userProfile: prev.userProfile ? { ...prev.userProfile, ...formData } : null,
      isUpdating: false
    }));
    message.success(' 个人信息更新成功');
    return true;
  } catch (error) {
    message.error(' 更新个人信息失败');
    setState(prev => ({ ...prev, isUpdating: false }));
    return false;
  }
}, []);
const updatePassword = useCallback(async (formData: PasswordFormData) => {
  setState(prev => ({ ...prev, isUpdating: true }));
  try {
    // 模拟 API 调用
    await new Promise(resolve => setTimeout(resolve, 1000));
    setState(prev => ({ ...prev, isUpdating: false }));
    message.success(' 密码修改成功');
    return true;
  } catch (error) {
    message.error(' 密码修改失败');
```



```

        setState(prev => ({ ...prev, isUpdating: false }));
        return false;
    }
}, []);
const updatePreferences = useCallback(async (preferences: PreferenceSettings) => {
    setState(prev => ({ ...prev, isUpdating: true }));
    try {
        // 模拟 API 调用
        await new Promise(resolve => setTimeout(resolve, 800));
        setState(prev => ({ ...prev, preferences, isUpdating: false }));
        message.success(' 偏好设置已更新');
        return true;
    } catch (error) {
        message.error(' 更新偏好设置失败');
        setState(prev => ({ ...prev, isUpdating: false }));
        return false;
    }
}, []);
useEffect(() => {
    fetchUserProfile();
}, [fetchUserProfile]);
return {
    ...state,
    updateProfile,
    updatePassword,
    updatePreferences
};
};
import React from 'react';
import { Card, Form, Input, Button, Switch, Select, Avatar, Space, Divider, Typography } from 'antd';
import { UserOutlined, MailOutlined, PhoneOutlined, TeamOutlined, SafetyCertificateOutlined, SettingOutlined } from '@ant-design/icons';
import { ProfileFormData, PasswordFormData, PreferenceSettings } from './profile_types';
const { Title, Text } = Typography;
const { Option } = Select;
interface ProfileInfoFormProps {
    initialData: ProfileFormData;
    onSubmit: (data: ProfileFormData) => Promise<boolean>;
    loading: boolean;
}
export const ProfileInfoForm: React.FC<ProfileInfoFormProps> = ({
    initialData,
    onSubmit,
    loading
}) => {
    const [form] = Form.useForm();
    const handleSubmit = async (values: ProfileFormData) => {
        const success = await onSubmit(values);
        if (success) {
            form.setFieldsValue(values);
        }
    };
    return (
        <Card
            className="card-bg-color rounded-lg shadow-md"

```

```

title={
  <Space>
    <UserOutlined className="primary-color" />
    <span className="text-primary"> 个人信息 </span>
  </Space>
}
>
<Form
  form={form}
  layout="vertical"
  initialValues={initialData}
  onFinish={handleSubmit}
  size="large"
>
  <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
    <Form.Item
      label="用户名"
      name="username"
      rules={[{ required: true, message: '请输入用户名' }]}
    >
      <Input
        prefix={<UserOutlined className="text-secondary" />}
        placeholder="请输入用户名"
      />
    </Form.Item>
    <Form.Item
      label="邮箱"
      name="email"
      rules={[
        { required: true, message: '请输入邮箱' },
        { type: 'email', message: '请输入有效的邮箱地址' }
      ]}
    >
      <Input
        prefix={<MailOutlined className="text-secondary" />}
        placeholder="请输入邮箱"
      />
    </Form.Item>
    <Form.Item
      label="手机号"
      name="phone"
      rules={[
        { required: true, message: '请输入手机号' },
        { pattern: /^1[3-9]\d{9}$/, message: '请输入有效的手机号' }
      ]}
    >
      <Input
        prefix={<PhoneOutlined className="text-secondary" />}
        placeholder="请输入手机号"
      />
    </Form.Item>
    <Form.Item
      label="部门"
      name="department"
      rules={[{ required: true, message: '请输入部门' }]}
    >

```

```

    >
    <Input
      prefix={<TeamOutlined className="text-secondary" />}
      placeholder=" 请输入部门"
    />
  </Form.Item>
  <Form.Item
    label=" 职位"
    name="position"
    rules={[{ required: true, message: ' 请输入职位' }]}
  >
    <Input
      placeholder=" 请输入职位"
    />
  </Form.Item>
</div>
<Form.Item className="mt-6">
  <Button
    type="primary"
    htmlType="submit"
    loading={loading}
    className="primary-color"
  >
    保存修改
  </Button>
</Form.Item>
</Form>
</Card>
);
};
interface PasswordFormProps {
  onSubmit: (data: PasswordFormData) => Promise<boolean>;
  loading: boolean;
}
export const PasswordForm: React.FC<PasswordFormProps> = ({
  onSubmit,
  loading
}) => {
  const [form] = Form.useForm();
  const handleSubmit = async (values: PasswordFormData) => {
    const success = await onSubmit(values);
    if (success) {
      form.resetFields();
    }
  };
  return (
    <Card
      className="card-bg-color rounded-lg shadow-md"
      title={
        <Space>
        <SafetyCertificateOutlined className="primary-color" />
        <span className="text-primary"> 修改密码 </span>
        </Space>
      }
    >

```

```

<Form
  form={form}
  layout="vertical"
  onFinish={handleSubmit}
  size="large"
>
  <Form.Item
    label=" 当前密码"
    name="currentPassword"
    rules={[{ required: true, message: ' 请输入当前密码' }]}
  >
    <Input.Password
      placeholder=" 请输入当前密码"
    />
  </Form.Item>
  <Form.Item
    label=" 新密码"
    name="newPassword"
    rules={[
      { required: true, message: ' 请输入新密码' },
      { min: 6, message: ' 密码长度至少 6 位' }
    ]}
  >
    <Input.Password
      placeholder=" 请输入新密码"
    />
  </Form.Item>
  <Form.Item
    label=" 确认新密码"
    name="confirmPassword"
    dependencies={['newPassword']}
    rules={[
      { required: true, message: ' 请确认新密码' },
      ({ getFieldValue }) => ({
        validator(_, value) {
          if (!value || getFieldValue('newPassword') === value) {
            return Promise.resolve();
          }
          return Promise.reject(new Error(' 两次输入的密码不一致'));
        },
      }),
    ]}
  >
    <Input.Password
      placeholder=" 请确认新密码"
    />
  </Form.Item>
  <Form.Item className="mt-6">
    <Button
      type="primary"
      htmlType="submit"
      loading={loading}
      className="primary-color"
    >
      修改密码

```

```

        </Button>
      </Form.Item>
    </Form>
  </Card>
);
};
interface PreferencesFormProps {
  initialData: PreferenceSettings;
  onSubmit: (data: PreferenceSettings) => Promise<boolean>;
  loading: boolean;
}
export const PreferencesForm: React.FC<PreferencesFormProps> = ({
  initialData,
  onSubmit,
  loading
}) => {
  const [form] = Form.useForm();
  const handleSubmit = async (values: PreferenceSettings) => {
    const success = await onSubmit(values);
    if (success) {
      form.setFieldsValue(values);
    }
  };
  return (
    <Card
      className="card-bg-color rounded-lg shadow-md"
      title={
        <Space>
        <SettingOutlined className="primary-color" />
        <span className="text-primary"> 偏好设置 </span>
        </Space>
      }
    >
    <Form
      form={form}
      layout="vertical"
      initialValues={initialData}
      onFinish={handleSubmit}
      size="large"
    >
    <Title level={5}> 界面设置 </Title>
    <div className="grid grid-cols-1 md:grid-cols-2 gap-6 mb-6">
      <Form.Item
        label=" 主题模式"
        name="theme"
      >
        <Select>
          <Option value="light"> 浅色模式 </Option>
          <Option value="dark"> 深色模式 </Option>
          <Option value="auto"> 跟随系统 </Option>
        </Select>
      </Form.Item>
      <Form.Item
        label=" 语言"
        name="language"

```

```

    >
    <Select>
      <Option value="zh-CN"> 简体中文 </Option>
      <Option value="en-US">English</Option>
    </Select>
  </Form.Item>
  <Form.Item
    label=" 工作台布局"
    name="dashboardLayout"
  >
    <Select>
      <Option value="comfortable"> 舒适布局 </Option>
      <Option value="compact"> 紧凑布局 </Option>
    </Select>
  </Form.Item>
</div>
<Divider />
<Title level={5}> 通知设置 </Title>
<div className="space-y-4 mb-6">
  <Form.Item
    label=" 启用系统通知"
    name="notificationEnabled"
    valuePropName="checked"
  >
    <Switch />
  </Form.Item>
  <Form.Item
    label=" 启用邮件通知"
    name="emailNotifications"
    valuePropName="checked"
  >
    <Switch />
  </Form.Item>
</div>
<Divider />
<Title level={5}> 其他设置 </Title>
<div className="mb-6">
  <Form.Item
    label=" 自动保存"
    name="autoSave"
    valuePropName="checked"
  >
    <Switch />
  </Form.Item>
</div>
<Form.Item>
  <Button
    type="primary"
    htmlType="submit"
    loading={loading}
    className="primary-color"
  >
    保存设置
  </Button>
</Form.Item>

```

```

    </Form>
  </Card>
);
};
interface ProfileHeaderProps {
  userProfile: {
    username: string;
    email: string;
    department: string;
    position: string;
    lastLoginAt: string;
  };
}
export const ProfileHeader: React.FC<ProfileHeaderProps> = ({ userProfile }) => {
  return (
    <Card className="card-bg-color rounded-lg shadow-md mb-6">
      <div className="flex items-center space-x-4">
        <Avatar size={64} icon={<UserOutlined />} className="bg-primary-color" />
        <div className="flex-1">
          <div className="flex items-center space-x-2 mb-2">
            <Title level={4} className="text-primary mb-0">{userProfile.username}</Title>
            <Text className="text-secondary">{userProfile.position}</Text>
          </div>
          <div className="grid grid-cols-1 md:grid-cols-3 gap-4 text-sm">
            <div>
              <Text type="secondary"> 邮箱: </Text>
              <Text>{userProfile.email}</Text>
            </div>
            <div>
              <Text type="secondary"> 部门: </Text>
              <Text>{userProfile.department}</Text>
            </div>
            <div>
              <Text type="secondary"> 最后登录: </Text>
              <Text>{new Date(userProfile.lastLoginAt).toLocaleString()}</Text>
            </div>
          </div>
        </div>
      </div>
    </Card>
  );
};
import React from 'react';
import { Tabs, Spin } from 'antd';
import { UserOutlined, SafetyCertificateOutlined, SettingOutlined } from '@ant-design/icons';
import { useProfile } from './profile_hooks';
import { ProfileHeader, ProfileInfoForm, PasswordForm, PreferencesForm } from './profile_components';
const { TabPane } = Tabs;
export const ProfilePage: React.FC = () => {
  const {
    userProfile,
    preferences,
    isLoading,
    isUpdating,
    updateProfile,
  }

```

```

    updatePassword,
    updatePreferences
  } = useProfile();
  if (isLoading || !userProfile) {
    return (
      <div className="flex justify-center items-center min-h-64">
        <Spin size="large" />
      </div>
    );
  }
  const profileFormData = {
    username: userProfile.username,
    email: userProfile.email,
    phone: userProfile.phone,
    department: userProfile.department,
    position: userProfile.position
  };
  return (
    <div className="page-bg-color min-h-screen p-6">
      <div className="max-w-4xl mx-auto">
        <ProfileHeader userProfile={userProfile} />
        <Tabs
          defaultActiveKey="profile"
          size="large"
          className="bg-card-bg-color rounded-lg shadow-md"
          items={[
            {
              key: 'profile',
              label: (
                <span>
                  <UserOutlined />
                  个人信息
                </span>
              ),
              children: (
                <ProfileInfoForm
                  initialData={profileFormData}
                  onSubmit={updateProfile}
                  loading={isUpdating}
                />
              )
            },
            {
              key: 'password',
              label: (
                <span>
                  <SafetyCertificateOutlined />
                  修改密码
                </span>
              ),
              children: (
                <PasswordForm
                  onSubmit={updatePassword}
                  loading={isUpdating}
                />
              )
            }
          ]}
        />
      </div>
    </div>
  );

```



```

    )
  },
  {
    key: 'preferences',
    label: (
      <span>
        <SettingOutlined />
        偏好设置
      </span>
    ),
    children: (
      <PreferencesForm
        initialData={preferences}
        onSubmit={updatePreferences}
        loading={isUpdating}
      />
    )
  }
]
/>
</div>
</div>
);
};
import { PasswordFormData } from './profile_types';
export const validatePasswordStrength = (password: string): boolean => {
  const minLength = 6;
  const hasUpperCase = /[A-Z]/.test(password);
  const hasLowerCase = /[a-z]/.test(password);
  const hasNumbers = /\d/.test(password);
  const hasSpecialChar = /[!@#$$%^&*()_,?":{}|<>]/.test(password);
  return password.length >= minLength &&
    (hasUpperCase || hasLowerCase) &&
    hasNumbers;
};
export const formatPhoneNumber = (phone: string): string => {
  const cleaned = phone.replace(/\D/g, '');
  if (cleaned.length === 11) {
    return cleaned.replace(/(\d{3})(\d{4})(\d{4})/, '$1 **** $3');
  }
  return phone;
};
export const validateProfileData = (data: any): string[] => {
  const errors: string[] = [];
  if (!data.username || data.username.trim().length < 2) {
    errors.push('用户名至少需要 2 个字符');
  }
  if (!data.email || !/^[^\s@]+@[^\s@]+\.[^\s@]+$/ .test(data.email)) {
    errors.push('请输入有效的邮箱地址');
  }
  if (!data.phone || !/^1[3-9]\d{9}$/ .test(data.phone)) {
    errors.push('请输入有效的手机号码');
  }
  if (!data.department || data.department.trim().length === 0) {
    errors.push('部门信息不能为空');
  }

```

```

    }
    if (!data.position || data.position.trim().length === 0) {
      errors.push(' 职位信息不能为空');
    }
    return errors;
  };
export const validatePasswordData = (data: PasswordFormData): string[] => {
  const errors: string[] = [];
  if (!data.currentPassword) {
    errors.push(' 请输入当前密码');
  }
  if (!data.newPassword) {
    errors.push(' 请输入新密码');
  } else if (!validatePasswordStrength(data.newPassword)) {
    errors.push(' 新密码强度不足, 请包含字母和数字, 长度至少 6 位');
  }
  if (data.newPassword !== data.confirmPassword) {
    errors.push(' 两次输入的密码不一致');
  }
  return errors;
};
export interface SecurityLogRecord {
  id: string;
  userId: string;
  userName: string;
  actionType: 'LOGIN' | 'LOGOUT' | 'DATA_ACCESS' | 'DATA_MODIFY' | 'SYSTEM_OPERATION';
  actionDescription: string;
  ipAddress: string;
  userAgent: string;
  deviceInfo: string;
  timestamp: Date;
  status: 'SUCCESS' | 'FAILED';
  targetResource?: string;
  details?: Record<string, any>;
}
export interface SecurityLogQueryParams {
  page: number;
  pageSize: number;
  startTime?: Date;
  endTime?: Date;
  actionType?: string;
  status?: string;
  userName?: string;
  ipAddress?: string;
}
export interface SecurityLogResponse {
  records: SecurityLogRecord[];
  total: number;
  currentPage: number;
  pageSize: number;
}
export const ACTION_TYPE_OPTIONS = [
  { label: ' 登录', value: 'LOGIN' },
  { label: ' 登出', value: 'LOGOUT' },
  { label: ' 数据访问', value: 'DATA_ACCESS' },
  { label: ' 数据修改', value: 'DATA_MODIFY' },

```

```

    { label: '系统操作', value: 'SYSTEM_OPERATION' }
  ];
  export const STATUS_OPTIONS = [
    { label: '成功', value: 'SUCCESS' },
    { label: '失败', value: 'FAILED' }
  ];
  import { SecurityLogRecord, SecurityLogQueryParams, SecurityLogResponse } from './security_log_types';
  class SecurityLogService {
    private baseUrl = '/api/security-logs';
    async getOperationLogs(params: SecurityLogQueryParams): Promise<SecurityLogResponse> {
      const queryString = new URLSearchParams({
        page: params.page.toString(),
        pageSize: params.pageSize.toString(),
        ...(params.startTime && { startTime: params.startTime.toISOString() }),
        ...(params.endTime && { endTime: params.endTime.toISOString() }),
        ...(params.actionType && { actionType: params.actionType }),
        ...(params.status && { status: params.status }),
        ...(params.userName && { userName: params.userName }),
        ...(params.ipAddress && { ipAddress: params.ipAddress })
      }).toString();
      const response = await fetch(`${this.baseUrl}/operation?${queryString}`);
      if (!response.ok) {
        throw new Error('获取操作日志失败');
      }
      return response.json();
    }
    async getLoginLogs(params: SecurityLogQueryParams): Promise<SecurityLogResponse> {
      const queryString = new URLSearchParams({
        page: params.page.toString(),
        pageSize: params.pageSize.toString(),
        ...(params.startTime && { startTime: params.startTime.toISOString() }),
        ...(params.endTime && { endTime: params.endTime.toISOString() }),
        ...(params.userName && { userName: params.userName }),
        ...(params.ipAddress && { ipAddress: params.ipAddress }),
        ...(params.status && { status: params.status })
      }).toString();
      const response = await fetch(`${this.baseUrl}/login?${queryString}`);
      if (!response.ok) {
        throw new Error('获取登录日志失败');
      }
      return response.json();
    }
    async getDataAccessLogs(params: SecurityLogQueryParams): Promise<SecurityLogResponse> {
      const queryString = new URLSearchParams({
        page: params.page.toString(),
        pageSize: params.pageSize.toString(),
        ...(params.startTime && { startTime: params.startTime.toISOString() }),
        ...(params.endTime && { endTime: params.endTime.toISOString() }),
        ...(params.userName && { userName: params.userName }),
        ...(params.ipAddress && { ipAddress: params.ipAddress })
      }).toString();
      const response = await fetch(`${this.baseUrl}/data-access?${queryString}`);
      if (!response.ok) {
        throw new Error('获取数据访问日志失败');
      }
    }
  }

```

```

    }
    return response.json();
  }
  async exportLogs(params: SecurityLogQueryParams, logType: string): Promise<Blob> {
    const queryString = new URLSearchParams({
      ...params,
      startTime: params.startTime?.toISOString(),
      endTime: params.endTime?.toISOString()
    } as any).toString();
    const response = await fetch(`${this.baseUrl}/export/${logType}?${queryString}`);
    if (!response.ok) {
      throw new Error(' 导出日志失败');
    }
    return response.blob();
  }
}

export const securityLogService = new SecurityLogService();
import { useState, useCallback } from 'react';
import { SecurityLogRecord, SecurityLogQueryParams, SecurityLogResponse } from './security_log_types';
import { securityLogService } from './security_log_service';
export const useSecurityLogs = (logType: 'operation' | 'login' | 'data-access') => {
  const [loading, setLoading] = useState(false);
  const [data, setData] = useState<SecurityLogResponse>({
    records: [],
    total: 0,
    currentPage: 1,
    pageSize: 20
  });
  const [error, setError] = useState<string | null>(null);
  const fetchLogs = useCallback(async (params: SecurityLogQueryParams) => {
    setLoading(true);
    setError(null);
    try {
      let response: SecurityLogResponse;
      switch (logType) {
        case 'operation':
          response = await securityLogService.getOperationLogs(params);
          break;
        case 'login':
          response = await securityLogService.getLoginLogs(params);
          break;
        case 'data-access':
          response = await securityLogService.getDataAccessLogs(params);
          break;
        default:
          throw new Error(' 不支持的日志类型');
      }
      setData(response);
    } catch (err) {
      setError(err instanceof Error ? err.message : ' 获取日志数据失败');
    } finally {
      setLoading(false);
    }
  }, [logType]);

```

```

const exportLogs = useCallback(async (params: SecurityLogQueryParams): Promise<boolean> => {
  try {
    const blob = await securityLogService.exportLogs(params, logType);
    const url = window.URL.createObjectURL(blob);
    const link = document.createElement('a');
    link.href = url;
    link.download = `${logType}-logs-${new Date().getTime()}.xlsx`;
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
    window.URL.revokeObjectURL(url);
    return true;
  } catch (err) {
    setError(err instanceof Error ? err.message : '导出日志失败');
    return false;
  }
}, [logType]);
return {
  loading,
  data,
  error,
  fetchLogs,
  exportLogs
};
};
import React from 'react';
import { Card, Form, Input, Select, DatePicker, Button, Space } from 'antd';
import { SearchOutlined, ReloadOutlined } from '@ant-design/icons';
import { SecurityLogQueryParams, ACTION_TYPE_OPTIONS, STATUS_OPTIONS } from './security_log_types';
const { RangePicker } = DatePicker;
const { Option } = Select;
interface SecurityLogFiltersProps {
  onSearch: (params: SecurityLogQueryParams) => void;
  onReset: () => void;
  loading?: boolean;
  showActionType?: boolean;
  showStatus?: boolean;
}
export const SecurityLogFilters: React.FC<SecurityLogFiltersProps> = ({
  onSearch,
  onReset,
  loading = false,
  showActionType = false,
  showStatus = false
}) => {
  const [form] = Form.useForm();
  const handleSearch = (values: any) => {
    const params: SecurityLogQueryParams = {
      page: 1,
      pageSize: 20,
      userName: values.userName,
      ipAddress: values.ipAddress,
      actionType: values.actionType,
      status: values.status
    };
  };

```

```

};
if (values.timeRange && values.timeRange.length === 2) {
  params.startTime = values.timeRange[0];
  params.endTime = values.timeRange[1];
}
onSearch(params);
};
const handleReset = () => {
  form.resetFields();
  onReset();
};
return (
  <Card className="rounded-lg shadow-md mb-4">
    <Form
      form={form}
      layout="inline"
      onFinish={handleSearch}
    >
      <Form.Item name="userName" label="用户名">
        <Input placeholder="请输入用户名" style={{ width: 150 }} />
      </Form.Item>
      <Form.Item name="ipAddress" label="IP 地址">
        <Input placeholder="请输入 IP 地址" style={{ width: 150 }} />
      </Form.Item>
      {showActionType && (
        <Form.Item name="actionType" label="操作类型">
          <Select placeholder="请选择操作类型" style={{ width: 150 }}>
            {ACTION_TYPE_OPTIONS.map(option => (
              <Option key={option.value} value={option.value}>
                {option.label}
              </Option>
            ))}
          </Select>
        </Form.Item>
      )}
      {showStatus && (
        <Form.Item name="status" label="状态">
          <Select placeholder="请选择状态" style={{ width: 120 }}>
            {STATUS_OPTIONS.map(option => (
              <Option key={option.value} value={option.value}>
                {option.label}
              </Option>
            ))}
          </Select>
        </Form.Item>
      )}
      <Form.Item name="timeRange" label="时间范围">
        <RangePicker showTime />
      </Form.Item>
      <Form.Item>
        <Space>
          <Button
            type="primary"
            htmlType="submit"
            icon={<SearchOutlined />}
          </Button>
        </Space>
      </Form.Item>
    </Form>
  </Card>
)

```

```

        loading={loading}
        className="primary-color"
      >
        搜索
      </Button>
      <Button
        onClick={handleReset}
        icon={<ReloadOutlined />}
      >
        重置
      </Button>
    </Space>
  </Form.Item>
</Form>
</Card>
);
};
import React from 'react';
import { Table, Tag, Tooltip } from 'antd';
import { SecurityLogRecord } from './security_log_types';
import { formatDateTime, getActionTypeColor, getStatusColor } from './security_log_utils';
interface SecurityLogTableProps {
  data: SecurityLogRecord[];
  loading?: boolean;
  pagination: any;
  onChange: (pagination: any) => void;
  logType: 'operation' | 'login' | 'data-access';
}
export const SecurityLogTable: React.FC<SecurityLogTableProps> = ({
  data,
  loading,
  pagination,
  onChange,
  logType
}) => {
  const getActionTypeText = (type: string) => {
    const map: Record<string, string> = {
      'LOGIN': '登录',
      'LOGOUT': '登出',
      'DATA_ACCESS': '数据访问',
      'DATA_MODIFY': '数据修改',
      'SYSTEM_OPERATION': '系统操作'
    };
    return map[type] || type;
  };
  const getStatusText = (status: string) => {
    return status === 'SUCCESS' ? '成功' : '失败';
  };
  const baseColumns = [
    {
      title: '用户名',
      dataIndex: 'userName',
      key: 'userName',
      width: 120
    },
  ],

```

```

    {
      title: 'IP 地址',
      dataIndex: 'ipAddress',
      key: 'ipAddress',
      width: 140
    },
    {
      title: ' 操作时间',
      dataIndex: 'timestamp',
      key: 'timestamp',
      width: 180,
      render: (timestamp: Date) => formatDateTime(timestamp)
    },
    {
      title: ' 状态',
      dataIndex: 'status',
      key: 'status',
      width: 80,
      render: (status: string) => (
        <Tag color={getStatusColor(status)}>
          {getStatusText(status)}
        </Tag>
      )
    }
  ];
  const operationColumns = [
    ...baseColumns,
    {
      title: ' 操作类型',
      dataIndex: 'actionType',
      key: 'actionType',
      width: 120,
      render: (type: string) => (
        <Tag color={getActionTypeColor(type)}>
          {getActionTypeText(type)}
        </Tag>
      )
    },
    {
      title: ' 操作描述',
      dataIndex: 'actionDescription',
      key: 'actionDescription',
      ellipsis: true,
      render: (text: string) => (
        <Tooltip title={text}>
          <span>{text}</span>
        </Tooltip>
      )
    }
  ];
  const loginColumns = [
    ...baseColumns,
    {
      title: ' 设备信息',
      dataIndex: 'deviceInfo',

```



```

        key: 'deviceInfo',
        ellipsis: true,
        render: (text: string) => (
          <Tooltip title={text}>
            <span>{text}</span>
          </Tooltip>
        )
      },
      {
        title: ' 用户代理',
        dataIndex: 'userAgent',
        key: 'userAgent',
        ellipsis: true,
        render: (text: string) => (
          <Tooltip title={text}>
            <span>{text}</span>
          </Tooltip>
        )
      }
    ];
    const dataAccessColumns = [
      ...baseColumns,
      {
        title: ' 目标资源',
        dataIndex: 'targetResource',
        key: 'targetResource',
        ellipsis: true,
        render: (text: string) => (
          <Tooltip title={text}>
            <span>{text}</span>
          </Tooltip>
        )
      },
      {
        title: ' 操作描述',
        dataIndex: 'actionDescription',
        key: 'actionDescription',
        ellipsis: true,
        render: (text: string) => (
          <Tooltip title={text}>
            <span>{text}</span>
          </Tooltip>
        )
      }
    ];
    const getColumns = () => {
      switch (logType) {
        case 'operation':
          return operationColumns;
        case 'login':
          return loginColumns;
        case 'data-access':
          return dataAccessColumns;
        default:
          return baseColumns;
      }
    };

```

```

    }
  };
  return (
    <Table
      columns={getColumns()}
      dataSource={data}
      loading={loading}
      pagination={pagination}
      onChange={onChange}
      rowKey="id"
      scroll={{ x: 800 }}
    />
  );
};
import React, { useState, useEffect } from 'react';
import { Card, Button, Space, message } from 'antd';
import { ExportOutlined } from '@ant-design/icons';
import { SecurityLogFilters } from './security_log_filters';
import { SecurityLogTable } from './security_log_table';
import { useSecurityLogs } from './security_log_hooks';
import { SecurityLogQueryParams } from './security_log_types';
export const OperationLogPage: React.FC = () => {
  const [queryParams, setQueryParams] = useState<SecurityLogQueryParams>({
    page: 1,
    pageSize: 20
  });
  const { loading, data, error, fetchLogs, exportLogs } = useSecurityLogs('operation');
  useEffect(() => {
    fetchLogs(queryParams);
  }, [queryParams]);
  useEffect(() => {
    if (error) {
      message.error(error);
    }
  }, [error]);
  const handleSearch = (params: SecurityLogQueryParams) => {
    setQueryParams(params);
  };
  const handleReset = () => {
    setQueryParams({
      page: 1,
      pageSize: 20
    });
  };
  const handleTableChange = (pagination: any) => {
    setQueryParams(prev => ({
      ...prev,
      page: pagination.current,
      pageSize: pagination.pageSize
    }));
  };
  const handleExport = async () => {
    const success = await exportLogs(queryParams);
    if (success) {
      message.success('导出成功');
    }
  };

```

```

    }
  };
  return (
    <div className="page-bg-color p-4">
      <Card className="rounded-lg shadow-md">
        <div className="flex justify-between items-center mb-4">
          <h2 className="text-primary text-lg font-semibold"> 操作日志 </h2>
          <Space>
            <Button
              type="primary"
              icon={<ExportOutlined />}
              onClick={handleExport}
              loading={loading}
              className="accent-color"
            >
              导出日志
            </Button>
          </Space>
        </div>
        <SecurityLogFilters
          onSearch={handleSearch}
          onReset={handleReset}
          loading={loading}
          showActionType={true}
          showStatus={true}
        />
        <SecurityLogTable
          data={data.records}
          loading={loading}
          pagination={{
            current: data.currentPage,
            pageSize: data.pageSize,
            total: data.total,
            showSizeChanger: true,
            showQuickJumper: true,
            showTotal: (total) => '共 ${total} 条记录 '
          }}
          onChange={handleTableChange}
          logType="operation"
        />
      </Card>
    </div>
  );
};
import React, { useState, useEffect } from 'react';
import { Card, Button, Space, message } from 'antd';
import { ExportOutlined } from '@ant-design/icons';
import { SecurityLogFilters } from './security_log_filters';
import { SecurityLogTable } from './security_log_table';
import { useSecurityLogs } from './security_log_hooks';
import { SecurityLogQueryParams } from './security_log_types';
export const LoginLogPage: React.FC = () => {
  const [queryParams, setQueryParams] = useState<SecurityLogQueryParams>({
    page: 1,
    pageSize: 20
  });

```

```

});
const { loading, data, error, fetchLogs, exportLogs } = useSecurityLogs('login');
useEffect(() => {
  fetchLogs(queryParams);
}, [queryParams]);
useEffect(() => {
  if (error) {
    message.error(error);
  }
}, [error]);
const handleSearch = (params: SecurityLogQueryParams) => {
  setQueryParams(params);
};
const handleReset = () => {
  setQueryParams({
    page: 1,
    pageSize: 20
  });
};
const handleTableChange = (pagination: any) => {
  setQueryParams(prev => ({
    ...prev,
    page: pagination.current,
    pageSize: pagination.pageSize
  }));
};
const handleExport = async () => {
  const success = await exportLogs(queryParams);
  if (success) {
    message.success('导出成功');
  }
};
return (
  <div className="page-bg-color p-4">
    <Card className="rounded-lg shadow-md">
      <div className="flex justify-between items-center mb-4">
        <h2 className="text-primary text-lg font-semibold"> 登录日志 </h2>
        <Space>
          <Button
            type="primary"
            icon={<ExportOutlined />}
            onClick={handleExport}
            loading={loading}
            className="accent-color"
          >
            导出日志
          </Button>
        </Space>
      </div>
      <SecurityLogFilters
        onSearch={handleSearch}
        onReset={handleReset}
        loading={loading}
        showStatus={true}
      />
    </Card>
  </div>

```

```

    <SecurityLogTable
      data={data.records}
      loading={loading}
      pagination={{
        current: data.currentPage,
        pageSize: data.pageSize,
        total: data.total,
        showSizeChanger: true,
        showQuickJumper: true,
        showTotal: (total) => `共 ${total} 条记录`
      }}
      onChange={handleTableChange}
      logType="login"
    />
  </Card>
</div>
);
};
import React, { useState, useEffect } from 'react';
import { Card, Button, Space, message } from 'antd';
import { ExportOutlined } from '@ant-design/icons';
import { SecurityLogFilters } from './security_log_filters';
import { SecurityLogTable } from './security_log_table';
import { useSecurityLogs } from './security_log_hooks';
import { SecurityLogQueryParams } from './security_log_types';
export const DataAccessLogPage: React.FC = () => {
  const [queryParams, setQueryParams] = useState<SecurityLogQueryParams>({
    page: 1,
    pageSize: 20
  });
  const { loading, data, error, fetchLogs, exportLogs } = useSecurityLogs('data-access');
  useEffect(() => {
    fetchLogs(queryParams);
  }, [queryParams]);
  useEffect(() => {
    if (error) {
      message.error(error);
    }
  }, [error]);
  const handleSearch = (params: SecurityLogQueryParams) => {
    setQueryParams(params);
  };
  const handleReset = () => {
    setQueryParams({
      page: 1,
      pageSize: 20
    });
  };
  const handleTableChange = (pagination: any) => {
    setQueryParams(prev => ({
      ...prev,
      page: pagination.current,
      pageSize: pagination.pageSize
    }));
  };
};

```

```

const handleExport = async () => {
  const success = await exportLogs(queryParams);
  if (success) {
    message.success('导出成功');
  }
};

return (
  <div className="page-bg-color p-4">
    <Card className="rounded-lg shadow-md">
      <div className="flex justify-between items-center mb-4">
        <h2 className="text-primary text-lg font-semibold"> 数据访问日志 </h2>
        <Space>
          <Button
            type="primary"
            icon={<ExportOutlined />}
            onClick={handleExport}
            loading={loading}
            className="accent-color"
          >
            导出日志
          </Button>
        </Space>
      </div>
      <SecurityLogFilters
        onSearch={handleSearch}
        onReset={handleReset}
        loading={loading}
      />
      <SecurityLogTable
        data={data.records}
        loading={loading}
        pagination={{
          current: data.currentPage,
          pageSize: data.pageSize,
          total: data.total,
          showSizeChanger: true,
          showQuickJumper: true,
          showTotal: (total) => '共 ${total} 条记录 '
        }}
        onChange={handleTableChange}
        logType="data-access"
      />
    </Card>
  </div>
);
};

export const formatDate = (date: Date): string => {
  return new Intl.DateTimeFormat('zh-CN', {
    year: 'numeric',
    month: '2-digit',
    day: '2-digit',
    hour: '2-digit',
    minute: '2-digit',
    second: '2-digit'
  }).format(new Date(date));
};

```

```
};  
export const getActionTypeColor = (actionType: string): string => {  
  const colorMap: Record<string, string> = {  
    'LOGIN': 'green',  
    'LOGOUT': 'blue',  
    'DATA_ACCESS': 'orange',  
    'DATA_MODIFY': 'red',  
    'SYSTEM_OPERATION': 'purple'  
  };  
  return colorMap[actionType] || 'default';  
};  
export const getStatusColor = (status: string): string => {  
  return status === 'SUCCESS' ? 'success' : 'error';  
};  
export const downloadBlob = (blob: Blob, filename: string): void => {  
  const url = window.URL.createObjectURL(blob);  
  const link = document.createElement('a');  
  link.href = url;  
  link.download = filename;  
  document.body.appendChild(link);  
  link.click();  
  document.body.removeChild(link);  
  window.URL.revokeObjectURL(url);  
};
```