Phone Auth API - 程式碼更新部署文件

📌 更新摘要

主要修正 `verify_otp` 函式的驗證邏輯，從模擬實作改為真實的 Firebase ID Token 驗證。

更新內容

- ✅ 修正 `verify_otp` 方法，實作真實 Firebase 驗證
- ✅ 改善 LOCKED 狀態錯誤訊息，增加 `retry_after` 欄位
- ✅ 更新錯誤處理機制，提供更明確的錯誤訊息

受影響的檔案

- 📁 `phone_auth/firebase_service.py` - 核心驗證邏輯
- 📁 `phone_auth/views.py` - API 回應格式

🔧 需要更新的檔案

1️⃣ firebase_service.py

檔案位置： `phone_auth/firebase_service.py`

需要更新的函式： `verify_otp` 方法（第 113-147 行）

原始程式碼（請刪除）

```
def verify_otp(self, verification_id: str, otp_code: str) -> dict:
    """
    驗證 OTP 代碼

    注意：本專案採用 6 位 OTP（verification_id + otp_code）為唯一流程。
    若在你的架構中需與 Firebase 前端驗證整合，請依實際需求調整此方法實作。

    Args:
        verification_id: Firebase 返回的驗證 session ID
        otp_code: 使用者輸入的驗證碼

    Returns:
        dict: {
            'success': bool,
            'phone_number': str (如果成功),
            'uid': str (Firebase user ID),
            'error': str (如果失敗)
        }
    """
    try:
        # 這裡提供一個簡化的實作範例
```

```
        logger.info(f"驗證 OTP：verification_id={verification_id}, code=
{otp_code}")

        # 模擬驗證（實際應對接你的 OTP 驗證機制）

        return {
            'success': True,
            'message': '此為模擬實作，請替換為實際的 OTP 驗證邏輯'
        }
    except Exception as e:
        logger.error(f"驗證 OTP 時發生錯誤：{str(e)}")
        return {
            'success': False,
            'error': str(e)
        }
```

新程式碼（請替換）

```
def verify_otp(self, verification_id: str, otp_code: str) -> dict:
    """
    驗證 OTP 代碼（使用 Firebase ID Token）

    實際流程：
    1. 前端使用 Firebase JS SDK 完成 phone auth（signInWithPhoneNumber +
confirm）
    2. 成功後 Firebase 返回 idToken
    3. 前端將 idToken 作為 verification_id 傳給後端
    4. 後端驗證 idToken 並取得已驗證的手機號碼

    Args:
        verification_id: Firebase ID Token（前端驗證成功後取得）
        otp_code: 使用者輸入的驗證碼（用於前端驗證，後端僅記錄）

    Returns:
        dict: {
            'success': bool,
            'phone_number': str (如果成功),
            'uid': str (Firebase user ID),
            'error': str (如果失敗)
        }
    """
    try:
        logger.info(f"開始驗證 Firebase ID Token，OTP code: {otp_code}")

        # 使用 Firebase Admin SDK 驗證 ID Token
        decoded_token = auth.verify_id_token(verification_id)

        # 從 token 中取得已驗證的資訊
        uid = decoded_token.get('uid')
        phone_number = decoded_token.get('phone_number')
```

```
        if not phone_number:
            logger.warning("ID Token 中沒有手機號碼資訊")
            return {
                'success': False,
                'error': 'ID Token 中沒有手機號碼資訊，請確認前端使用 Phone Auth
方式登入'
            }

        logger.info(f"驗證成功：uid={uid}, phone={phone_number}")

        return {
            'success': True,
            'phone_number': phone_number,
            'uid': uid,
            'message': '手機號碼驗證成功'
        }

    except auth.InvalidIdTokenError:
        logger.error("無效的 ID Token")
        return {
            'success': False,
            'error': '驗證碼已過期或無效，請重新發送'
        }
    except auth.ExpiredIdTokenError:
        logger.error("ID Token 已過期")
        return {
            'success': False,
            'error': '驗證碼已過期，請重新發送'
        }
    except Exception as e:
        logger.error(f"驗證 OTP 時發生錯誤：{str(e)}")
        return {
            'success': False,
            'error': f'驗證失敗：{str(e)}'
        }
```

2️⃣ views.py

檔案位置： `phone_auth/views.py`

需要更新的位置： 共 3 處

更新 1：verify_otp 函式 - LOCKED 狀態檢查（第 223-230 行）

原始程式碼：

```
# 檢查是否已鎖定
if user.verification_status == CustomUser.VerificationStatus.LOCKED:
    logger.warning(f"使用者 {user.username} 已被鎖定")
    return Response(
```

```
            {
                'status': 'LOCKED',
                'message': '驗證失敗次數過多，請重新發送驗證碼'
            },
            status=status.HTTP_403_FORBIDDEN
        )
```

新程式碼：

```
    # 檢查是否已鎖定
    if user.verification_status == CustomUser.VerificationStatus.LOCKED:
        logger.warning(f"使用者 {user.username} 已被鎖定")
        return Response(
            {
                'status': 'LOCKED',
                'message': '驗證失敗次數過多，請 60 秒後重新發送驗證碼',
                'retry_after': 60
            },
            status=status.HTTP_403_FORBIDDEN
        )
```

更新 2：verify_otp 函式 - 驗證成功處理（第 236-264 行）

原始程式碼：

```
    result = firebase_service.verify_otp(verification_id, otp_code)

    if result.get('success'):
        # 模擬驗證成功（實際應從 result 取得已驗證的手機號碼）
        verified_phone = user.phone_number

        user.phone_verified = True
        user.verification_status = CustomUser.VerificationStatus.VERIFIED
        user.otp_attempts = 0
        user.save()

        # 記錄日誌
        OTPVerificationLog.objects.create(
            user=user,
            phone_number=verified_phone or '',
            action='VERIFY_SUCCESS',
            success=True
        )

        logger.info(f"手機驗證成功：user={user.username}, phone=
    {verified_phone}")

        return Response(
            {
```

```
            'status': 'VERIFIED',
            'phone_number': verified_phone,
            'message': '手機號碼驗證成功'
        },
        status=status.HTTP_200_OK
    )
```

新程式碼：

```
result = firebase_service.verify_otp(verification_id, otp_code)

if result.get('success'):
    # 從 Firebase 驗證結果取得已驗證的手機號碼
    verified_phone = result.get('phone_number')
    firebase_uid = result.get('uid')

    # 檢查手機號碼是否與使用者當前綁定的號碼一致
    if user.phone_number and user.phone_number != verified_phone:
        logger.warning(f"手機號碼不符：user.phone={user.phone_number},
verified={verified_phone}")
        return Response(
            {
                'error': 'PHONE_MISMATCH',
                'message': '驗證的手機號碼與您的帳號不符，請確認'
            },
            status=status.HTTP_400_BAD_REQUEST
        )

    # 更新使用者資料
    user.phone_number = verified_phone
    user.phone_verified = True
    user.verification_status = CustomUser.VerificationStatus.VERIFIED
    user.otp_attempts = 0
    user.save()

    # 記錄日誌
    OTPVerificationLog.objects.create(
        user=user,
        phone_number=verified_phone or '',
        action='VERIFY_SUCCESS',
        success=True
    )

    logger.info(f"手機驗證成功：user={user.username}, phone={verified_phone},
uid={firebase_uid}")

    return Response(
        {
            'status': 'VERIFIED',
            'phone_number': verified_phone,
            'message': '手機號碼驗證成功'
        },
```

```
            status=status.HTTP_200_OK
        )
```

更新 3：verify_otp 函式 - 驗證失敗後的 LOCKED 回應（第 278-285 行）

原始程式碼：

```
if user.verification_status == CustomUser.VerificationStatus.LOCKED:
    return Response(
        {
            'status': 'LOCKED',
            'message': '驗證失敗次數過多，請重新發送驗證碼'
        },
        status=status.HTTP_403_FORBIDDEN
    )
```

新程式碼：

```
if user.verification_status == CustomUser.VerificationStatus.LOCKED:
    return Response(
        {
            'status': 'LOCKED',
            'message': '驗證失敗次數過多，請 60 秒後重新發送驗證碼',
            'retry_after': 60
        },
        status=status.HTTP_403_FORBIDDEN
    )
```

📊 更新前後 API 回應變化

LOCKED 狀態回應（新增 retry_after）：

```
{
  "status": "LOCKED",
- "message": "驗證失敗次數過多，請重新發送驗證碼"
+ "message": "驗證失敗次數過多，請 60 秒後重新發送驗證碼",
+ "retry_after": 60
}
```

驗證失敗回應（新增詳細錯誤）：

```
{
  "status": "INVALID_OTP",
- "error": "驗證失敗"
```

```
+ "error": "驗證碼已過期或無效，請重新發送"
}
```