源代码

（软件著作权登记用）

软件名称：宠物档案系统 V1.0

著作权人：北京创物原宠物科技有限公司

开发完成日期：2026 年 8 月 17 日

第一部分　源代码（前 30 页，每页 50 行）

```
0001  /* ===== FILE: server/server.js (JavaScript) ===== */
0002  #!/usr/bin/env node
0003  /**
0004   * server.js - Pet Archive API Server
0005   *
0006   * 功能：
0007   * 1. 提供宠物列表 API（支持分类、搜索）
0008   * 2. 提供宠物详情 API
0009   * 3. 提供静态文件服务
0010   */
0011
0012  require('dotenv').config();
0013  const express = require('express');
0014  const path = require('path');
0015  const mysql = require('mysql2/promise');
0016  const { initRedis, cachedGet, bumpVer, delCache } = require('./redis-client');
0017
0018  const APP_PORT = process.env.APP_PORT || 3003;
0019
0020  const MYSQL_CONFIG = {
0021    host: process.env.MYSQL_HOST || '8.140.244.180',
0022    user: process.env.MYSQL_USER || 'root',
0023    password: process.env.MYSQL_PASSWORD || '4a926b9fed6c0d25',
0024    database: process.env.MYSQL_DATABASE || 'bjcwy_pc',
0025  };
0026
0027  // 艺术家数据
0028  const ARTISTS_DATA = [
0029    { code: 'xiaoju', name: '画师小橘', title: '海洋萌宠设计师', description: '深耕水生海洋萌宠绘制，画风柔和治愈。', followerCount: 3680, totalAuthorization: 22400, creatorLevel: 'S' },
0030    { code: 'achai', name: '阿柴画社', title: '治愈猫咪主理人', description: '专注治愈系猫咪萌宠创作。', followerCount: 2890, totalAuthorization: 18900, creatorLevel: 'A' },
0031    { code: 'yishou', name: '异兽造梦师', title: '幻想神兽缔造者', description: '擅长上古神兽和神话生物创作。', followerCount: 4200, totalAuthorization: 15620, creatorLevel: 'S' },
0032    { code: 'yinghuo', name: '萤火画师', title: '微光精灵创作者', description: '专注小精灵系列萌宠。', followerCount: 1860, totalAuthorization: 12300, creatorLevel: 'A' },
0033    { code: 'yunjing', name: '云鲸画师', title: '天空幻想家', description: '擅长天空主题萌宠创作。', followerCount: 1520, totalAuthorization: 10800, creatorLevel: 'B' },
0034    { code: 'mengchong', name: '萌宠小筑', title: '治愈系画师', description: '温暖治愈风格。', followerCount: 2150, totalAuthorization: 7200, creatorLevel: 'B' },
0035    { code: 'xigaodi', name: '西高地工坊', title: '萌犬专家', description: '专注狗狗萌宠创作。', followerCount: 3200, totalAuthorization: 6180, creatorLevel: 'A' },
0036    { code: 'luhuahua', name: '鹿花花', title: '森系精灵画师', description: '擅长森林主题萌宠创作。', followerCount: 1980, totalAuthorization: 5680, creatorLevel: 'B' },
0037    { code: 'huohua', name: '火华', title: '国潮新锐画师', description: '将传统国潮元素与萌宠结合。', followerCount: 2450, totalAuthorization: 8920, creatorLevel: 'A' }
0038  ];
0039
0040  // 宠物-艺术家关联
0041  const PET_ARTIST_MAP = {
0042    'westie': 'xigaodi', 'seal': 'xiaoju', 'baxi-turtle': 'xiaoju', 'penguin': 'xiaoju', 'sea-lion': 'xiaoju',
0043    'otter': 'xiaoju', 'samoyed': 'xigaodi', 'corgi': 'xigaodi', 'shiba': 'xigaodi', 'golden-retriever': 'xigaodi',
0044    'husky': 'xigaodi', 'chihuahua': 'xigaodi', 'orange-cat': 'achai', 'ragdoll-cat': 'achai', 'mainecoon': 'achai',
0045    'persian-cat': 'achai', 'azure-dragon': 'yishou', 'vermilion-bird': 'yishou', 'white-tiger': 'yishou',
0046    'pixiu': 'yishou', 'suanni': 'yishou', 'jiu-wei-hu': 'yishou', 'unicorn': 'yunjing', 'fenghuang': 'yunjing',
0047    'succulent-spirit': 'yinghuo', 'koala': 'mengchong', 'red-panda': 'mengchong', 'squirrel': 'mengchong',
0048    'groundhog': 'mengchong', 'hedgehog': 'mengchong', 'fox': 'luhuahua', 'raccoon': 'mengchong', 'sea-hare': 'xiaoju',
0049    'flamingo': 'huohua', 'serama-chicken': 'mengchong', 'bichon': 'xigaodi', 'schnauzer': 'xigaodi',
```

```
0050    'border-collie': 'xigaodi', 'tiger-striped-cat': 'achai', 'long-cat': 'mengchong',
'honey-quoll': 'mengchong',
```

```
0051    'pig': 'luhuahua', 'egret': 'huohua', 'fancy-mouse': 'huohua', 'garfield': 'achai',
'brown-bear': 'huohua',
0052    'koel-duck': 'xiaoju', 'black-shiba': 'xigaodi'
0053  };
0054
0055  // 连接池（新增互动功能使用）
0056  const pool = mysql.createPool({
0057    ...MYSQL_CONFIG,
0058    waitForConnections: true,
0059    connectionLimit: 10,
0060    enableKeepAlive: true,
0061    keepAliveInitialDelay: 10000,
0062    connectTimeout: 10000
0063  });
0064
0065  // 输入验证
0066  function validateInput(input, maxLength = 100) {
0067    if (!input) return null;
0068    const sanitized = String(input).trim().slice(0, maxLength);
0069    if (/[<>\"';&|]/.test(sanitized)) return null;
0070    return sanitized;
0071  }
0072
0073  const app = express();
0074  app.use(express.json());
0075
0076  // CORS
0077  app.use((req, res, next) => {
0078    res.header('Access-Control-Allow-Origin', '*');
0079    res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type,
Accept');
0080    next();
0081  });
0082
0083  // 创建路由器
0084  const router = express.Router();
0085
0086  // 1. 宠物列表 API（缓存首页数据，1 天）
0087  router.get('/pets', async (req, res) => {
0088    try {
0089      const { category, search, page = 1, limit = 50 } = req.query;
0090      const cacheKey = `pets:list:${category || 'all'}:${search || ''}:${page}:${limit}`;
0091
0092      const data = await cachedGet(cacheKey, 86400, async () => {
0093        let sql = 'SELECT id, code, name, description, category, categoryTab, tags,
effortImage FROM pet_archive WHERE 1=1';
0094        const params = [];
0095
0096        if (category) {
0097          sql += ' AND (category = ? OR categoryTab = ?)';
0098          params.push(category, category);
0099        }
0100
```

```
0101        if (search) {
0102          sql += ' AND (name LIKE ? OR description LIKE ?)';
0103          params.push(`%${search}%`, `%${search}%`);
0104        }
0105
0106        sql += ' ORDER BY id LIMIT ? OFFSET ?';
0107        params.push(parseInt(limit), (parseInt(page) - 1) * parseInt(limit));
0108
0109        const [rows] = await pool.execute(sql, params);
0110
0111        return rows.map(p => ({
0112          ...p,
0113          tags: p.tags ? JSON.parse(p.tags) : []
0114        }));
0115      });
0116
0117      res.json({ success: true, data });
0118    } catch (e) {
0119      res.json({ success: false, error: e.message });
0120    }
0121  });
0122
0123  // 批量宠物统计（缓存 1 天）
0124  router.get('/pets/stats-bulk', async (req, res) => {
0125    try {
0126      const data = await cachedGet('stats:bulk', 86400, async () => {
0127        try {
0128          const [[likes], [collects], [adopts], [flowers]] = await Promise.all([
0129            pool.execute('SELECT pet_code, COUNT(*) as cnt FROM pet_likes GROUP BY pet_code'),
0130            pool.execute("SELECT pet_code, COUNT(*) as cnt FROM pet_favorites WHERE action =
'favorite' GROUP BY pet_code"),
0131            pool.execute('SELECT pet_code, COUNT(*) as cnt FROM pet_adopts GROUP BY
pet_code'),
0132            pool.execute('SELECT pet_code, COUNT(*) as cnt FROM pet_flowers GROUP BY
pet_code')
0133          ]);
0134          const stats = {};
0135          [likes, collects, adopts, flowers].forEach(rows => {
0136            rows.forEach(r => {
0137              if (!stats[r.pet_code]) stats[r.pet_code] = { likes: 0, collects: 0, adopts: 0,
flowers: 0 };
0138              if (r.cnt > 0) {
0139                if (rows === likes) stats[r.pet_code].likes = r.cnt;
0140                else if (rows === collects) stats[r.pet_code].collects = r.cnt;
0141                else if (rows === adopts) stats[r.pet_code].adopts = r.cnt;
0142                else if (rows === flowers) stats[r.pet_code].flowers = r.cnt;
0143              }
0144            });
0145          });
0146          return stats;
0147        } catch (e) { return {}; }
0148      });
0149      res.json({ success: true, data });
0150    } catch (e) { res.json({ success: false, error: e.message }); }
```

```
0151  });
0152
0153  // 2. 宠物详情 API
0154  router.get('/pets/:code', async (req, res) => {
0155    try {
0156      const { code } = req.params;
0157
0158      const [rows] = await pool.execute(
0159        'SELECT p.*, a.code as artistCode, a.name as artistName, a.title as artistTitle FROM
pet_archive p LEFT JOIN pet_artist a ON p.artistId = a.id WHERE p.code = ?',
0160        [code]
0161      );
0162
0163      if (rows.length === 0) {
0164        return res.json({ success: false, error: 'Pet not found' });
0165      }
0166
0167      const [stages] = await pool.execute(
0168        'SELECT level, imageUrl, styleName FROM pet_archive_stage WHERE petArchiveId = ?
ORDER BY level, styleName',
0169        [rows[0].id]
0170      );
0171
0172      const pet = {
0173        ...rows[0],
0174        tags: rows[0].tags ? JSON.parse(rows[0].tags) : [],
0175        stages: stages.filter(s => !s.styleName).map(s => ({ level: s.level, image:
s.imageUrl })),
0176        styleStages: stages.filter(s => s.styleName).reduce((acc, s) => {
0177          if (!acc[s.styleName]) acc[s.styleName] = [];
0178          acc[s.styleName].push({ level: s.level, image: s.imageUrl });
0179          return acc;
0180        }, {})
0181      };
0182
0183      res.json({ success: true, data: pet });
0184    } catch (e) {
0185      res.json({ success: false, error: e.message });
0186    }
0187  });
0188
0189  // 3. 分类列表 API
0190  router.get('/categories', async (req, res) => {
0191    try {
0192      const [rows] = await pool.execute(
0193        'SELECT DISTINCT category, categoryTab FROM pet_archive WHERE category IS NOT NULL
AND category != ""'
0194      );
0195
0196      const categories = rows.map(r => r.category || r.categoryTab).filter(Boolean);
0197      res.json({ success: true, data: [...new Set(categories)] });
0198    } catch (e) {
0199      res.json({ success: false, error: e.message });
0200    }
```

```
0201  });
0202
0203  // 4. 画师列表API（缓存1天）
0204  app.get('/apic/artists', async (req, res) => {
0205    try {
0206      const { page = 1, limit = 20, level } = req.query;
0207      const cacheKey = `artists:list:${level || 'all'}:${page}:${limit}`;
0208
0209      const data = await cachedGet(cacheKey, 86400, async () => {
0210        let sql = `SELECT a.id, a.code, a.name, a.title, a.description, a.avatar_url as
avatarUrl,
0211                   a.follower_count as followerCount, a.total_authorization as
totalAuthorization,
0212                   a.creator_level as creatorLevel,
0213                   (SELECT COUNT(*) FROM pet_archive p WHERE p.artistId = a.id) as petCount
0214                   FROM pet_artist a WHERE a.status = "active"`;
0215        const params = [];
0216
0217        if (level) {
0218          sql += ' AND creator_level = ?';
0219          params.push(level);
0220        }
0221
0222        sql += ' ORDER BY total_authorization DESC LIMIT ? OFFSET ?';
0223        params.push(parseInt(limit), (parseInt(page) - 1) * parseInt(limit));
0224
0225        const [rows] = await pool.execute(sql, params);
0226        return rows;
0227      });
0228
0229      res.json({ success: true, data });
0230    } catch (e) {
0231      res.json({ success: false, error: e.message });
0232    }
0233  });
0234
0235  // 5. 画师详情API
0236  router.get('/artists/:code', async (req, res) => {
0237    try {
0238      const { code } = req.params;
0239
0240      const [artistRows] = await pool.execute(
0241        'SELECT id, code, name, title, description, avatar_url as avatarUrl, follower_count
as followerCount, total_authorization as totalAuthorization, creator_level as creatorLevel FROM
pet_artist WHERE code = ?',
0242        [code]
0243      );
0244
0245      if (artistRows.length === 0) {
0246        return res.json({ success: false, error: 'Artist not found' });
0247      }
0248
0249      const [petRows] = await pool.execute(
0250        'SELECT id, code, name, description, tags, rarity, petTalk, petHabit, sceneType FROM
pet_archive WHERE artistId = ?',
```

```
0251        [artistRows[0].id]
0252      );
0253
0254      const artist = {
0255        ...artistRows[0],
0256        pets: petRows.map(p => ({
0257          ...p,
0258          tags: p.tags ? JSON.parse(p.tags) : []
0259        }))
0260      };
0261
0262      res.json({ success: true, data: artist });
0263    } catch (e) {
0264      res.json({ success: false, error: e.message });
0265    }
0266  });
0267
0268  // 6. 关注/取消关注画师 API
0269  router.post('/artists/:code/follow', async (req, res) => {
0270    try {
0271      const { code } = req.params;
0272      const { userId, follow } = req.body;
0273
0274      const [artistRows] = await pool.execute('SELECT id FROM pet_artist WHERE code = ?',
[code]);
0275      if (artistRows.length === 0) {
0276        return res.json({ success: false, error: 'Artist not found' });
0277      }
0278
0279      const artistId = artistRows[0].id;
0280
0281      if (follow) {
0282        await pool.execute(
0283          'INSERT IGNORE INTO pet_artist_follow (user_id, artist_id) VALUES (?, ?)',
0284          [userId, artistId]
0285        );
0286      } else {
0287        await pool.execute(
0288          'DELETE FROM pet_artist_follow WHERE user_id = ? AND artist_id = ?',
0289          [userId, artistId]
0290        );
0291      }
0292
0293      const [countRows] = await pool.execute(
0294        'SELECT COUNT(*) as count FROM pet_artist_follow WHERE artist_id = ?',
0295        [artistId]
0296      );
0297
0298      await pool.execute(
0299        'UPDATE pet_artist SET follower_count = ? WHERE id = ?',
0300        [countRows[0].count, artistId]
```

```
0301      );
0302
0303      await bumpVer('artists');
0304
0305      res.json({ success: true, data: { followed: follow, followerCount:
countRows[0].count } });
0306    } catch (e) {
0307      res.json({ success: false, error: e.message });
0308    }
0309  });
0310
0311  // 7. 排行榜 API
0312  router.get('/rank', async (req, res) => {
0313    try {
0314      const { type = 'adopt', period = 'week' } = req.query;
0315
0316      const [rows] = await pool.execute(
0317        `SELECT r.rank_num as rankNum, r.artist_id as artistId, r.rank_value as rankValue,
a.name, a.title, a.avatar_url as avatarUrl, a.creator_level as creatorLevel
0318         FROM pet_rank r
0319         JOIN pet_artist a ON r.artist_id = a.id
0320         WHERE r.rank_type = ? AND r.period = ?
0321         ORDER BY r.rank_num ASC`,
0322        [type, period]
0323      );
0324
0325      res.json({ success: true, data: rows });
0326    } catch (e) {
0327      res.json({ success: false, error: e.message });
0328    }
0329  });
0330
0331  // 8. 获取宠物带画师信息 API
0332  router.get('/pets-with-artist', async (req, res) => {
0333    try {
0334      const { category, search, page = 1, limit = 50 } = req.query;
0335
0336      let sql = `SELECT p.*, a.code as artistCode, a.name as artistName, a.title as
artistTitle
0337                 FROM pet_archive p
0338                 LEFT JOIN pet_artist a ON p.artistId = a.id
0339                 WHERE 1=1`;
0340      const params = [];
0341
0342      if (category) {
0343        sql += ' AND (p.category = ? OR p.categoryTab = ?)';
0344        params.push(category, category);
0345      }
0346
0347      if (search) {
0348        sql += ' AND (p.name LIKE ? OR p.description LIKE ?)';
0349        params.push(`%${search}%`, `%${search}%`);
0350      }
```

```
0351
0352      sql += ' ORDER BY p.id LIMIT ? OFFSET ?';
0353      params.push(parseInt(limit), (parseInt(page) - 1) * parseInt(limit));
0354
0355      const [rows] = await conn.execute(sql, params);
0356      await conn.end();
0357
0358      const pets = rows.map(p => ({
0359        ...p,
0360        tags: p.tags ? JSON.parse(p.tags) : []
0361      }));
0362
0363      res.json({ success: true, data: pets });
0364    } catch (e) {
0365      res.json({ success: false, error: e.message });
0366    }
0367  });
0368
0369  // ========== 用户/互动 API（新增功能）==========
0370
0371  // 用户注册
0372  router.post('/users/register', async (req, res) => {
0373    try {
0374      const { username, password } = req.body;
0375      const safeUsername = validateInput(username, 50);
0376      const safePassword = validateInput(password, 100);
0377      if (!safeUsername || !safePassword) return res.json({ success: false, error: '用户名和密
码不能为空' });
0378      const [rows] = await pool.execute('SELECT id FROM users WHERE username = ?',
[safeUsername]);
0379      if (rows.length > 0) return res.json({ success: false, error: '用户名已存在' });
0380      const [result] = await pool.execute(
0381        "INSERT INTO users (username, password, nickname, role) VALUES (?, ?, ?, 'teacher')",
0382        [safeUsername, safePassword, safeUsername]
0383      );
0384      res.json({ success: true, data: { id: result.insertId, username: safeUsername, isTrial:
true } });
0385    } catch (e) { res.json({ success: false, error: e.message }); }
0386  });
0387
0388  // 用户登录
0389  router.post('/users/login', async (req, res) => {
0390    try {
0391      const { username, password } = req.body;
0392      const safeUsername = validateInput(username, 50);
0393      const safePassword = validateInput(password, 100);
0394      if (!safeUsername || !safePassword) return res.json({ success: false, error: '用户名和密
码不能为空' });
0395      const [rows] = await pool.execute(
0396        'SELECT id, username FROM users WHERE username = ? AND password = ?',
0397        [safeUsername, safePassword]
0398      );
0399      if (rows.length === 0) return res.json({ success: false, error: '用户名或密码错误' });
0400      res.json({ success: true, data: rows[0] });
```

```
0401    } catch (e) { res.json({ success: false, error: e.message }); }
0402  });
0403
0404  // 喜欢切换
0405  router.post('/likes/toggle', async (req, res) => {
0406    try {
0407      const { userId, petCode } = req.body;
0408      const safeUserId = parseInt(userId) || 0;
0409      const safePetCode = validateInput(petCode, 50);
0410      if (!safeUserId || !safePetCode) return res.json({ success: false, error: '参数错误' });
0411      const [exists] = await pool.execute('SELECT id FROM pet_likes WHERE user_id = ? AND
pet_code = ?', [safeUserId, safePetCode]);
0412      if (exists.length > 0) await pool.execute('DELETE FROM pet_likes WHERE user_id = ? AND
pet_code = ?', [safeUserId, safePetCode]);
0413      else await pool.execute('INSERT INTO pet_likes (user_id, pet_code) VALUES (?, ?)',
[safeUserId, safePetCode]);
0414      const [countRows] = await pool.execute('SELECT COUNT(*) as cnt FROM pet_likes WHERE
pet_code = ?', [safePetCode]);
0415      bumpVer('stats:bulk');
0416      res.json({ success: true, data: { liked: exists.length === 0, count:
countRows[0].cnt } });
0417    } catch (e) { res.json({ success: false, error: e.message }); }
0418  });
0419
0420  // 领养切换
0421  router.post('/adopts/toggle', async (req, res) => {
0422    try {
0423      const { userId, petCode } = req.body;
0424      const safeUserId = parseInt(userId) || 0;
0425      const safePetCode = validateInput(petCode, 50);
0426      if (!safeUserId || !safePetCode) return res.json({ success: false, error: '参数错误' });
0427      const [exists] = await pool.execute('SELECT id FROM pet_adopts WHERE user_id = ? AND
pet_code = ?', [safeUserId, safePetCode]);
0428      if (exists.length > 0) await pool.execute('DELETE FROM pet_adopts WHERE user_id = ? AND
pet_code = ?', [safeUserId, safePetCode]);
0429      else await pool.execute('INSERT INTO pet_adopts (user_id, pet_code) VALUES (?, ?)',
[safeUserId, safePetCode]);
0430      const [countRows] = await pool.execute('SELECT COUNT(*) as cnt FROM pet_adopts WHERE
pet_code = ?', [safePetCode]);
0431      bumpVer('stats:bulk');
0432      delCache(`user:${safeUserId}:adopts`);
0433      res.json({ success: true, data: { adopted: exists.length === 0, count:
countRows[0].cnt } });
0434    } catch (e) { res.json({ success: false, error: e.message }); }
0435  });
0436
0437  // 送小红花
0438  router.post('/flowers/send', async (req, res) => {
0439    try {
0440      const { userId, petCode } = req.body;
0441      const safeUserId = parseInt(userId) || 0;
0442      const safePetCode = validateInput(petCode, 50);
0443      if (!safeUserId || !safePetCode) return res.json({ success: false, error: '参数错误' });
0444      await pool.execute('INSERT INTO pet_flowers (user_id, pet_code) VALUES (?, ?)',
[safeUserId, safePetCode]);
0445      const [countRows] = await pool.execute('SELECT COUNT(*) as cnt FROM pet_flowers WHERE
pet_code = ?', [safePetCode]);
0446      bumpVer('stats:bulk');
0447      res.json({ success: true, data: { count: countRows[0].cnt } });
0448    } catch (e) { res.json({ success: false, error: e.message }); }
0449  });
0450
```

```
0451  // 收藏切换（使用原项目 pet_favorites 表）
0452  router.post('/favorites/toggle', async (req, res) => {
0453    try {
0454      const { userId, petCode } = req.body;
0455      const safeUserId = parseInt(userId) || 0;
0456      const safePetCode = validateInput(petCode, 50);
0457      if (!safeUserId || !safePetCode) return res.json({ success: false, error: '参数错误' });
0458      const [exists] = await pool.execute(
0459        "SELECT id FROM pet_favorites WHERE student_id = ? AND pet_code = ? AND action =
'favorite'",
0460        [safeUserId, safePetCode]
0461      );
0462      if (exists.length > 0) {
0463        await pool.execute("DELETE FROM pet_favorites WHERE student_id = ? AND pet_code = ?
AND action = 'favorite'", [safeUserId, safePetCode]);
0464        bumpVer('stats:bulk');
0465        delCache(`user:${safeUserId}:favorites`);
0466        return res.json({ success: true, data: { favorited: false } });
0467      }
0468      await pool.execute("INSERT INTO pet_favorites (pet_code, student_id, action) VALUES
(?, ?, 'favorite')", [safePetCode, safeUserId]);
0469      bumpVer('stats:bulk');
0470      delCache(`user:${safeUserId}:favorites`);
0471      res.json({ success: true, data: { favorited: true } });
0472    } catch (e) { res.json({ success: false, error: e.message }); }
0473  });
0474
0475  // 关注切换（使用原项目 pet_artist_follow 表）
0476  router.post('/follows/toggle', async (req, res) => {
0477    try {
0478      const { userId, artistCode } = req.body;
0479      const safeUserId = parseInt(userId) || 0;
0480      const safeArtistCode = validateInput(artistCode, 50);
0481      if (!safeUserId || !safeArtistCode) return res.json({ success: false, error: '参数错误
' });
0482      const [artistRows] = await pool.execute('SELECT id FROM pet_artist WHERE code = ?',
[safeArtistCode]);
0483      if (artistRows.length === 0) return res.json({ success: false, error: '画师不存在' });
0484      const artistId = artistRows[0].id;
0485      const [exists] = await pool.execute('SELECT id FROM pet_artist_follow WHERE user_id = ?
AND artist_id = ?', [safeUserId, artistId]);
0486      let followed = false;
0487      if (exists.length > 0) await pool.execute('DELETE FROM pet_artist_follow WHERE user_id
= ? AND artist_id = ?', [safeUserId, artistId]);
0488      else { await pool.execute('INSERT INTO pet_artist_follow (user_id, artist_id) VALUES
(?, ?)', [safeUserId, artistId]); followed = true; }
0489      const [countRows] = await pool.execute('SELECT COUNT(*) as cnt FROM pet_artist_follow
WHERE artist_id = ?', [artistId]);
0490      bumpVer('artists');
0491      res.json({ success: true, data: { followed, followerCount: countRows[0].cnt } });
0492    } catch (e) { res.json({ success: false, error: e.message }); }
0493  });
0494
0495  // 获取用户关注的艺术家列表
0496  app.get('/apic/follows/list', async (req, res) => {
0497    try {
0498      const { userId } = req.query;
0499      const safeUserId = parseInt(userId) || 0;
0500      if (!safeUserId) return res.json({ success: true, data: [] });
```

```
0501      const [rows] = await pool.execute(
0502        'SELECT a.code FROM pet_artist_follow f JOIN pet_artist a ON f.artist_id = a.id WHERE
f.user_id = ?',
0503        [safeUserId]
0504      );
0505      res.json({ success: true, data: rows.map(r => r.code) });
0506    } catch (e) { res.json({ success: true, data: [] }); }
0507  });
0508
0509  // 获取单个宠物统计
0510  app.get('/apic/pets/:code/stats', async (req, res) => {
0511    try {
0512      const { code } = req.params;
0513      const safeCode = validateInput(code, 50);
0514      const [[likes], [collects], [adopts], [flowers]] = await Promise.all([
0515        pool.execute('SELECT COUNT(*) as cnt FROM pet_likes WHERE pet_code = ?', [safeCode]),
0516        pool.execute("SELECT COUNT(*) as cnt FROM pet_favorites WHERE pet_code = ? AND action
= 'favorite'", [safeCode]),
0517        pool.execute('SELECT COUNT(*) as cnt FROM pet_adopts WHERE pet_code = ?', [safeCode]),
0518        pool.execute('SELECT COUNT(*) as cnt FROM pet_flowers WHERE pet_code = ?', [safeCode])
0519      ]);
0520      res.json({ success: true, data: {
0521        likes: likes[0].cnt, collects: collects[0].cnt,
0522        adopts: adopts[0].cnt, flowers: flowers[0].cnt
0523      }});
0524    } catch (e) { res.json({ success: false, error: e.message }); }
0525  });
0526
0527  // 获取用户对某宠物的状态
0528  router.get('/pets/:code/user-status', async (req, res) => {
0529    try {
0530      const { code } = req.params;
0531      const { userId } = req.query;
0532      const safeUserId = parseInt(userId) || 0;
0533      const safeCode = validateInput(code, 50);
0534      if (!safeUserId) return res.json({ success: true, data: { liked: false, collected:
false, adopted: false } });
0535
0536      const [[liked], [collected], [adopted]] = await Promise.all([
0537        pool.execute('SELECT id FROM pet_likes WHERE user_id = ? AND pet_code = ?',
[safeUserId, safeCode]),
0538        pool.execute("SELECT id FROM pet_favorites WHERE student_id = ? AND pet_code = ? AND
action = 'favorite'", [safeUserId, safeCode]),
0539        pool.execute('SELECT id FROM pet_adopts WHERE user_id = ? AND pet_code = ?',
[safeUserId, safeCode])
0540      ]);
0541      res.json({ success: true, data: {
0542        liked: liked.length > 0, collected: collected.length > 0, adopted: adopted.length > 0
0543      }});
0544    } catch (e) { res.json({ success: false, error: e.message }); }
0545  });
0546
0547  // 获取用户领养的宠物列表（缓存 5 分钟）
0548  app.get('/apic/users/:userId/adopts', async (req, res) => {
0549    try {
0550      const { userId } = req.params;
```

```
0551      const safeUserId = parseInt(userId) || 0;
0552      if (!safeUserId) return res.json({ success: false, error: '无效用户 ID' });
0553
0554      const cacheKey = `user:${safeUserId}:adopts`;
0555      const data = await cachedGet(cacheKey, 300, async () => {
0556        const [rows] = await pool.execute('SELECT pet_code FROM pet_adopts WHERE user_id = ?',
[safeUserId]);
0557        return rows.map(r => r.pet_code);
0558      });
0559
0560      res.json({ success: true, data });
0561    } catch (e) { res.json({ success: false, error: e.message }); }
0562  });
0563
0564  // 获取用户收藏的宠物列表（缓存 5 分钟）
0565  router.get('/users/:userId/favorites', async (req, res) => {
0566    try {
0567      const { userId } = req.params;
0568      const safeUserId = parseInt(userId) || 0;
0569      if (!safeUserId) return res.json({ success: false, error: '无效用户 ID' });
0570
0571      const cacheKey = `user:${safeUserId}:favorites`;
0572      const data = await cachedGet(cacheKey, 300, async () => {
0573        const [rows] = await pool.execute("SELECT pet_code FROM pet_favorites WHERE
student_id = ? AND action = 'favorite'", [safeUserId]);
0574        return rows.map(r => r.pet_code);
0575      });
0576
0577      res.json({ success: true, data });
0578    } catch (e) { res.json({ success: false, error: e.message }); }
0579  });
0580
0581  // 获取用户关注的艺术家列表（缓存 5 分钟）
0582  router.get('/users/:userId/follows', async (req, res) => {
0583    try {
0584      const { userId } = req.params;
0585      const safeUserId = parseInt(userId) || 0;
0586      if (!safeUserId) return res.json({ success: false, error: '无效用户 ID' });
0587
0588      const cacheKey = `user:${safeUserId}:follows`;
0589      const data = await cachedGet(cacheKey, 300, async () => {
0590        const [rows] = await pool.execute('SELECT a.code FROM pet_artist_follow f JOIN
pet_artist a ON f.artist_id = a.id WHERE f.user_id = ?', [safeUserId]);
0591        return rows.map(r => r.code);
0592      });
0593
0594      res.json({ success: true, data });
0595    } catch (e) { res.json({ success: false, error: e.message }); }
0596  });
0597
0598  // 挂载 API 路由（同时支持两种模式）
0599  app.use('/apic', router);
0600  app.use('/cwtj/apic', router);
```

```
0601
0602  // 静态文件（同时支持两种模式）
0603  app.use('/images', express.static(path.join(__dirname, '..', 'public', 'images')));
0604  app.use('/cwtj/images', express.static(path.join(__dirname, '..', 'public', 'images')));
0605
0606  app.use(express.static(path.join(__dirname, '..', 'public')));
0607  app.use('/cwtj', express.static(path.join(__dirname, '..', 'public')));
0608
0609  // 首页（同时支持两种模式）
0610  app.get('/', (req, res) => {
0611    res.sendFile(path.join(__dirname, '..', 'public', 'index.html'));
0612  });
0613  app.get('/cwtj', (req, res) => {
0614    res.sendFile(path.join(__dirname, '..', 'public', 'index.html'));
0615  });
0616
0617  // 启动
0618  (async () => {
0619    await initRedis();
0620
0621    // 自动建表（新功能需要的表）
0622    const initConn = await mysql.createConnection(MYSQL_CONFIG);
0623
0624    // 创建艺术家表
0625    await initConn.execute(`CREATE TABLE IF NOT EXISTS pet_artist (
0626      id INT PRIMARY KEY AUTO_INCREMENT,
0627      code VARCHAR(50) UNIQUE NOT NULL,
0628      name VARCHAR(100) NOT NULL,
0629      title VARCHAR(100),
0630      description TEXT,
0631      avatar_url VARCHAR(255),
0632      follower_count INT DEFAULT 0,
0633      total_authorization INT DEFAULT 0,
0634      creator_level VARCHAR(10) DEFAULT 'B',
0635      status ENUM('active', 'inactive') DEFAULT 'active',
0636      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
0637      updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
0638    ) ENGINE=InnoDB CHARSET=utf8mb4`);
0639
0640    const newTables = [
0641      `CREATE TABLE IF NOT EXISTS pet_likes (
0642        id INT AUTO_INCREMENT PRIMARY KEY, user_id INT NOT NULL, pet_code VARCHAR(50) NOT
NULL,
0643        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
0644        UNIQUE KEY uk_user_pet (user_id, pet_code), INDEX idx_pet_code (pet_code)
0645      ) ENGINE=InnoDB CHARSET=utf8mb4`,
0646      `CREATE TABLE IF NOT EXISTS pet_adopts (
0647        id INT AUTO_INCREMENT PRIMARY KEY, user_id INT NOT NULL, pet_code VARCHAR(50) NOT
NULL,
0648        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
0649        UNIQUE KEY uk_user_pet (user_id, pet_code), INDEX idx_pet_code (pet_code)
0650      ) ENGINE=InnoDB CHARSET=utf8mb4`,
```

```
0651      `CREATE TABLE IF NOT EXISTS pet_flowers (
0652        id INT AUTO_INCREMENT PRIMARY KEY, user_id INT NOT NULL, pet_code VARCHAR(50) NOT
NULL,
0653        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, INDEX idx_pet_code (pet_code)
0654      ) ENGINE=InnoDB CHARSET=utf8mb4`,
0655      `CREATE TABLE IF NOT EXISTS pet_artist_follow (
0656        id INT AUTO_INCREMENT PRIMARY KEY, user_id INT NOT NULL, artist_id INT NOT NULL,
0657        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
0658        UNIQUE KEY uk_user_artist (user_id, artist_id), INDEX idx_artist_id (artist_id)
0659      ) ENGINE=InnoDB CHARSET=utf8mb4`
0660    ];
0661    for (const sql of newTables) {
0662      try { await initConn.execute(sql); } catch (e) {}
0663    }
0664    try { await initConn.execute('ALTER TABLE pet_favorites ADD INDEX idx_fav_pet_action
(pet_code, action)'); } catch (e) {}
0665    try { await initConn.execute('ALTER TABLE pet_favorites ADD INDEX idx_fav_user
(student_id)'); } catch (e) {}
0666
0667    // 同步艺术家数据
0668    for (const a of ARTISTS_DATA) {
0669      await initConn.execute(
0670        `INSERT INTO pet_artist (code, name, title, description, follower_count,
total_authorization, creator_level, status)
0671         VALUES (?, ?, ?, ?, ?, ?, ?, 'active')
0672         ON DUPLICATE KEY UPDATE name=VALUES(name), title=VALUES(title),
description=VALUES(description)`,
0673        [a.code, a.name, a.title, a.description, a.followerCount, a.totalAuthorization,
a.creatorLevel]
0674      );
0675    }
0676
0677    // 更新宠物-艺术家关联
0678    for (const [petCode, artistCode] of Object.entries(PET_ARTIST_MAP)) {
0679      await initConn.execute(
0680        `UPDATE pet_archive p, pet_artist a SET p.artistId = a.id WHERE p.code = ? AND a.code
= ?`,
0681        [petCode, artistCode]
0682      );
0683    }
0684
0685    await initConn.end();
0686    console.log('检查表结构完成 + 艺术家数据同步完成 + 宠物关联更新完成');
0687
0688    app.listen(APP_PORT, () => {
0689      console.log('\n========================================');
0690      console.log('  Pet Archive API Server 启动成功! ');
0691      console.log(`  http://localhost:${APP_PORT}`);
0692      console.log('  MySQL 连接池: 10 个连接');
0693      console.log('  Redis 缓存: 已启用');
0694      console.log('========================================\n');
0695    });
0696  })();
0697  /* ===== END OF server/server.js ===== */
0698  /* ===== FILE: server/redis-client.js (JavaScript) ===== */
0699  /**
0700   * Redis 缓存模块（版本号失效）
```

```
0701   * 写操作 INCR 版本号，读操作 key 含版本号，天然失效
0702   */
0703  const { createClient } = require('redis');
0704
0705  const REDIS_HOST = process.env.REDIS_HOST || '127.0.0.1';
0706  const REDIS_PORT = parseInt(process.env.REDIS_PORT) || 6378;
0707  const REDIS_PASSWORD = process.env.REDIS_PASSWORD || 'bjcwy';
0708  const REDIS_PREFIX = process.env.REDIS_PREFIX || 'bjwy:pet:';
0709
0710  let client = null;
0711
0712  function key(k) { return REDIS_PREFIX + k; }
0713
0714  async function initRedis() {
0715    try {
0716      client = createClient({
0717        socket: {
0718          host: REDIS_HOST,
0719          port: REDIS_PORT,
0720          reconnectStrategy: (retries) => { if (retries > 3) return false; return 2000; }
0721        },
0722        password: REDIS_PASSWORD || undefined
0723      });
0724      client.on('error', (err) => {});
0725      await client.connect();
0726      await client.ping();
0727      console.log('Redis 连接成功');
0728      return true;
0729    } catch (e) {
0730      console.log('Redis 未连接，跳过缓存');
0731      client = null;
0732      return false;
0733    }
0734  }
0735
0736  // 读缓存：自动拼接版本号
0737  async function cachedGet(cacheKey, ttlSeconds, fetchFn) {
0738    if (!client) return fetchFn();
0739    try {
0740      const ver = await client.get(key(`ver:${cacheKey}`)) || '0';
0741      const cacheWithVer = `${cacheKey}:v${ver}`;
0742      const cached = await client.get(key(cacheWithVer));
0743      if (cached) return JSON.parse(cached);
0744      const data = await fetchFn();
0745      if (data !== null && data !== undefined) {
0746        await client.setEx(key(cacheWithVer), ttlSeconds, JSON.stringify(data));
0747      }
0748      return data;
0749    } catch (e) {
0750      return fetchFn();
```

```
0751    }
0752  }
0753
0754  // 写操作后失效：INCR 版本号
0755  async function bumpVer(cacheKey) {
0756    if (!client) return;
0757    try { await client.incr(key(`ver:${cacheKey}`)); } catch (e) {}
0758  }
0759
0760  // 删除指定缓存 key
0761  async function delCache(cacheKey) {
0762    if (!client) return;
0763    try { await client.del(key(cacheKey)); } catch (e) {}
0764  }
0765
0766  module.exports = { initRedis, cachedGet, bumpVer, delCache };
0767  /* ===== END OF server/redis-client.js ===== */
0768  /* ===== FILE: scripts/init-tables.js (JavaScript) ===== */
0769  #!/usr/bin/env node
0770  /**
0771   * init-tables.js - 初始化数据库表（不删除现有数据）
0772   * 只创建 pet-archive 相关的新表
0773   */
0774
0775  const mysql = require('mysql2/promise');
0776
0777  const MYSQL_CONFIG = {
0778    host: '8.140.244.180',
0779    user: 'root',
0780    password: '4a926b9fed6c0d25',
0781    database: 'bjcwy_pc',
0782  };
0783
0784  const TABLES = [
0785    `CREATE TABLE IF NOT EXISTS pet_archive (
0786      id INT PRIMARY KEY AUTO_INCREMENT,
0787      code VARCHAR(50) UNIQUE NOT NULL,
0788      namespace VARCHAR(20) DEFAULT 'pet',
0789      name VARCHAR(100) NOT NULL,
0790      description TEXT,
0791      category VARCHAR(50),
0792      category_tab VARCHAR(50),
0793      tags TEXT,
0794      effort_image TEXT,
0795      artist_id INT DEFAULT NULL,
0796      pet_talk VARCHAR(200) DEFAULT NULL,
0797      pet_habit TEXT,
0798      rarity VARCHAR(20) DEFAULT 'normal',
0799      scene_type VARCHAR(50) DEFAULT 'class',
0800      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
0801      updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
0802      INDEX idx_category (category),
0803      INDEX idx_namespace (namespace),
0804      INDEX idx_artist_id (artist_id)
0805    ) ENGINE=InnoDB CHARSET=utf8mb4`,
0806
0807    `CREATE TABLE IF NOT EXISTS pet_archive_stage (
0808      id INT PRIMARY KEY AUTO_INCREMENT,
0809      pet_archive_id INT NOT NULL,
0810      level INT DEFAULT 1,
0811      image_url TEXT,
0812      style_name VARCHAR(50),
0813      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
0814      UNIQUE KEY idx_unique_stage (pet_archive_id, level, style_name),
0815      INDEX idx_pet_archive_id (pet_archive_id),
0816      FOREIGN KEY (pet_archive_id) REFERENCES pet_archive(id) ON DELETE CASCADE
0817    ) ENGINE=InnoDB CHARSET=utf8mb4`,
0818
0819    `CREATE TABLE IF NOT EXISTS pet_artist (
0820      id INT PRIMARY KEY AUTO_INCREMENT,
0821      code VARCHAR(50) UNIQUE NOT NULL,
0822      name VARCHAR(100) NOT NULL,
0823      title VARCHAR(100),
0824      description TEXT,
0825      avatar_url TEXT,
0826      follower_count INT DEFAULT 0,
0827      total_authorization INT DEFAULT 0,
0828      creator_level VARCHAR(20) DEFAULT 'C',
0829      status VARCHAR(20) DEFAULT 'active',
0830      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
0831      updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
0832      INDEX idx_status (status),
0833      INDEX idx_creator_level (creator_level)
0834    ) ENGINE=InnoDB CHARSET=utf8mb4`,
0835
0836    `CREATE TABLE IF NOT EXISTS pet_artist_follow (
0837      id INT PRIMARY KEY AUTO_INCREMENT,
0838      user_id INT NOT NULL,
0839      artist_id INT NOT NULL,
0840      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
0841      UNIQUE KEY idx_unique_follow (user_id, artist_id),
0842      INDEX idx_follow_artist_id (artist_id),
0843      FOREIGN KEY (artist_id) REFERENCES pet_artist(id) ON DELETE CASCADE
0844    ) ENGINE=InnoDB CHARSET=utf8mb4`,
0845
0846    `CREATE TABLE IF NOT EXISTS pet_rank (
0847      id INT PRIMARY KEY AUTO_INCREMENT,
0848      rank_type VARCHAR(20) NOT NULL,
0849      artist_id INT NOT NULL,
0850      rank_value INT DEFAULT 0,
```

```
    rank_num INT DEFAULT 0,
    period VARCHAR(20) DEFAULT 'week',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    UNIQUE KEY idx_unique_rank (rank_type, artist_id, period),
    INDEX idx_rank_type (rank_type),
    INDEX idx_rank_period (period),
    FOREIGN KEY (artist_id) REFERENCES pet_artist(id) ON DELETE CASCADE
  ) ENGINE=InnoDB CHARSET=utf8mb4`,

  `CREATE TABLE IF NOT EXISTS pet_archive_sync_log (
    id INT PRIMARY KEY AUTO_INCREMENT,
    source_url TEXT NOT NULL,
    pet_count INT DEFAULT 0,
    image_count INT DEFAULT 0,
    status VARCHAR(20) DEFAULT 'pending',
    error_msg TEXT,
    started_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    finished_at TIMESTAMP NULL,
    INDEX idx_sync_status (status)
  ) ENGINE=InnoDB CHARSET=utf8mb4`,
];

async function initTables() {
  const conn = await mysql.createConnection(MYSQL_CONFIG);

  for (const sql of TABLES) {
    try {
      await conn.execute(sql);
      console.log('✓ 表创建成功');
    } catch (e) {
      if (e.code === 'ER_TABLE_EXISTS_ERROR') {
        console.log('⚠ 表已存在');
      } else {
        console.error('✗ 创建失败:', e.message);
      }
    }
  }

  await conn.end();
  console.log('\n🎉 数据库初始化完成');
}

initTables().catch(e => {
  console.error('Error:', e.message);
  process.exit(1);
});
/* ===== END OF scripts/init-tables.js ===== */
/* ===== FILE: scripts/sync-data.js (JavaScript) ===== */
#!/usr/bin/env node
```

```
0901  /**
0902   * sync-data.js - 一次性同步线上宠物数据到本地 MySQL
0903   *
0904   * 功能：
0905   * 1. 拉取线上 API 所有宠物数据
0906   * 2. 存储到 pet_archive 表
0907   * 3. 提取阶段图片信息存储到 pet_archive_stage 表
0908   * 4. 记录同步日志
0909   */
0910
0911  const axios = require('axios');
0912  const mysql = require('mysql2/promise');
0913
0914  const API_URL = process.env.PET_API_URL || 'https://llm-
api.ourschool.cc/workflows/resources/pet';
0915
0916  const MYSQL_CONFIG = {
0917    host: '8.140.244.180',
0918    user: 'root',
0919    password: '4a926b9fed6c0d25',
0920    database: 'bjcwy_pc',
0921  };
0922
0923  async function fetchPets() {
0924    console.log('📡 拉取线上宠物数据...');
0925    const res = await axios.get(API_URL, { timeout: 60000 });
0926    if (!Array.isArray(res.data)) {
0927      throw new Error('Invalid API response');
0928    }
0929    console.log(`✅ 获取到 ${res.data.length} 只宠物`);
0930    return res.data;
0931  }
0932
0933  async function syncPets(pets) {
0934    const conn = await mysql.createConnection(MYSQL_CONFIG);
0935
0936    let petCount = 0;
0937    let imageCount = 0;
0938
0939    for (const pet of pets) {
0940      // 分类映射：英文 -> 中文 Tab
0941      const categoryMap = {
0942        'dog': '萌犬天团',
0943        'cat': '软萌喵星',
0944        'wild': '绿野部落',
0945        'guofeng': '国风神兽',
0946        'shanhai': '山海灵宠',
0947        'water': '水中伙伴',
0948        'zodiac': '生肖萌宝'
0949      };
0950      const rawCategory = pet.data?.category || '';
```

```
0951      const categoryTab = categoryMap[rawCategory] || '';  // 中文 Tab
0952      const category = rawCategory;  // 英文分类存 code
0953
0954      // 1. 插入/更新宠物基本信息
0955      const tags = JSON.stringify(pet.data?.tags || []);
0956
0957      await conn.execute(
0958        `INSERT INTO pet_archive (code, namespace, name, description, category, categoryTab,
tags, effortImage)
0959         VALUES (?, ?, ?, ?, ?, ?, ?, ?)
0960         ON DUPLICATE KEY UPDATE
0961           name = VALUES(name),
0962           description = VALUES(description),
0963           category = VALUES(category),
0964           categoryTab = VALUES(categoryTab),
0965           tags = VALUES(tags),
0966           effortImage = VALUES(effortImage),
0967           updatedAt = NOW()`,
0968        [
0969          pet.code,
0970          pet.namespace || 'pet',
0971          pet.name,
0972          pet.description || '',
0973          category,
0974          categoryTab,
0975          tags,
0976          pet.data?.effortImage || ''
0977        ]
0978      );
0979
0980      // 获取宠物 ID
0981      const [rows] = await conn.execute(
0982        'SELECT id FROM pet_archive WHERE code = ?',
0983        [pet.code]
0984      );
0985      const petId = rows[0].id;
0986      petCount++;
0987
0988      // 2. 处理进化阶段图片（默认 stages）
0989      if (pet.data?.stages) {
0990        for (const stage of pet.data.stages) {
0991          await conn.execute(
0992            `INSERT INTO pet_archive_stage (petArchiveId, level, imageUrl, styleName)
0993             VALUES (?, ?, ?, ?)
0994             ON DUPLICATE KEY UPDATE imageUrl = VALUES(imageUrl)`,
0995            [petId, stage.level, stage.image, null]
0996          );
0997          imageCount++;
0998        }
0999      }
1000
```

```
    // 3. 处理风格阶段图片（styleStages）
    if (pet.data?.styleStages) {
      for (const [styleName, stages] of Object.entries(pet.data.styleStages)) {
        for (const stage of stages) {
          await conn.execute(
            `INSERT INTO pet_archive_stage (petArchiveId, level, imageUrl, styleName)
             VALUES (?, ?, ?, ?)
             ON DUPLICATE KEY UPDATE imageUrl = VALUES(imageUrl)`,
            [petId, stage.level, stage.image, styleName]
          );
          imageCount++;
        }
      }
    }

    // 进度输出
    if (petCount % 10 === 0) {
      console.log(`  已处理 ${petCount}/${pets.length} 只宠物`);
    }
  }

  await conn.end();
  return { petCount, imageCount };
}

async function main() {
  console.log('\n========== 宠物档案同步 ==========\n');

  // 记录同步开始
  const conn = await mysql.createConnection(MYSQL_CONFIG);
  const [result] = await conn.execute(
    `INSERT INTO pet_archive_sync_log (sourceUrl, status) VALUES (?, 'running')`,
    [API_URL]
  );
  const syncId = result.insertId;
  await conn.end();

  try {
    // 拉取数据
    const pets = await fetchPets();

    // 同步到数据库
    const { petCount, imageCount } = await syncPets(pets);

    // 更新同步成功
    const conn2 = await mysql.createConnection(MYSQL_CONFIG);
    await conn2.execute(
      `UPDATE pet_archive_sync_log SET status = 'success', petCount = ?, imageCount = ?,
finishedAt = NOW() WHERE id = ?`,
      [petCount, imageCount, syncId]
    );
```

```
    await conn2.end();

    console.log(`\n✅ 同步完成: ${petCount} 只宠物, ${imageCount} 张图片`);
    console.log(`📝 同步记录 ID: ${syncId}`);

  } catch (e) {
    // 更新同步失败
    const conn2 = await mysql.createConnection(MYSQL_CONFIG);
    await conn2.execute(
      `UPDATE pet_archive_sync_log SET status = 'failed', errorMsg = ?, finishedAt = NOW() WHERE id = ?`,
      [e.message, syncId]
    );
    await conn2.end();

    console.error(`\n❌ 同步失败: ${e.message}`);
    process.exit(1);
  }
}

main();
/* ===== END OF scripts/sync-data.js ===== */
/* ===== FILE: scripts/sync-incremental.js (JavaScript) ===== */
#!/usr/bin/env node
/**
 * sync-incremental.sh - 增量同步脚本（Linux Shell 版）
 *
 * 用法:
 *   ./scripts/sync-incremental.sh        # 检测并增量同步
 *   ./scripts/sync-incremental.sh -f     # 强制全量同步
 *
 * crontab 配置:
 *   0 */6 * * * cd /path/to/pet-archive && ./scripts/sync-incremental.sh >> /var/log/pet-archive-sync.log 2>&1
 */

const axios = require('axios');
const mysql = require('mysql2/promise');
const fs = require('fs');
const path = require('path');
const { execSync } = require('child_process');

const API_URL = process.env.PET_API_URL || 'https://llm-api.ourschool.cc/workflows/resources/pet';

const MYSQL_CONFIG = {
  host: process.env.MYSQL_HOST || '8.140.244.180',
  user: process.env.MYSQL_USER || 'root',
  password: process.env.MYSQL_PASSWORD || '4a926b9fed6c0d25',
  database: process.env.MYSQL_DATABASE || 'bjcwy_pc',
};

const IMAGE_DIR = process.env.IMAGE_DIR || path.join(__dirname, '..', 'public', 'images');
```

```
1101
1102  async function fetchPets() {
1103    console.log(`📡 拉取线上数据: ${API_URL}`);
1104    const res = await axios.get(API_URL, { timeout: 60000 });
1105    return Array.isArray(res.data) ? res.data : [];
1106  }
1107
1108  async function getLocalPetCodes() {
1109    const conn = await mysql.createConnection(MYSQL_CONFIG);
1110    const [rows] = await conn.execute('SELECT code FROM pet_archive');
1111    await conn.end();
1112    return new Set(rows.map(r => r.code));
1113  }
1114
1115  function ensureDir(dir) {
1116    if (!fs.existsSync(dir)) {
1117      fs.mkdirSync(dir, { recursive: true });
1118    }
1119  }
1120
1121  async function downloadImage(url, filepath) {
1122    try {
1123      ensureDir(path.dirname(filepath));
1124      const res = await axios.get(url, { responseType: 'arraybuffer', timeout: 30000 });
1125      fs.writeFileSync(filepath, res.data);
1126      return true;
1127    } catch {
1128      return false;
1129    }
1130  }
1131
1132  async function syncIncremental(force = false) {
1133    const conn = await mysql.createConnection(MYSQL_CONFIG);
1134    const pets = await fetchPets();
1135    const localCodes = force ? new Set() : await getLocalPetCodes();
1136
1137    let newCount = 0;
1138    let updateCount = 0;
1139    let imageCount = 0;
1140
1141    for (const pet of pets) {
1142      const isNew = !localCodes.has(pet.code);
1143
1144      // 插入/更新宠物
1145      const tags = JSON.stringify(pet.data?.tags || []);
1146      await conn.execute(
1147        `INSERT INTO pet_archive (code, namespace, name, description, category, categoryTab,
tags, effortImage)
1148         VALUES (?, ?, ?, ?, ?, ?, ?, ?)
1149         ON DUPLICATE KEY UPDATE
1150           name = VALUES(name),
```

```
1151           description = VALUES(description),
1152           category = VALUES(category),
1153           categoryTab = VALUES(categoryTab),
1154           tags = VALUES(tags),
1155           effortImage = VALUES(effortImage),
1156           updatedAt = NOW()`,
1157         [pet.code, pet.namespace || 'pet', pet.name, pet.description || '', pet.category ||
'', pet.categoryTab || pet.category || '', tags, pet.data?.effortImage || '']
1158       );
1159
1160       if (isNew) newCount++;
1161       else updateCount++;
1162
1163       // 获取 ID 并处理图片
1164       const [rows] = await conn.execute('SELECT id FROM pet_archive WHERE code = ?',
[pet.code]);
1165       const petId = rows[0].id;
1166
1167       // 默认阶段图片
1168       if (pet.data?.stages) {
1169         for (const stage of pet.data.stages) {
1170           const filename = `${stage.level}.png`;
1171           const filepath = path.join(IMAGE_DIR, pet.code, filename);
1172           if (!fs.existsSync(filepath) && stage.image) {
1173             const ok = await downloadImage(stage.image, filepath);
1174             if (ok) imageCount++;
1175           }
1176           await conn.execute(
1177             `INSERT INTO pet_archive_stage (petArchiveId, level, imageUrl, styleName) VALUES
(?, ?, ?, NULL)
1178              ON DUPLICATE KEY UPDATE imageUrl = VALUES(imageUrl)`,
1179             [petId, stage.level, stage.image]
1180           );
1181         }
1182       }
1183
1184       // 风格图片
1185       if (pet.data?.styleStages) {
1186         for (const [styleName, stages] of Object.entries(pet.data.styleStages)) {
1187           for (const stage of stages) {
1188             const filename = `${stage.level}-${styleName}.png`;
1189             const filepath = path.join(IMAGE_DIR, pet.code, filename);
1190             if (!fs.existsSync(filepath) && stage.image) {
1191               const ok = await downloadImage(stage.image, filepath);
1192               if (ok) imageCount++;
1193             }
1194             await conn.execute(
1195               `INSERT INTO pet_archive_stage (petArchiveId, level, imageUrl, styleName)
VALUES (?, ?, ?, ?)
1196                ON DUPLICATE KEY UPDATE imageUrl = VALUES(imageUrl)`,
1197               [petId, stage.level, stage.image, styleName]
1198             );
1199           }
1200         }
```

```
    }
  }

  await conn.end();
  return { newCount, updateCount, imageCount };
}

async function main() {
  const force = process.argv.includes('-f');
  console.log(`\n========== 增量同步 (force=${force}) ==========\n`);

  try {
    const { newCount, updateCount, imageCount } = await syncIncremental(force);
    console.log(`\n✓ 同步完成: 新增${newCount}, 更新${updateCount}, 新图${imageCount}`);
  } catch (e) {
    console.error(`\n✗ 同步失败: ${e.message}`);
    process.exit(1);
  }
}

main();
/* ===== END OF scripts/sync-incremental.js ===== */
/* ===== FILE: scripts/download-images.js (JavaScript) ===== */
#!/usr/bin/env node
/**
 * download-images.js - 下载所有宠物图片到本地
 *
 * 功能:
 * 1. 从数据库获取所有图片 URL
 * 2. 下载并保存到 public/images/ 目录
 * 3. 结构: public/images/{code}/{level}-{style}.png
 */

const axios = require('axios');
const fs = require('fs');
const path = require('path');
const mysql = require('mysql2/promise');

const MYSQL_CONFIG = {
  host: '8.140.244.180',
  user: 'root',
  password: '4a926b9fed6c0d25',
  database: 'bjcwy_pc',
};

const IMAGE_DIR = path.join(__dirname, '..', 'public', 'images');

async function fetchImages() {
  const conn = await mysql.createConnection(MYSQL_CONFIG);
  const [rows] = await conn.execute(`
```

```
    SELECT p.code, s.level, s.imageUrl, s.styleName
    FROM pet_archive_stage s
    JOIN pet_archive p ON s.petArchiveId = p.id
    WHERE s.imageUrl IS NOT NULL AND s.imageUrl != ''
  `);
  await conn.end();
  return rows;
}

function ensureDir(dir) {
  if (!fs.existsSync(dir)) {
    fs.mkdirSync(dir, { recursive: true });
  }
}

async function downloadImage(url, filepath) {
  try {
    const res = await axios.get(url, { responseType: 'arraybuffer', timeout: 30000 });
    fs.writeFileSync(filepath, res.data);
    return true;
  } catch (e) {
    console.error(`  ✗ 下载失败: ${url} - ${e.message}`);
    return false;
  }
}

async function main() {
  console.log('\n========== 下载宠物图片 ==========\n');

  ensureDir(IMAGE_DIR);

  const images = await fetchImages();
  console.log(`📊 待下载: ${images.length} 张图片`);

  let successCount = 0;
  let failCount = 0;

  for (const img of images) {
    // 构建本地路径
    const petDir = path.join(IMAGE_DIR, img.code);
    ensureDir(petDir);

    const styleSuffix = img.styleName ? `-${img.styleName}` : '';
    const filename = `${img.level}${styleSuffix}.png`;
    const filepath = path.join(petDir, filename);

    // 如果已存在则跳过
    if (fs.existsSync(filepath)) {
      successCount++;
      continue;
```

```
    }

    // 下载
    const ok = await downloadImage(img.imageUrl, filepath);
    if (ok) {
      successCount++;
    } else {
      failCount++;
    }

    // 进度
    const total = successCount + failCount;
    if (total % 50 === 0) {
      console.log(`  📊 进度: ${total}/${images.length}`);
    }
  }

  console.log(`\n✅ 下载完成: ${successCount} 成功, ${failCount} 失败`);
}

main().catch(e => {
  console.error('Error:', e.message);
  process.exit(1);
});
/* ===== END OF scripts/download-images.js ===== */
/* ===== FILE: scripts/add-fields.js (JavaScript) ===== */
#!/usr/bin/env node
/**
 * add-fields.js - 为已存在的表添加新字段
 */

const mysql = require('mysql2/promise');

const MYSQL_CONFIG = {
  host: '8.140.244.180',
  user: 'root',
  password: '4a926b9fed6c0d25',
  database: 'bjcwy_pc',
};

const NEW_FIELDS = [
  { name: 'artistId', type: 'INT DEFAULT NULL', after: 'effortImage' },
  { name: 'petTalk', type: 'VARCHAR(200) DEFAULT NULL', after: 'artistId' },
  { name: 'petHabit', type: 'TEXT', after: 'petTalk' },
  { name: 'rarity', type: "VARCHAR(20) DEFAULT 'normal'", after: 'petHabit' },
  { name: 'sceneType', type: "VARCHAR(50) DEFAULT 'class'", after: 'rarity' },
];

async function addFields() {
  const conn = await mysql.createConnection(MYSQL_CONFIG);
```

```
1351
1352    const [columns] = await conn.execute(
1353      "SHOW COLUMNS FROM pet_archive"
1354    );
1355    const existingFields = columns.map(col => col.Field);
1356
1357    for (const field of NEW_FIELDS) {
1358      if (!existingFields.includes(field.name)) {
1359        try {
1360          const sql = `ALTER TABLE pet_archive ADD COLUMN ${field.name} ${field.type} AFTER
${field.after}`;
1361          await conn.execute(sql);
1362          console.log('✅ 字段添加成功:', field.name);
1363        } catch (e) {
1364          console.error('❌ 添加失败:', field.name, '-', e.message);
1365        }
1366      } else {
1367        console.log('⚠️ 字段已存在:', field.name);
1368      }
1369    }
1370
1371    const [indexes] = await conn.execute(
1372      "SHOW INDEX FROM pet_archive"
1373    );
1374    const existingIndexes = indexes.map(idx => idx.Key_name);
1375
1376    if (!existingIndexes.includes('idx_artist_id')) {
1377      try {
1378        await conn.execute('ALTER TABLE pet_archive ADD INDEX idx_artist_id (artistId)');
1379        console.log('✅ 索引添加成功: idx_artist_id');
1380      } catch (e) {
1381        console.error('❌ 索引添加失败:', e.message);
1382      }
1383    } else {
1384      console.log('⚠️ 索引已存在: idx_artist_id');
1385    }
1386
1387    await conn.end();
1388    console.log('\n🎉 字段添加完成');
1389  }
1390
1391  addFields().catch(e => {
1392    console.error('Error:', e.message);
1393    process.exit(1);
1394  });
1395  /* ===== END OF scripts/add-fields.js ===== */
1396  /* ===== FILE: server/migrate.sql (SQL) ===== */
1397  -- ========================================
1398  -- 萌宠星球 上线建表脚本
1399  -- 数据库: bjcwy_pc
1400  -- 执行方式: mysql -u root -p bjcwy_pc < server/migrate.sql
```

```
-- ============================================

-- 1. pet_likes 喜欢表
CREATE TABLE IF NOT EXISTS pet_likes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  pet_code VARCHAR(50) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY uk_user_pet (user_id, pet_code),
  INDEX idx_pet_code (pet_code)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 2. pet_adopts 领养表
CREATE TABLE IF NOT EXISTS pet_adopts (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  pet_code VARCHAR(50) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY uk_user_pet (user_id, pet_code),
  INDEX idx_pet_code (pet_code)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 3. pet_flowers 小红花表
CREATE TABLE IF NOT EXISTS pet_flowers (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  pet_code VARCHAR(50) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_pet_code (pet_code)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 4. pet_favorites 补充索引
ALTER TABLE pet_favorites ADD INDEX idx_fav_pet_action (pet_code, action);
/* ===== END OF server/migrate.sql ===== */
/* ===== FILE: sql/indexes.sql (SQL) ===== */
-- ================================================
-- pet-archive 上线索引脚本
-- 执行方式: mysql -h 8.140.244.180 -u root -p bjcwy_pc < sql/indexes.sql
-- ================================================

-- pet_artist_follow 表（关注表）
CREATE TABLE IF NOT EXISTS pet_artist_follow (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  artist_id INT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY uk_user_artist (user_id, artist_id),
  INDEX idx_artist_id (artist_id)
) ENGINE=InnoDB CHARSET=utf8mb4;

```

```
-- pet_favorites 表索引
ALTER TABLE pet_favorites ADD INDEX idx_fav_pet_action (pet_code, action);
ALTER TABLE pet_favorites ADD INDEX idx_fav_user (student_id);

-- pet_likes 表索引（如果表已存在但缺少索引）
-- ALTER TABLE pet_likes ADD UNIQUE KEY uk_user_pet (user_id, pet_code);
-- ALTER TABLE pet_likes ADD INDEX idx_pet_code (pet_code);

-- pet_adopts 表索引（如果表已存在但缺少索引）
-- ALTER TABLE pet_adopts ADD UNIQUE KEY uk_user_pet (user_id, pet_code);
-- ALTER TABLE pet_adopts ADD INDEX idx_pet_code (pet_code);

-- pet_flowers 表索引（如果表已存在但缺少索引）
-- ALTER TABLE pet_flowers ADD INDEX idx_pet_code (pet_code);

-- 验证索引
SHOW INDEX FROM pet_artist_follow;
SHOW INDEX FROM pet_favorites;
SHOW INDEX FROM pet_likes;
SHOW INDEX FROM pet_adopts;
SHOW INDEX FROM pet_flowers;
/* ===== END OF sql/indexes.sql ===== */
/* ===== FILE: ecosystem.config.js (JavaScript) ===== */
module.exports = {
  apps: [{
    name: "bctj",
    script: "server/server.js",
    instances: 1,
    max_memory_restart: "252M",
    env: {
      "NODE_ENV": "production",
      "APP_PORT": 3003,
      "MYSQL_HOST": "8.140.244.180",
      "MYSQL_USER": "root",
      "MYSQL_PASSWORD": "4a926b9fed6c0d25",
      "MYSQL_DATABASE": "bjcwy_pc",
      "REDIS_HOST": "127.0.0.1",
      "REDIS_PORT": "6378",
      "REDIS_PASSWORD": "bjcwy",
      "REDIS_PREFIX": "bjwy:pet:"
    },
    error_file: "logs/error.log",
    out_file: "logs/out.log",
    time: true
  }]
};
/* ===== END OF ecosystem.config.js ===== */
/* ===== FILE: public/index.html (HTML) ===== */
<!DOCTYPE html>
<html lang="zh-CN">
```

第二部分　源代码（后 30 页，每页 50 行）

```
2715
2716      document.getElementById('btn-like').addEventListener('click', async () => {
2717        const userId = localStorage.getItem('pet_user_id');
2718        if (!userId) { showLoginModal(); return; }
2719        const petCode = currentPetData?.code;
2720        try {
2721          const res = await fetch(`${API_BASE}/likes/toggle`, {
2722            method: 'POST', headers: { 'Content-Type': 'application/json' },
2723            body: JSON.stringify({ userId: parseInt(userId), petCode })
2724          });
2725          const json = await res.json();
2726          if (json.success) {
2727            const b = document.getElementById('btn-like');
2728            b.innerHTML = `<span>♡</span><span>${json.data.count}</span>`;
2729            b.style.background = json.data.liked ? '#ec4899' : '#fdf2f8';
2730            b.style.color = json.data.liked ? 'white' : '#ec4899';
2731            showToast(json.data.liked ? '♡ 喜欢成功！' : '已取消喜欢', json.data.liked ?
'success' : 'info');
2732          }
2733        } catch (e) {}
2734      });
2735
2736      document.getElementById('btn-collect').addEventListener('click', async () => {
2737        const userId = localStorage.getItem('pet_user_id');
2738        if (!userId) { showLoginModal(); return; }
2739        const petCode = currentPetData?.code;
2740        try {
2741          const res = await fetch(`${API_BASE}/favorites/toggle`, {
2742            method: 'POST', headers: { 'Content-Type': 'application/json' },
2743            body: JSON.stringify({ userId: parseInt(userId), petCode })
2744          });
2745          const json = await res.json();
2746          if (json.success) {
2747            const countRes = await fetch(`${API_BASE}/pets/${petCode}/stats`);
2748            const countJson = await countRes.json();
2749            const cnt = countJson.success ? countJson.data.collects : 0;
2750            const b = document.getElementById('btn-collect');
2751            b.innerHTML = `<span>♡</span><span>${cnt}</span>`;
2752            b.style.background = json.data.favorited ? '#ef4444' : 'var(--pet-primary-soft)';
2753            b.style.color = json.data.favorited ? 'white' : 'var(--pet-primary)';
2754            showToast(json.data.favorited ? '♡ 收藏成功！' : '已取消收藏', json.data.favorited ?
'success' : 'info');
2755          }
2756        } catch (e) {}
2757      });
2758
2759      document.getElementById('btn-adopt').addEventListener('click', async () => {
2760        const userId = localStorage.getItem('pet_user_id');
2761        if (!userId) { showLoginModal(); return; }
2762        const petCode = currentPetData?.code;
2763        const b = document.getElementById('btn-adopt');
2764
```

```
2765        // 已领养 → 点击进入我的宠物页面
2766        if (b.dataset.adopted === 'true') {
2767          window.location.href = '/cwtj/my-pets.html';
2768          return;
2769        }
2770
2771        try {
2772          const res = await fetch(`${API_BASE}/adopts/toggle`, {
2773            method: 'POST', headers: { 'Content-Type': 'application/json' },
2774            body: JSON.stringify({ userId: parseInt(userId), petCode })
2775          });
2776          const json = await res.json();
2777          if (json.success) {
2778            b.innerHTML = `<span>🏠</span><span>${json.data.count}</span>`;
2779            if (json.data.adopted) {
2780              b.style.background = '#22c55e'; b.style.color = 'white';
2781              b.dataset.adopted = 'true';
2782              showToast('🏠 领养成功！再点一次进入我的宠物~');
2783            } else {
2784              b.style.background = 'var(--pet-primary)'; b.style.color = 'white';
2785              b.dataset.adopted = 'false';
2786              showToast('已取消领养', 'info');
2787            }
2788          }
2789        } catch (e) {}
2790      });
2791
2792      document.getElementById('btn-flower').addEventListener('click', async () => {
2793        const userId = localStorage.getItem('pet_user_id');
2794        if (!userId) { showLoginModal(); return; }
2795        const petCode = currentPetData?.code;
2796        try {
2797          const res = await fetch(`${API_BASE}/flowers/send`, {
2798            method: 'POST', headers: { 'Content-Type': 'application/json' },
2799            body: JSON.stringify({ userId: parseInt(userId), petCode })
2800          });
2801          const json = await res.json();
2802          if (json.success) {
2803            document.getElementById('btn-flower').innerHTML =
`<span>✿</span><span>${json.data.count}</span>`;
2804            const messages = JSON.parse(localStorage.getItem('pet_messages') || '[]');
2805            messages.push({
2806              id: Date.now(), petCode, petName: currentPetData?.name,
2807              artistName: currentPetData?.artistName,
2808              message: getRandomMessage(currentPetData?.name),
2809              time: new Date().toLocaleString('zh-CN'), type: 'flower'
2810            });
2811            localStorage.setItem('pet_messages', JSON.stringify(messages));
2812            showToast('✿ 已发送小红花！期待作者更新~');
2813          }
2814        } catch (e) {}
```

```
2815      });
2816
2817      function getRandomMessage(petName) {
2818        const messages = [
2819          `求求更新 ${petName} 的新皮肤~ 拜托啦！`,
2820          `作者大大快更新 ${petName} 的新造型吧！期待 ing~`,
2821          `送你一朵小红花 🌷 希望 ${petName} 能有更多可爱皮肤！`,
2822          `好喜欢 ${petName} 呀！求更多换装~`,
2823          `🌸 期待 ${petName} 的新衣服！`
2824        ];
2825        return messages[Math.floor(Math.random() * messages.length)];
2826      }
2827
2828      document.getElementById('btn-follow-artist').addEventListener('click', async () => {
2829        const userId = localStorage.getItem('pet_user_id');
2830        if (!userId) {
2831          showLoginModal();
2832          return;
2833        }
2834
2835        const artistCode = currentPetData?.artistCode;
2836        if (!artistCode) {
2837          showToast('无法获取艺术家信息', 'error');
2838          return;
2839        }
2840
2841        let follows = JSON.parse(localStorage.getItem('pet_follows') || '[]');
2842        const followBtn = document.getElementById('btn-follow-artist');
2843
2844        if (follows.includes(artistCode)) {
2845          follows = follows.filter(f => f !== artistCode);
2846          followBtn.classList.remove('followed');
2847          followBtn.textContent = '关注';
2848          followBtn.style.background = 'var(--pet-primary-soft)';
2849          followBtn.style.color = 'var(--pet-primary)';
2850          showToast('已取消关注', 'info');
2851        } else {
2852          follows.push(artistCode);
2853          followBtn.classList.add('followed');
2854          followBtn.textContent = '已关注';
2855          followBtn.style.background = '#9ca3af';
2856          followBtn.style.color = 'white';
2857          showToast(`成功关注 ${currentPetData?.artistName || '艺术家'} 🎉`);
2858        }
2859
2860        localStorage.setItem('pet_follows', JSON.stringify(follows));
2861      });
2862
2863      document.getElementById('btn-close-login').addEventListener('click', hideLoginModal);
2864
```

```
    document.getElementById('btn-login').addEventListener('click', async () => {
      const username = document.getElementById('login-username').value;
      const password = document.getElementById('login-password').value;
      if (!username || !password) {
        showToast('请输入用户名和密码', 'error');
        return;
      }
      try {
        const res = await fetch(`${API_BASE}/users/login`, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ username, password })
        });
        const json = await res.json();
        if (json.success) {
          localStorage.setItem('pet_user_id', json.data.id);
          localStorage.setItem('pet_username', json.data.username);
          hideLoginModal();
          showToast('登录成功！欢迎回来~ 🎉');
          loadPetStatus();
          loadFollowStatus();
          loadStats();
        } else {
          showToast(json.error || '登录失败', 'error');
        }
      } catch (e) {
        showToast('网络错误，请重试', 'error');
      }
    });

    document.getElementById('btn-register').addEventListener('click', async () => {
      const username = document.getElementById('login-username').value;
      const password = document.getElementById('login-password').value;
      if (!username || !password) {
        showToast('请输入用户名和密码', 'error');
        return;
      }
      try {
        const res = await fetch(`${API_BASE}/users/register`, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ username, password })
        });
        const json = await res.json();
        if (json.success) {
          localStorage.setItem('pet_user_id', json.data.id);
          localStorage.setItem('pet_username', json.data.username);
          hideLoginModal();
          showToast('注册成功！欢迎加入~ 🎉');
          loadPetStatus();
```

```
            loadFollowStatus();
            loadStats();
          } else {
            showToast(json.error || '注册失败', 'error');
          }
        } catch (e) {
          showToast('网络错误，请重试', 'error');
        }
      });

      // ESC 关闭全屏
      document.addEventListener('keydown', (e) => {
        if (e.key === 'Escape') {
          hideFullscreenModal();
        }
      });

      // 获取 URL 参数并初始化
      const urlParams = new URLSearchParams(window.location.search);
      const petCode = urlParams.get('code');

      if (petCode) {
        loadPetDetail(petCode);
        loadPetList();
      }
    </script>

    <style>
      .stage-item {
        cursor: pointer;
        transition: all 0.2s;
        opacity: 0.6;
        filter: grayscale(100%);
      }
      .stage-item.active {
        opacity: 1;
        filter: none;
        transform: scale(1.05);
      }
      .stage-item img {
        border-radius: 12px;
        background: var(--pet-primary-soft);
      }
      .stage-item.active img {
        box-shadow: 0 4px 12px rgba(232, 122, 56, 0.3);
      }
      .style-tag {
        padding: 6px 14px;
        border-radius: 20px;
        border: 1px solid var(--pet-border);
```

```
      font-size: 12px;
      cursor: pointer;
      transition: all 0.2s;
      background: white;
    }
    .style-tag.active {
      background: var(--pet-primary);
      color: white;
      border-color: var(--pet-primary);
    }
    .section-card {
      transform-origin: top center;
    }
    #banner-section {
      position: sticky;
      top: 0;
      z-index: 10;
      padding-bottom: 60px;
    }
    #pet-image-container {
      min-height: 280px;
    }
    #pet-image {
      transition: transform 0.3s ease-out, opacity 0.3s ease-out;
    }
    #pet-tags {
      transition: opacity 0.3s ease-out, transform 0.3s ease-out;
    }
    #swipe-hint {
      padding: 8px 0;
    }
    #content-wrapper {
      position: relative;
      z-index: 5;
      margin-top: 0;
      padding-top: 30px;
    }
    #name-card {
      position: relative;
      z-index: 25;
      margin-top: 20px;
    }
    #fullscreen-modal {
      background: linear-gradient(135deg, #fff7ed 0%, #fef3c7 50%, #fde68a 100%);
    }
  </style>
  <!-- 百度统计 -->
  <script>
  var _hmt = _hmt || [];
  (function() {
```

```
      var hm = document.createElement("script");
      hm.src = "https://hm.baidu.com/hm.js?ca7a005fbc8241312a51eacc3e983489";
      var s = document.getElementsByTagName("script")[0];
      s.parentNode.insertBefore(hm, s);
    })();
    </script>
  </body>
  </html>
  /* ===== END OF public/pet-detail.html ===== */
  /* ===== FILE: public/artist.html (HTML) ===== */
  <!DOCTYPE html>
  <html lang="zh-CN">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="萌宠星球艺术家广场 - 汇聚顶尖萌宠画师，原创设计、风格多样，治愈系萌宠 IP 集结地">
    <meta name="keywords" content="艺术家广场,萌宠画师,原创宠物,宠物设计,IP 授权,萌宠星球">
    <meta property="og:title" content="艺术家广场 | 萌宠星球">
    <meta property="og:description" content="汇聚顶尖萌宠画师，原创设计治愈系萌宠">
    <meta property="og:type" content="website">
    <meta name="robots" content="index,follow">
    <link rel="canonical" href="/cwtj/artist.html">
    <title>艺术家广场 | 萌宠星球</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@unocss/reset/tailwind.min.css">
    <style>
      :root {
        --pet-primary: #e87a38;
        --pet-primary-soft: #fff5eb;
        --pet-bg: #faf6f1;
        --pet-bg-gradient: radial-gradient(ellipse 120% 80% at 50% -20%, #fef3e9 0%, #faf6f1 50%, #f3f0eb 100%);
        --pet-text: #0f172a;
        --pet-muted: #475569;
        --pet-border: #e2e8f0;
      }
      body { background: var(--pet-bg-gradient); background-color: var(--pet-bg); font-family: ui-sans-serif, system-ui, -apple-system, sans-serif; color: var(--pet-text); }
      .scrollbar-hide::-webkit-scrollbar { display: none; }
      .scrollbar-hide { -ms-overflow-style: none; scrollbar-width: none; }
      @keyframes fadeIn { from { opacity: 0; transform: translateY(10px); } to { opacity: 1; transform: translateY(0); } }
      .animate-fadeIn { animation: fadeIn 0.3s ease-out; }
      @keyframes slideUp { from { opacity: 0; transform: translateY(20px) scale(0.95); } to { opacity: 1; transform: translateY(0) scale(1); } }
      @keyframes fadeOut { from { opacity: 1; transform: translateY(0); } to { opacity: 0; transform: translateY(-10px); } }
      @keyframes bounce { 0%, 100% { transform: translateX(-50%) translateY(0); } 50% { transform: translateX(-50%) translateY(-8px); } }
    </style>
  </head>
  <body class="min-h-screen py-6 px-4 md:py-10 md:px-8">
    <div class="max-w-7xl mx-auto w-full">
      <!-- Header -->
      <div class="sticky-header sticky top-0 z-11 -mx-4 px-4 md:mx-0 md:px-0 pt-6 pb-4 md:pt-8 md:pb-6 bg-[var(--pet-bg)]">
        <header class="text-center mb-6 md:mb-10">
```

```
3065          <h1 class="text-2xl sm:text-3xl md:text-4xl font-extrabold mb-2 md:mb-3">艺术家广场
</h1>
3066          <p class="text-[var(--pet-muted)] text-xs sm:text-sm max-w-md mx-auto">探索创作者的精
彩世界，发现更多优质作品！</p>
3067        </header>
3068
3069        <div class="flex flex-col md:flex-row md:items-center gap-3 md:gap-4">
3070          <div class="relative w-full md:w-auto">
3071            <input id="artist-search-box" type="search" placeholder="搜索艺术家"
autocomplete="new-password" class="w-full md:w-64 h-11 pl-4 pr-10 rounded-full bg-white/90
backdrop-blur-sm border border-[var(--pet-border)] text-sm outline-none focus:border-orange-300
focus:ring-2 focus:ring-orange-100">
3072            <button id="btn-clear-search" class="absolute right-2 top-1/2 -translate-y-1/2 w-
7 h-7 rounded-full text-[var(--pet-muted)] hover:bg-slate-100">×</button>
3073          </div>
3074          <a href="/cwtj/" class="shrink-0 px-4 py-2.5 bg-white text-[var(--pet-primary)]
text-sm font-medium rounded-full border border-[var(--pet-border)] hover:bg-[var(--pet-primary-
soft)] transition-colors whitespace-nowrap">
3075            🐾 返回图鉴
3076          </a>
3077          <a href="/cwtj/ranking.html" class="shrink-0 px-4 py-2.5 bg-[var(--pet-primary)]
text-white text-sm font-medium rounded-full hover:opacity-90 transition-opacity whitespace-
nowrap">
3078            🏆 排行榜
3079          </a>
3080        </div>
3081      </div>
3082
3083      <!-- Stats -->
3084      <div class="grid grid-cols-3 gap-4 mb-8 max-w-lg mx-auto">
3085        <div class="text-center p-4 bg-white/80 rounded-2xl backdrop-blur-sm">
3086          <div id="artist-count" class="text-2xl md:text-3xl font-bold text-[var(--pet-
primary)]">--</div>
3087          <div class="text-xs text-[var(--pet-muted)] mt-1">艺术家</div>
3088        </div>
3089        <div class="text-center p-4 bg-white/80 rounded-2xl backdrop-blur-sm">
3090          <div id="pet-count" class="text-2xl md:text-3xl font-bold text-[var(--pet-
primary)]">--</div>
3091          <div class="text-xs text-[var(--pet-muted)] mt-1">作品数</div>
3092        </div>
3093        <div class="text-center p-4 bg-white/80 rounded-2xl backdrop-blur-sm">
3094          <div id="auth-count" class="text-2xl md:text-3xl font-bold text-[var(--pet-
primary)]">--</div>
3095          <div class="text-xs text-[var(--pet-muted)] mt-1">总授权</div>
3096        </div>
3097      </div>
3098
3099      <!-- Artists Grid -->
3100      <div id="artists-grid" class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4
md:gap-6">
3101        <div class="loading text-center py-20 text-[var(--pet-muted)]">加载中...</div>
3102      </div>
3103
3104      <!-- Artist Detail Modal -->
3105      <div id="artist-modal" class="fixed inset-0 bg-black/50 z-50 flex items-center justify-
center p-4 backdrop-blur-sm" style="display:none">
3106        <div class="bg-white/95 backdrop-blur-sm rounded-3xl w-full max-w-2xl max-h-[90vh]
overflow-y-auto shadow-2xl animate-fadeIn">
3107          <div class="relative">
3108            <div class="h-32 md:h-40 bg-gradient-to-r from-orange-100 via-amber-100 to-
yellow-100 rounded-t-3xl"></div>
3109            <button id="btn-close-modal" class="absolute top-4 right-4 w-10 h-10 rounded-full
bg-white shadow-md flex items-center justify-center hover:opacity-80 transition-opacity">
3110              <svg class="w-5 h-5 text-[var(--pet-primary)]" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2"><path d="M18 6L6 18M6 6l12 12"/></svg>
3111            </button>
3112            <div class="absolute -bottom-12 left-6 md:left-10">
```

```
3113              <img id="modal-avatar" src="" alt="" class="w-20 h-20 md:w-24 md:h-24 rounded-
full border-4 border-white shadow-lg object-cover">
3114            </div>
```

```
3115          </div>
3116          <div class="p-6 md:p-10 pt-16">
3117            <div class="flex items-start justify-between mb-4">
3118              <div>
3119                <h2 id="modal-name" class="text-2xl md:text-3xl font-bold mb-1"></h2>
3120                <span id="modal-title" class="text-sm text-[var(--pet-muted)] bg-[var(--pet-
primary-soft)] px-3 py-1 rounded-full"></span>
3121              </div>
3122              <button id="btn-follow" class="px-4 py-2 bg-[var(--pet-primary)] text-white
rounded-full text-sm font-medium hover:opacity-90 transition-opacity flex items-center gap-2">
3123                <svg class="w-4 h-4" viewBox="0 0 24 24" fill="none" stroke="currentColor"
stroke-width="2"><path d="M20.84 4.61a5.5 5.5 0 0 0-7.78 0L12 5.67l-1.06-1.06a5.5 5.5 0 0 0-7.78
7.78l1.06 1.06L12 21.23l7.78-7.78 1.06-1.06a5.5 5.5 0 0 0 0-7.78z"/></svg>
3124                关注
3125              </button>
3126            </div>
3127            <p id="modal-desc" class="text-[var(--pet-muted)] text-sm mb-6 line-clamp-3"></p>
3128            <div class="grid grid-cols-3 gap-4 mb-6">
3129              <div class="text-center p-3 bg-[var(--pet-bg)] rounded-xl">
3130                <div id="modal-pet-count" class="text-lg font-bold text-[var(--pet-
primary)]">0</div>
3131                <div class="text-xs text-[var(--pet-muted)]">作品</div>
3132              </div>
3133              <div class="text-center p-3 bg-[var(--pet-bg)] rounded-xl">
3134                <div id="modal-follower-count" class="text-lg font-bold text-[var(--pet-
primary)]">0</div>
3135                <div class="text-xs text-[var(--pet-muted)]">粉丝</div>
3136              </div>
3137              <div class="text-center p-3 bg-[var(--pet-bg)] rounded-xl">
3138                <div id="modal-auth-count" class="text-lg font-bold text-[var(--pet-
primary)]">0</div>
3139                <div class="text-xs text-[var(--pet-muted)]">授权</div>
3140              </div>
3141            </div>
3142            <div>
3143              <h3 class="font-bold mb-3 flex items-center gap-2">
3144                <div class="w-2 h-4 bg-[var(--pet-primary)] rounded-full"></div>
3145                代表作品
3146              </h3>
3147              <div id="modal-works" class="grid grid-cols-2 sm:grid-cols-3 gap-3">
3148                <div class="loading py-8 text-center text-[var(--pet-muted)]">加载中...</div>
3149              </div>
3150            </div>
3151          </div>
3152        </div>
3153      </div>
3154
3155      <!-- Login Modal -->
3156      <div id="login-modal" class="fixed inset-0 bg-black/50 z-50 flex items-center justify-
center p-4" style="display:none">
3157        <div class="bg-white rounded-2xl p-6 w-full max-w-sm animate-fadeIn">
3158          <h3 class="text-xl font-bold text-center mb-4">请先登录</h3>
3159          <div class="space-y-3">
3160            <input type="text" id="login-username" placeholder="用户名" class="w-full px-4 py-
3 border border-[var(--pet-border)] rounded-xl outline-none focus:border-[var(--pet-primary)]">
3161            <input type="password" id="login-password" placeholder="密码" class="w-full px-4
py-3 border border-[var(--pet-border)] rounded-xl outline-none focus:border-[var(--pet-
primary)]">
3162          </div>
3163          <button id="btn-login" class="w-full mt-4 py-3 bg-[var(--pet-primary)] text-white
font-bold rounded-xl hover:opacity-90">登录</button>
3164          <button id="btn-register" class="w-full mt-2 py-3 border border-[var(--pet-primary)]
text-[var(--pet-primary)] font-bold rounded-xl">注册</button>
```

```
3165          <button id="btn-close-login" class="w-full mt-2 py-2 text-[var(--pet-muted)] text-
sm">取消</button>
3166        </div>
3167      </div>
3168
3169      <!-- 萌化 Toast 提示组件 -->
3170      <div id="toast" class="fixed bottom-8 left-1/2 -translate-x-1/2 z-50"
style="display:none">
3171        <div class="relative">
3172          <img id="toast-decor-img" src="" class="absolute -top-10 left-1/2 -translate-x-1/2
w-12 h-12 object-contain" style="filter: drop-shadow(0 4px 6px rgba(0,0,0,0.1)); animation:
bounce 0.6s ease-in-out infinite;">
3173          <div id="toast-bubble" class="px-6 py-4 rounded-2xl shadow-xl flex items-center
gap-3 min-w-[180px]">
3174            <span id="toast-icon" class="text-2xl">✨</span>
3175            <span id="toast-text" class="text-sm font-medium"></span>
3176          </div>
3177        </div>
3178      </div>
3179    </div>
3180
3181    <script>
3182      // 动态获取 API 基础路径（支持 / 和 /cwtj/ 两种模式）
3183      const isSubPath = window.location.pathname.startsWith('/cwtj');
3184      const API_BASE = window.location.origin + (isSubPath ? '/cwtj/apic' : '/apic');
3185      let allArtists = [];
3186      let searchKeyword = '';
3187
3188      function showToast(message, type = 'success') {
3189        const toast = document.getElementById('toast');
3190        const bubble = document.getElementById('toast-bubble');
3191        const icon = document.getElementById('toast-icon');
3192        const text = document.getElementById('toast-text');
3193        const decorImg = document.getElementById('toast-decor-img');
3194
3195        text.textContent = message;
3196
3197        if (currentArtistCode) {
3198          decorImg.src = `/images/avatars/${currentArtistCode}.png`;
3199          decorImg.style.display = 'block';
3200          decorImg.onerror = function() {
3201            this.src = `data:image/svg+xml,%3Csvg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 48 48"%3E%3Ccircle cx="24" cy="24" r="20" fill="%23e87a38"%3E%3C/circle%3E%3Ctext
x="24" y="30" text-anchor="middle" fill="white" font-size="20" font-
weight="bold"%3E${currentArtistCode.charAt(0).toUpperCase()}%3C/text%3E%3C/svg%3E`;
3202          };
3203        } else {
3204          decorImg.src = 'data:image/svg+xml,%3Csvg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 24 24" fill="%23e87a38"%3E%3Cpath d="M12 21.35l-1.45-1.32C5.4 15.36 2 12.28 2 8.5 2
5.42 4.42 3 7.5 3c1.74 0 3.41.81 4.5 2.09C13.09 3.81 14.76 3 16.5 3 19.58 3 22 5.42 22 8.5c0
3.78-3.4 6.86-8.55 11.54L12 21.35z"/%3E%3C/svg%3E';
3205          decorImg.style.display = 'block';
3206        }
3207
3208        if (type === 'success') {
3209          icon.textContent = '✨';
3210          bubble.style.background = 'linear-gradient(135deg, #fff5eb 0%, #ffffff 100%)';
3211          bubble.style.border = '2px solid #e87a38';
3212          bubble.style.borderLeft = '4px solid #22c55e';
3213        } else if (type === 'error') {
3214          icon.textContent = '🙀';
```

```
        bubble.style.background = 'linear-gradient(135deg, #fef2f2 0%, #ffffff 100%)';
        bubble.style.border = '2px solid #ef4444';
        bubble.style.borderLeft = '4px solid #ef4444';
      } else if (type === 'info') {
        icon.textContent = '💡';
        bubble.style.background = 'linear-gradient(135deg, #eff6ff 0%, #ffffff 100%)';
        bubble.style.border = '2px solid #3b82f6';
        bubble.style.borderLeft = '4px solid #3b82f6';
      }

      toast.style.display = 'block';
      toast.style.animation = 'slideUp 0.4s cubic-bezier(0.68, -0.55, 0.265, 1.55)';

      setTimeout(() => {
        toast.style.animation = 'fadeOut 0.3s ease-out forwards';
        setTimeout(() => {
          toast.style.display = 'none';
        }, 300);
      }, 2500);
    }

    async function fetchArtists() {
      try {
        const res = await fetch(`${API_BASE}/artists`);
        const json = await res.json();
        if (json.success) {
          allArtists = json.data;
          renderArtists();
          updateStats();
        }
      } catch (e) {
        console.error('加载失败:', e);
      }
    }

    function updateStats() {
      document.getElementById('artist-count').textContent = allArtists.length || '--';

      const totalPets = allArtists.reduce((sum, a) => sum + (a.petCount || 0), 0);
      document.getElementById('pet-count').textContent = totalPets || '--';

      const totalAuth = allArtists.reduce((sum, a) => sum + (a.totalAuthorization || 0), 0);
      document.getElementById('auth-count').textContent = formatNumber(totalAuth) || '--';
    }

    function formatNumber(num) {
      if (num >= 10000) return (num / 10000).toFixed(1) + '万';
      return num.toString();
    }

```

```
3265      function renderArtists() {
3266        const filtered = allArtists.filter(a => {
3267          if (!searchKeyword) return true;
3268          return a.name.includes(searchKeyword) || a.title.includes(searchKeyword);
3269        });
3270
3271        if (filtered.length === 0) {
3272          document.getElementById('artists-grid').innerHTML = '<div class="col-span-full
text-center py-20 text-[var(--pet-muted)]">没有找到匹配的艺术家</div>';
3273          return;
3274        }
3275
3276        document.getElementById('artists-grid').innerHTML = filtered.map(artist => `
3277          <div class="artist-card bg-white/90 backdrop-blur-sm rounded-2xl overflow-hidden
shadow-md hover:shadow-lg transition-shadow cursor-pointer animate-fadeIn"
onclick="showArtistDetail('${artist.code}')">
3278            <div class="relative h-32 bg-gradient-to-br from-orange-50 via-amber-50 to-
yellow-50 flex items-center justify-center">
3279              <img src="/cwtj/images/avatars/${artist.code}.png" alt="${artist.name}"
class="w-16 h-16 rounded-full object-cover border-4 border-white shadow-md"
onerror="this.src='data:image/svg+xml,%3Csvg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220
0 24 24%22 fill=%22%23e2e8f0%22%3E%3Cpath d=%22M12 12c2.21 0 4-1.79 4-4s-1.79-4-4-4-4 1.79-4 4
1.79 4 4 4zm0 2c-2.67 0-8 1.34-8 4v2h16v-2c0-2.66-5.33-4-8-4z%22/%3E%3C/svg%3E'">
3280            </div>
3281            <div class="p-4">
3282              <div class="flex items-center gap-2 mb-1">
3283                <h3 class="font-bold">${artist.name}</h3>
3284                <span class="text-xs text-[var(--pet-muted)] bg-[var(--pet-primary-soft)] px-
2 py-0.5 rounded-full">${artist.title}</span>
3285              </div>
3286              <p class="text-xs text-[var(--pet-muted)] line-clamp-2 mb-
3">${artist.description}</p>
3287              <div class="flex items-center justify-between text-xs text-[var(--pet-muted)]">
3288                <span>🎨 ${artist.petCount || 0} 作品</span>
3289                <span>👥 ${artist.followerCount || 0} 粉丝</span>
3290              </div>
3291            </div>
3292          </div>
3293        `).join('');
3294      }
3295
3296      let currentArtistCode = null;
3297
3298      async function showArtistDetail(code) {
3299        try {
3300          currentArtistCode = code;
3301          const res = await fetch(`${API_BASE}/artists/${code}`);
3302          const json = await res.json();
3303          if (!json.success) return;
3304
3305          const artist = json.data;
3306          document.getElementById('modal-name').textContent = artist.name;
3307          document.getElementById('modal-title').textContent = artist.title;
3308          document.getElementById('modal-desc').textContent = artist.description;
3309          document.getElementById('modal-avatar').src = `/images/avatars/${artist.code}.png`;
3310          document.getElementById('modal-avatar').onerror = function() {
3311            this.src = 'data:image/svg+xml,%3Csvg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 24 24" fill="%23e2e8f0"%3E%3Cpath d="M12 12c2.21 0 4-1.79 4-4s-1.79-4-4-4-4 1.79-4 4
1.79 4 4 4zm0 2c-2.67 0-8 1.34-8 4v2h16v-2c0-2.66-5.33-4-8-4z"/%3E%3C/svg%3E';
3312          };
3313          document.getElementById('modal-pet-count').textContent = artist.pets?.length || 0;
3314          document.getElementById('modal-follower-count').textContent = artist.followerCount
|| 0;
```

```
3315          document.getElementById('modal-auth-count').textContent =
formatNumber(artist.totalAuthorization || 0);
3316
3317          const userId = localStorage.getItem('pet_user_id');
3318          if (userId) {
3319            const followRes = await fetch(`${API_BASE}/follows/list?userId=${userId}`);
3320            const followJson = await followRes.json();
3321            if (followJson.success && followJson.data.includes(code)) {
3322              updateFollowButton(true);
3323            } else {
3324              updateFollowButton(false);
3325            }
3326          } else {
3327            updateFollowButton(false);
3328          }
3329
3330          if (artist.pets && artist.pets.length > 0) {
3331            document.getElementById('modal-works').innerHTML = artist.pets.map(pet => `
3332              <div class="work-item bg-[var(--pet-bg)] rounded-xl overflow-hidden cursor-
pointer" onclick="goToPetDetail('${pet.code}')">
3333                <div class="aspect-square flex items-center justify-center p-2">
3334                  <img src="/cwtj/images/${pet.code}/1.png" alt="${pet.name}" class="max-w-
full max-h-full object-contain"
onerror="this.style.display='none';this.parentElement.innerHTML='${pet.name}'">
3335                </div>
3336                <div class="p-2 text-center">
3337                  <span class="text-xs font-medium truncate block">${pet.name}</span>
3338                </div>
3339              </div>
3340            `).join('');
3341          } else {
3342            document.getElementById('modal-works').innerHTML = '<div class="col-span-full
text-center py-8 text-[var(--pet-muted)]">暂无作品</div>';
3343          }
3344
3345          document.getElementById('artist-modal').style.display = 'flex';
3346        } catch (e) {
3347          console.error('加载失败:', e);
3348        }
3349      }
3350
3351      function goToPetDetail(code) {
3352        window.location.href = `/cwtj/pet-detail.html?code=${code}`;
3353      }
3354
3355      function showLoginModal() {
3356        document.getElementById('login-modal').style.display = 'flex';
3357      }
3358
3359      function hideLoginModal() {
3360        document.getElementById('login-modal').style.display = 'none';
3361      }
3362
3363      function updateFollowButton(followed) {
3364        const btn = document.getElementById('btn-follow');
```

```
3365        if (followed) {
3366          btn.textContent = '已关注';
3367          btn.classList.remove('bg-[var(--pet-primary)]');
3368          btn.classList.add('bg-gray-400');
3369          btn.dataset.followed = 'true';
3370        } else {
3371          btn.textContent = '关注';
3372          btn.classList.remove('bg-gray-400');
3373          btn.classList.add('bg-[var(--pet-primary)]');
3374          btn.dataset.followed = 'false';
3375        }
3376      }
3377
3378      document.getElementById('artist-search-box').addEventListener('input', (e) => {
3379        searchKeyword = e.target.value;
3380        renderArtists();
3381      });
3382
3383      document.getElementById('btn-clear-search').addEventListener('click', () => {
3384        document.getElementById('artist-search-box').value = '';
3385        searchKeyword = '';
3386        renderArtists();
3387      });
3388
3389      document.getElementById('btn-close-modal').addEventListener('click', () => {
3390        document.getElementById('artist-modal').style.display = 'none';
3391      });
3392
3393      document.getElementById('btn-follow').addEventListener('click', async () => {
3394        const userId = localStorage.getItem('pet_user_id');
3395        if (!userId) {
3396          showLoginModal();
3397          return;
3398        }
3399
3400        if (!currentArtistCode) {
3401          showToast('画师信息加载失败，请重试', 'error');
3402          return;
3403        }
3404
3405        const btn = document.getElementById('btn-follow');
3406        const artistName = document.getElementById('modal-name').textContent;
3407
3408        try {
3409          const res = await fetch(`${API_BASE}/follows/toggle`, {
3410            method: 'POST',
3411            headers: { 'Content-Type': 'application/json' },
3412            body: JSON.stringify({ userId: parseInt(userId), artistCode: currentArtistCode })
3413          });
3414          const json = await res.json();
```

```
3415
3416          if (json.success) {
3417            if (json.data.followed) {
3418              updateFollowButton(true);
3419              showToast(`成功关注 ${artistName} 😘`);
3420            } else {
3421              updateFollowButton(false);
3422              showToast(`已取消关注 ${artistName}`, 'info');
3423            }
3424            document.getElementById('modal-follower-count').textContent =
json.data.followerCount || 0;
3425          } else {
3426            showToast('操作失败，请重试', 'error');
3427          }
3428        } catch (e) {
3429          showToast('网络错误，请重试', 'error');
3430        }
3431      });
3432
3433      document.getElementById('btn-close-login').addEventListener('click', hideLoginModal);
3434
3435      document.getElementById('btn-login').addEventListener('click', async () => {
3436        const username = document.getElementById('login-username').value;
3437        const password = document.getElementById('login-password').value;
3438        if (!username || !password) {
3439          showToast('请输入用户名和密码', 'error');
3440          return;
3441        }
3442        try {
3443          const res = await fetch(`${API_BASE}/users/login`, {
3444            method: 'POST',
3445            headers: { 'Content-Type': 'application/json' },
3446            body: JSON.stringify({ username, password })
3447          });
3448          const json = await res.json();
3449          if (json.success) {
3450            localStorage.setItem('pet_user_id', json.data.id);
3451            localStorage.setItem('pet_username', json.data.username);
3452            hideLoginModal();
3453            showToast('登录成功！欢迎回来~ 🎉');
3454          } else {
3455            showToast(json.error || '登录失败', 'error');
3456          }
3457        } catch (e) {
3458          showToast('网络错误，请重试', 'error');
3459        }
3460      });
3461
3462      document.getElementById('btn-register').addEventListener('click', async () => {
3463        const username = document.getElementById('login-username').value;
3464        const password = document.getElementById('login-password').value;
```

```
3465        if (!username || !password) {
3466          showToast('请输入用户名和密码', 'error');
3467          return;
3468        }
3469        try {
3470          const res = await fetch(`${API_BASE}/users/register`, {
3471            method: 'POST',
3472            headers: { 'Content-Type': 'application/json' },
3473            body: JSON.stringify({ username, password })
3474          });
3475          const json = await res.json();
3476          if (json.success) {
3477            localStorage.setItem('pet_user_id', json.data.id);
3478            localStorage.setItem('pet_username', json.data.username);
3479            hideLoginModal();
3480            showToast('注册成功! 欢迎加入~ 🐾');
3481          } else {
3482            showToast(json.error || '注册失败', 'error');
3483          }
3484        } catch (e) {
3485          showToast('网络错误，请重试', 'error');
3486        }
3487      });
3488
3489      fetchArtists().then(() => {
3490        const urlParams = new URLSearchParams(window.location.search);
3491        const artistCode = urlParams.get('artist');
3492        if (artistCode) {
3493          showArtistDetail(artistCode);
3494        }
3495      });
3496    </script>
3497
3498    <style>
3499      .artist-card:hover { transform: translateY(-2px); }
3500      .artist-card { transition: all 0.2s; }
3501    </style>
3502    <!-- 百度统计 -->
3503    <script>
3504    var _hmt = _hmt || [];
3505    (function() {
3506      var hm = document.createElement("script");
3507      hm.src = "https://hm.baidu.com/hm.js?ca7a005fbc8241312a51eacc3e983489";
3508      var s = document.getElementsByTagName("script")[0];
3509      s.parentNode.insertBefore(hm, s);
3510    })();
3511    </script>
3512  </body>
3513  </html>
3514  /* ===== END OF public/artist.html ===== */
```

```
/* ===== FILE: public/ranking.html (HTML) ===== */
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="萌宠星球艺术家排行榜 - 粉丝榜、作品榜、授权榜，发现最受欢迎的画师大大">
  <meta name="keywords" content="艺术家排行榜,萌宠画师,粉丝榜,作品榜,授权榜,宠物画师排名">
  <meta property="og:title" content="艺术家排行榜 | 萌宠星球">
  <meta property="og:description" content="发现最受欢迎的萌宠画师">
  <meta property="og:type" content="website">
  <meta name="robots" content="index,follow">
  <link rel="canonical" href="/cwtj/ranking.html">
  <title>艺术家排行榜 | 萌宠星球</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <style>
    :root {
      --pet-primary: #e87a38;
      --pet-primary-soft: #fff5eb;
      --pet-bg: #faf6f1;
      --pet-bg-gradient: radial-gradient(ellipse 120% 80% at 50% -20%, #fef3e9 0%, #faf6f1 50%, #f3f0eb 100%);
      --pet-text: #0f172a;
      --pet-muted: #475569;
      --pet-border: #e2e8f0;
    }
    body { background: var(--pet-bg-gradient); background-color: var(--pet-bg); font-family: ui-sans-serif, system-ui, -apple-system, sans-serif; color: var(--pet-text); }
    .scrollbar-hide::-webkit-scrollbar { display: none; }
    .scrollbar-hide { -ms-overflow-style: none; scrollbar-width: none; }
    @keyframes fadeIn { from { opacity: 0; transform: translateY(10px); } to { opacity: 1; transform: translateY(0); } }
    .animate-fadeIn { animation: fadeIn 0.3s ease-out; }
    @keyframes slideUp { from { opacity: 0; transform: translateY(20px) scale(0.95); } to { opacity: 1; transform: translateY(0) scale(1); } }
    .animate-slideUp { animation: slideUp 0.3s ease-out; }
  </style>
</head>
<body class="min-h-screen py-6 px-4">
  <div class="max-w-4xl mx-auto w-full">
    <div class="sticky-header sticky top-0 z-10 -mx-4 px-4 pt-6 pb-4 bg-[var(--pet-bg)]">
      <div class="flex items-center justify-between">
        <button id="btn-back" class="w-10 h-10 rounded-full bg-white/90 backdrop-blur-sm shadow-md flex items-center justify-center hover:shadow-lg transition-shadow">
          <svg class="w-5 h-5 text-[var(--pet-primary)]" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2"><path d="M19 12H5M12 19l-7-7 7-7"/></svg>
        </button>
        <h1 class="text-xl sm:text-2xl font-extrabold">🏆 艺术家排行榜</h1>
        <div class="w-10"></div>
      </div>
    </div>

    <div class="flex gap-2 mb-6 mt-4">
      <button id="tab-followers" class="flex-1 py-3 bg-[var(--pet-primary)] text-white font-bold rounded-xl">👥 粉丝榜</button>
      <button id="tab-works" class="flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-bold rounded-xl hover:bg-white">🎨 作品榜</button>
      <button id="tab-authorize" class="flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-bold rounded-xl hover:bg-white">★ 授权榜</button>
```

```
3565      </div>
3566
3567      <div id="ranking-list" class="space-y-4">
3568      </div>
3569
3570      <div id="toast" class="fixed bottom-8 left-1/2 -translate-x-1/2 z-50 px-6 py-3 bg-
white/95 backdrop-blur-sm rounded-full shadow-lg flex items-center gap-2 animate-slideUp"
style="display:none">
3571        <span id="toast-icon">🐾</span>
3572        <span id="toast-text" class="text-sm font-medium"></span>
3573      </div>
3574
3575      <div id="login-modal" class="fixed inset-0 bg-black/50 z-50 flex items-center justify-
center p-4" style="display:none">
3576        <div class="bg-white rounded-2xl p-6 w-full max-w-sm animate-fadeIn">
3577          <h3 class="text-xl font-bold text-center mb-4">请先登录</h3>
3578          <div class="space-y-3">
3579            <input type="text" id="login-username" placeholder="用户名" class="w-full px-4 py-
3 border border-[var(--pet-border)] rounded-xl outline-none focus:border-[var(--pet-primary)]">
3580            <input type="password" id="login-password" placeholder="密码" class="w-full px-4
py-3 border border-[var(--pet-border)] rounded-xl outline-none focus:border-[var(--pet-
primary)]">
3581          </div>
3582          <button id="btn-login" class="w-full mt-4 py-3 bg-[var(--pet-primary)] text-white
font-bold rounded-xl hover:opacity-90">登录</button>
3583          <button id="btn-register" class="w-full mt-2 py-3 border border-[var(--pet-primary)]
text-[var(--pet-primary)] font-bold rounded-xl">注册</button>
3584          <button id="btn-close-login" class="w-full mt-2 py-2 text-[var(--pet-muted)] text-
sm">取消</button>
3585        </div>
3586      </div>
3587    </div>
3588
3589    <script>
3590      // 动态获取 API 基础路径（支持 / 和 /cwtj/ 两种模式）
3591      const isSubPath = window.location.pathname.startsWith('/cwtj');
3592      const API_BASE = window.location.origin + (isSubPath ? '/cwtj/apic' : '/apic');
3593      let allArtists = [];
3594
3595      async function loadArtists() {
3596        const res = await fetch(`${API_BASE}/artists`);
3597        const json = await res.json();
3598        if (json.success) {
3599          allArtists = json.data;
3600          renderFollowers();
3601        }
3602      }
3603
3604      function showToast(message, type = 'success') {
3605        const toast = document.getElementById('toast');
3606        const icon = document.getElementById('toast-icon');
3607        const text = document.getElementById('toast-text');
3608
3609        text.textContent = message;
3610
3611        if (type === 'success') {
3612          icon.textContent = '✨';
3613          toast.style.borderTop = '3px solid #22c55e';
3614          toast.style.background = 'linear-gradient(135deg, #f0fdf4 0%, #ffffff 100%)';
```

```
3615        } else if (type === 'error') {
3616          icon.textContent = '';
3617          toast.style.borderTop = '3px solid #ef4444';
3618          toast.style.background = 'linear-gradient(135deg, #fef2f2 0%, #ffffff 100%)';
3619        } else if (type === 'info') {
3620          icon.textContent = '';
3621          toast.style.borderTop = '3px solid #3b82f6';
3622          toast.style.background = 'linear-gradient(135deg, #eff6ff 0%, #ffffff 100%)';
3623        }
3624
3625        toast.style.display = 'flex';
3626        setTimeout(() => {
3627          toast.style.display = 'none';
3628        }, 2000);
3629      }
3630
3631      function showLoginModal() {
3632        document.getElementById('login-modal').style.display = 'flex';
3633      }
3634
3635      function hideLoginModal() {
3636        document.getElementById('login-modal').style.display = 'none';
3637      }
3638
3639      document.getElementById('btn-back').addEventListener('click', () => {
3640        window.history.back();
3641      });
3642
3643      const tabFollowers = document.getElementById('tab-followers');
3644      const tabWorks = document.getElementById('tab-works');
3645      const tabAuthorize = document.getElementById('tab-authorize');
3646      const rankingList = document.getElementById('ranking-list');
3647
3648      function renderRanking(data, label) {
3649        rankingList.innerHTML = data.map((item, index) => {
3650          const rankIcon = item.rank === 1 ? '' : item.rank === 2 ? '' : item.rank === 3 ?
'' : item.rank;
3651          const rankBg = item.rank === 1 ? 'bg-gradient-to-br from-yellow-400 to-amber-500' :
3652                         item.rank === 2 ? 'bg-gradient-to-br from-gray-300 to-gray-400' :
3653                         item.rank === 3 ? 'bg-gradient-to-br from-amber-600 to-amber-700' :
'bg-[var(--pet-border)]';
3654          const rankTextColor = item.rank <= 3 ? 'text-white' : 'text-[var(--pet-muted)]';
3655
3656          return `
3657            <div class="animate-fadeIn" style="animation-delay: ${index * 0.05}s">
3658              <div class="bg-white/90 backdrop-blur-sm rounded-2xl p-4 flex items-center gap-
4 shadow-md cursor-pointer hover:shadow-lg transition-shadow"
onclick="goToArtist('${item.code}')">
3659                <div class="w-12 h-12 ${rankBg} rounded-full flex items-center justify-center
${rankTextColor} text-xl font-bold shadow-lg">${rankIcon}</div>
3660                <img src="${item.avatar || getDefaultAvatar(item.code)}" alt="${item.name}"
class="w-14 h-14 rounded-full object-cover" onerror="this.src='${getDefaultAvatar(item.code)}'">
3661                <div class="flex-1">
3662                  <div class="font-bold text-lg">${item.name}</div>
3663                  <div class="text-sm text-[var(--pet-muted)]">${item.title}</div>
3664                  <div class="text-xs text-[var(--pet-primary)] font-medium mt-1">${label}:
${item.value}</div>
```

```
3665                </div>
3666                <button class="px-4 py-2 bg-[var(--pet-primary-soft)] text-[var(--pet-
primary)] rounded-full text-sm font-medium hover:bg-[var(--pet-primary)] hover:text-white
transition-all follow-btn" data-code="${item.code}">关注</button>
3667              </div>
3668            </div>
3669          `;
3670        }).join('');
3671
3672        bindFollowButtons();
3673      }
3674
3675      function getDefaultAvatar(code) {
3676        const colors = ['#e87a38', '#3b82f6', '#22c55e', '#f59e0b', '#ec4899'];
3677        const color = colors[code.length % colors.length];
3678        return `data:image/svg+xml,%3Csvg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 48
48"%3E%3Ccircle cx="24" cy="24" r="20" fill="${color}"%3E%3C/circle%3E%3Ctext x="24" y="30" text-
anchor="middle" fill="white" font-size="20" font-
weight="bold"%3E${code.charAt(0).toUpperCase()}%3C/text%3E%3C/svg%3E`;
3679      }
3680
3681      function goToArtist(code) {
3682        window.location.href = `/cwtj/artist.html?artist=${code}`;
3683      }
3684
3685      async function bindFollowButtons() {
3686        document.querySelectorAll('.follow-btn').forEach(btn => {
3687          btn.addEventListener('click', async (e) => {
3688            e.stopPropagation();
3689            const userId = localStorage.getItem('pet_user_id');
3690            if (!userId) {
3691              showLoginModal();
3692              return;
3693            }
3694
3695            const artistCode = btn.dataset.code;
3696            const artistName = btn.parentElement.querySelector('.font-bold').textContent;
3697
3698            const res = await fetch(`${API_BASE}/follows/toggle`, {
3699              method: 'POST',
3700              headers: { 'Content-Type': 'application/json' },
3701              body: JSON.stringify({ userId, artistCode })
3702            });
3703            const json = await res.json();
3704
3705            if (json.success) {
3706              if (json.data.followed) {
3707                btn.textContent = '已关注';
3708                btn.classList.remove('bg-[var(--pet-primary-soft)]', 'text-[var(--pet-
primary)]');
3709                btn.classList.add('bg-[var(--pet-primary)]', 'text-white');
3710                showToast(`成功关注 ${artistName} 🥰`);
3711              } else {
3712                btn.textContent = '关注';
3713                btn.classList.remove('bg-[var(--pet-primary)]', 'text-white');
3714                btn.classList.add('bg-[var(--pet-primary-soft)]', 'text-[var(--pet-
primary)]');
```

```
3715                showToast(`已取消关注 ${artistName}`, 'info');
3716              }
3717            }
3718          });
3719        });
3720      }
3721
3722      function renderFollowers() {
3723        const data = [...allArtists]
3724          .sort((a, b) => (b.followerCount || 0) - (a.followerCount || 0))
3725          .slice(0, 5)
3726          .map((artist, index) => ({
3727            rank: index + 1,
3728            name: artist.name,
3729            title: artist.title,
3730            value: (artist.followerCount || 0).toLocaleString(),
3731            avatar: artist.avatarUrl,
3732            code: artist.code
3733          }));
3734        renderRanking(data, '粉丝数');
3735      }
3736
3737      function renderWorks() {
3738        const data = [...allArtists]
3739          .sort((a, b) => (b.petCount || 0) - (a.petCount || 0))
3740          .slice(0, 5)
3741          .map((artist, index) => ({
3742            rank: index + 1,
3743            name: artist.name,
3744            title: artist.title,
3745            value: (artist.petCount || 0).toString(),
3746            avatar: artist.avatarUrl,
3747            code: artist.code
3748          }));
3749        renderRanking(data, '作品数');
3750      }
3751
3752      function renderAuthorize() {
3753        const data = [...allArtists]
3754          .sort((a, b) => (b.totalAuthorization || 0) - (a.totalAuthorization || 0))
3755          .slice(0, 5)
3756          .map((artist, index) => ({
3757            rank: index + 1,
3758            name: artist.name,
3759            title: artist.title,
3760            value: (artist.totalAuthorization || 0).toLocaleString(),
3761            avatar: artist.avatarUrl,
3762            code: artist.code
3763          }));
3764        renderRanking(data, '授权数');
```

```
3765      }
3766
3767      tabFollowers.addEventListener('click', () => {
3768        tabFollowers.className = 'flex-1 py-3 bg-[var(--pet-primary)] text-white font-bold
rounded-xl';
3769        tabWorks.className = 'flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-bold
rounded-xl hover:bg-white';
3770        tabAuthorize.className = 'flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-bold
rounded-xl hover:bg-white';
3771        renderFollowers();
3772      });
3773
3774      tabWorks.addEventListener('click', () => {
3775        tabFollowers.className = 'flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-bold
rounded-xl hover:bg-white';
3776        tabWorks.className = 'flex-1 py-3 bg-[var(--pet-primary)] text-white font-bold
rounded-xl';
3777        tabAuthorize.className = 'flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-bold
rounded-xl hover:bg-white';
3778        renderWorks();
3779      });
3780
3781      tabAuthorize.addEventListener('click', () => {
3782        tabFollowers.className = 'flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-bold
rounded-xl hover:bg-white';
3783        tabWorks.className = 'flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-bold
rounded-xl hover:bg-white';
3784        tabAuthorize.className = 'flex-1 py-3 bg-[var(--pet-primary)] text-white font-bold
rounded-xl';
3785        renderAuthorize();
3786      });
3787
3788      document.getElementById('btn-close-login').addEventListener('click', hideLoginModal);
3789
3790      document.getElementById('btn-login').addEventListener('click', async () => {
3791        const username = document.getElementById('login-username').value;
3792        const password = document.getElementById('login-password').value;
3793        if (!username || !password) {
3794          showToast('请输入用户名和密码', 'error');
3795          return;
3796        }
3797
3798        const res = await fetch(`${API_BASE}/users/login`, {
3799          method: 'POST',
3800          headers: { 'Content-Type': 'application/json' },
3801          body: JSON.stringify({ username, password })
3802        });
3803        const json = await res.json();
3804
3805        if (json.success) {
3806          localStorage.setItem('pet_user_id', json.data.id);
3807          hideLoginModal();
3808          showToast('登录成功! 欢迎回来~ 🎉');
3809          loadArtists();
3810        } else {
3811          showToast(json.error || '登录失败', 'error');
3812        }
3813      });
3814
```

```
3815      document.getElementById('btn-register').addEventListener('click', async () => {
3816        const username = document.getElementById('login-username').value;
3817        const password = document.getElementById('login-password').value;
3818        if (!username || !password) {
3819          showToast('请输入用户名和密码', 'error');
3820          return;
3821        }
3822
3823        const res = await fetch(`${API_BASE}/users/register`, {
3824          method: 'POST',
3825          headers: { 'Content-Type': 'application/json' },
3826          body: JSON.stringify({ username, password })
3827        });
3828        const json = await res.json();
3829
3830        if (json.success) {
3831          localStorage.setItem('pet_user_id', json.data.id);
3832          hideLoginModal();
3833          showToast('注册成功! 欢迎加入~ 🐾');
3834          loadArtists();
3835        } else {
3836          showToast(json.error || '注册失败', 'error');
3837        }
3838      });
3839
3840      loadArtists();
3841    </script>
3842  </body>
3843  </html>
3844  /* ===== END OF public/ranking.html ===== */
3845  /* ===== FILE: public/my-pets.html (HTML) ===== */
3846  <!DOCTYPE html>
3847  <html lang="zh-CN">
3848  <head>
3849    <meta charset="UTF-8">
3850    <meta name="viewport" content="width=device-width, initial-scale=1.0">
3851    <meta name="description" content="我的宠物小屋 - 领养、收藏、关注，打造你的专属萌宠空间">
3852    <meta name="keywords" content="我的宠物,宠物领养,宠物收藏,关注画师,萌宠星球">
3853    <meta name="robots" content="noindex,nofollow">
3854    <title>我的宠物小屋 | 萌宠星球</title>
3855    <script src="https://cdn.tailwindcss.com"></script>
3856    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/@unocss/reset/tailwind.min.css">
3857    <style>
3858      :root {
3859        --pet-primary: #e87a38;
3860        --pet-primary-soft: #fff5eb;
3861        --pet-bg: #faf6f1;
3862        --pet-bg-gradient: radial-gradient(ellipse 120% 80% at 50% -20%, #fef3e9 0%, #faf6f1
50%, #f3f0eb 100%);
3863        --pet-text: #0f172a;
3864        --pet-muted: #475569;
```

```
3865        --pet-border: #e2e8f0;
3866      }
3867      body { background: var(--pet-bg-gradient); background-color: var(--pet-bg); font-family:
ui-sans-serif, system-ui, -apple-system, sans-serif; color: var(--pet-text); }
3868      .scrollbar-hide::-webkit-scrollbar { display: none; }
3869      .scrollbar-hide { -ms-overflow-style: none; scrollbar-width: none; }
3870      @keyframes fadeIn { from { opacity: 0; transform: translateY(10px); } to { opacity: 1;
transform: translateY(0); } }
3871      .animate-fadeIn { animation: fadeIn 0.3s ease-out; }
3872    </style>
3873  </head>
3874  <body class="min-h-screen py-6 px-4">
3875    <div class="max-w-4xl mx-auto w-full">
3876      <!-- Header -->
3877      <div class="sticky-header sticky top-0 z-10 -mx-4 px-4 pt-6 pb-4 bg-[var(--pet-bg)]">
3878        <div class="flex items-center justify-between">
3879          <button id="btn-back" class="w-10 h-10 rounded-full bg-white/90 backdrop-blur-sm
shadow-md flex items-center justify-center hover:shadow-lg transition-shadow">
3880            <svg class="w-5 h-5 text-[var(--pet-primary)]" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2"><path d="M19 12H5M12 19l-7-7 7-7"/></svg>
3881          </button>
3882          <h1 class="text-xl sm:text-2xl font-extrabold">🐾 我的宠物小屋</h1>
3883          <div class="w-10"></div>
3884        </div>
3885      </div>
3886
3887      <!-- Stats -->
3888      <div class="grid grid-cols-2 gap-4 mb-6 mt-4">
3889        <div class="text-center p-4 bg-white/80 rounded-2xl backdrop-blur-sm">
3890          <div id="adopt-count" class="text-2xl md:text-3xl font-bold text-[var(--pet-
primary)]">0</div>
3891          <div class="text-xs text-[var(--pet-muted)] mt-1">已领养</div>
3892        </div>
3893        <div class="text-center p-4 bg-white/80 rounded-2xl backdrop-blur-sm">
3894          <div id="favorite-count" class="text-2xl md:text-3xl font-bold text-[var(--pet-
primary)]">0</div>
3895          <div class="text-xs text-[var(--pet-muted)] mt-1">收藏夹</div>
3896        </div>
3897      </div>
3898
3899      <!-- Tabs -->
3900      <div class="flex gap-2 mb-6">
3901        <button id="tab-adopt" class="flex-1 py-3 bg-[var(--pet-primary)] text-white font-
bold rounded-xl">🏠 我的宠物</button>
3902        <button id="tab-favorite" class="flex-1 py-3 bg-white/80 text-[var(--pet-muted)]
font-bold rounded-xl hover:bg-white">💜 收藏夹</button>
3903        <button id="tab-follows" class="flex-1 py-3 bg-white/80 text-[var(--pet-muted)] font-
bold rounded-xl hover:bg-white">👥 我的关注</button>
3904      </div>
3905
3906      <!-- Content -->
3907      <div id="content-adopt" class="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 gap-4">
3908        <div class="col-span-full text-center py-20 text-[var(--pet-muted)]">
3909          <div class="text-6xl mb-4">🏠</div>
3910          <p>还没有领养宠物</p>
3911          <p class="text-sm mt-2">去宠物图鉴领养吧！</p>
3912        </div>
3913      </div>
3914
```

```
3915      <div id="content-favorite" class="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 gap-4"
style="display:none">
3916        <div class="col-span-full text-center py-20 text-[var(--pet-muted)]">
3917          <div class="text-6xl mb-4">🤍</div>
3918          <p>还没有收藏宠物</p>
3919          <p class="text-sm mt-2">去宠物图鉴收藏吧！</p>
3920        </div>
3921      </div>
3922
3923      <div id="content-follows" class="space-y-4" style="display:none">
3924        <div class="col-span-full text-center py-20 text-[var(--pet-muted)]">
3925          <div class="text-6xl mb-4">👥</div>
3926          <p>还没有关注艺术家</p>
3927          <p class="text-sm mt-2">去艺术家广场看看吧！</p>
3928        </div>
3929      </div>
3930
3931      <!-- Login Modal -->
3932      <div id="login-modal" class="fixed inset-0 bg-black/50 z-50 flex items-center justify-
center p-4" style="display:none">
3933        <div class="bg-white rounded-2xl p-6 w-full max-w-sm animate-fadeIn">
3934          <h3 class="text-xl font-bold text-center mb-4">请先登录</h3>
3935          <div class="space-y-3">
3936            <input type="text" id="login-username" placeholder="用户名" class="w-full px-4 py-
3 border border-[var(--pet-border)] rounded-xl outline-none focus:border-[var(--pet-primary)]">
3937            <input type="password" id="login-password" placeholder="密码" class="w-full px-4
py-3 border border-[var(--pet-border)] rounded-xl outline-none focus:border-[var(--pet-
primary)]">
3938          </div>
3939          <button id="btn-login" class="w-full mt-4 py-3 bg-[var(--pet-primary)] text-white
font-bold rounded-xl hover:opacity-90">登录</button>
3940          <button id="btn-register" class="w-full mt-2 py-3 border border-[var(--pet-primary)]
text-[var(--pet-primary)] font-bold rounded-xl">注册</button>
3941          <button id="btn-close-login" class="w-full mt-2 py-2 text-[var(--pet-muted)] text-
sm">取消</button>
3942        </div>
3943      </div>
3944    </div>
3945
3946    <script>
3947      // 动态获取 API 基础路径（支持 / 和 /cwtj/ 两种模式）
3948      const isSubPath = window.location.pathname.startsWith('/cwtj');
3949      const API_BASE = window.location.origin + (isSubPath ? '/cwtj/apic' : '/apic');
3950      let adoptedPets = [];
3951      let favoritePets = [];
3952      let followedArtists = [];
3953
3954      async function loadMyPets() {
3955        const userId = localStorage.getItem('pet_user_id');
3956        if (!userId) {
3957          showLoginModal();
3958          return;
3959        }
3960
3961        const [adoptRes, favoriteRes, followRes] = await Promise.all([
3962          fetch(`${API_BASE}/users/${userId}/adopts`),
3963          fetch(`${API_BASE}/users/${userId}/favorites`),
3964          fetch(`${API_BASE}/users/${userId}/follows`)
```

```
3965        ]);
3966
3967        const adoptJson = await adoptRes.json();
3968        const favoriteJson = await favoriteRes.json();
3969        const followJson = await followRes.json();
3970
3971        adoptedPets = adoptJson.success ? adoptJson.data : [];
3972        favoritePets = favoriteJson.success ? favoriteJson.data : [];
3973        followedArtists = followJson.success ? followJson.data : [];
3974
3975        document.getElementById('adopt-count').textContent = adoptedPets.length;
3976        document.getElementById('favorite-count').textContent = favoritePets.length;
3977
3978        await renderAdoptPets();
3979      }
3980
3981      async function renderAdoptPets() {
3982        if (adoptedPets.length === 0) {
3983          document.getElementById('content-adopt').innerHTML = `
3984            <div class="col-span-full text-center py-20 text-[var(--pet-muted)]">
3985              <div class="text-6xl mb-4">🏠</div>
3986              <p>还没有领养宠物</p>
3987              <button onclick="goToIndex()" class="mt-4 px-6 py-2 bg-[var(--pet-primary)]
text-white rounded-full text-sm">去领养</button>
3988            </div>
3989          `;
3990          return;
3991        }
3992
3993        const html = await Promise.all(adoptedPets.map(async code => {
3994          const res = await fetch(`${API_BASE}/pets/${code}`);
3995          const json = await res.json();
3996          if (!json.success) return '';
3997
3998          const pet = json.data;
3999          return `
4000            <div class="pet-card bg-white/90 backdrop-blur-sm rounded-2xl overflow-hidden
shadow-md hover:shadow-lg transition-shadow cursor-pointer animate-fadeIn"
onclick="goToDetail('${code}')">
4001              <div class="aspect-square bg-gradient-to-br from-orange-50 to-amber-50 flex
items-center justify-center p-4">
4002                <img src="/cwtj/images/${code}/1.png" alt="${pet.name}" class="max-w-full
max-h-full object-contain" onerror="this.src='${pet.stages?.[0]?.image || ''}'">
4003              </div>
4004              <div class="p-3 text-center">
4005                <span class="font-bold text-sm">${pet.name}</span>
4006                <div class="flex items-center justify-center gap-1 mt-1">
4007                  <span class="text-[10px] text-[var(--pet-muted)]">Lv.${pet.stages?.length
|| 1}</span>
4008                </div>
4009              </div>
4010            </div>
4011          `;
4012        }));
4013
4014        document.getElementById('content-adopt').innerHTML = html.join('');
```

```
4015      }
4016
4017      async function renderFavoritePets() {
4018        if (favoritePets.length === 0) {
4019          document.getElementById('content-favorite').innerHTML = `
4020            <div class="col-span-full text-center py-20 text-[var(--pet-muted)]">
4021              <div class="text-6xl mb-4">💔</div>
4022              <p>还没有收藏宠物</p>
4023              <button onclick="goToIndex()" class="mt-4 px-6 py-2 bg-[var(--pet-primary)]
text-white rounded-full text-sm">去收藏</button>
4024            </div>
4025          `;
4026          return;
4027        }
4028
4029        const html = await Promise.all(favoritePets.map(async code => {
4030          const res = await fetch(`${API_BASE}/pets/${code}`);
4031          const json = await res.json();
4032          if (!json.success) return '';
4033
4034          const pet = json.data;
4035          return `
4036            <div class="pet-card bg-white/90 backdrop-blur-sm rounded-2xl overflow-hidden
shadow-md hover:shadow-lg transition-shadow cursor-pointer animate-fadeIn"
onclick="goToDetail('${code}')">
4037              <div class="aspect-square bg-gradient-to-br from-pink-50 to-red-50 flex items-
center justify-center p-4">
4038                <img src="/cwtj/images/${code}/1.png" alt="${pet.name}" class="max-w-full
max-h-full object-contain" onerror="this.src='${pet.stages?.[0]?.image || ''}'">
4039              </div>
4040              <div class="p-3 text-center">
4041                <span class="font-bold text-sm">${pet.name}</span>
4042                <div class="flex items-center justify-center gap-1 mt-1">
4043                  <span class="text-[10px] text-[var(--pet-muted)]">Lv.${pet.stages?.length
|| 1}</span>
4044                </div>
4045              </div>
4046            </div>
4047          `;
4048        }));
4049
4050        document.getElementById('content-favorite').innerHTML = html.join('');
4051      }
4052
4053      function renderFollowedArtists() {
4054        if (followedArtists.length === 0) {
4055          document.getElementById('content-follows').innerHTML = `
4056            <div class="col-span-full text-center py-20 text-[var(--pet-muted)]">
4057              <div class="text-6xl mb-4">👥</div>
4058              <p>还没有关注艺术家</p>
4059              <button onclick="goToArtist()" class="mt-4 px-6 py-2 bg-[var(--pet-primary)]
text-white rounded-full text-sm">去关注</button>
4060            </div>
4061          `;
4062          return;
4063        }
4064
```

```
4065        const html = followedArtists.map((code, index) => {
4066          const artist = getArtistByCode(code);
4067          return `
4068            <div class="artist-item bg-white/90 backdrop-blur-sm rounded-2xl p-4 flex items-
center gap-4 shadow-md cursor-pointer hover:shadow-lg transition-all animate-fadeIn"
style="animation-delay: ${index * 0.05}s" onclick="goToArtistDetail('${code}')">
4069              <img src="${artist?.avatar || getDefaultAvatar(code)}" alt="${artist?.name ||
code}" class="w-12 h-12 rounded-full object-cover"
onerror="this.src='${getDefaultAvatar(code)}'">
4070              <div class="flex-1">
4071                <div class="font-bold">${artist?.name || code}</div>
4072                <div class="text-sm text-[var(--pet-muted)]">${artist?.title || '艺术家
'}</div>
4073              </div>
4074              <button onclick="event.stopPropagation(); unfollowArtist('${code}')" class="px-
4 py-2 bg-red-50 text-red-500 rounded-full text-sm font-medium hover:bg-red-100 transition-
colors">取消关注</button>
4075            </div>
4076          `;
4077        }).join('');
4078
4079        document.getElementById('content-follows').innerHTML = html;
4080      }
4081
4082      function getArtistByCode(code) {
4083        const artists = [
4084          { code: 'xiaoju', name: '画师小橘', title: '海洋萌宠设计师', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20orange%20hair%20chibi%20artist%20avatar&ima
ge_size=square' },
4085          { code: 'achai', name: '阿柴画社', title: '治愈猫咪主理人', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20brown%20hair%20chibi%20artist%20avatar&imag
e_size=square' },
4086          { code: 'yishou', name: '异兽造梦师', title: '幻想神兽缔造者', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20purple%20hair%20magical%20chibi%20artist%20
avatar&image_size=square' },
4087          { code: 'yinghuo', name: '萤火画师', title: '微光精灵创作者', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20blonde%20hair%20firefly%20chibi%20artist%20
avatar&image_size=square' },
4088          { code: 'yunjing', name: '云鲸画师', title: '天空幻想家', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20blue%20hair%20whale%20chibi%20artist%20avat
ar&image_size=square' },
4089          { code: 'xigaodi', name: '西高地工坊', title: '萌犬专家', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20white%20hair%20dog%20chibi%20artist%20avata
r&image_size=square' },
4090          { code: 'mengchong', name: '萌宠小筑', title: '治愈系画师', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20pink%20hair%20cute%20chibi%20artist%20avata
r&image_size=square' },
4091          { code: 'luhuahua', name: '鹿花花', title: '森系精灵画师', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20green%20hair%20deer%20chibi%20artist%20avat
ar&image_size=square' },
4092          { code: 'huohua', name: '火华', title: '国潮新锐画师', avatar: 'https://neeko-
copilot.bytedance.net/api/text_to_image?prompt=cute%20red%20hair%20chinese%20style%20chibi%20arti
st%20avatar&image_size=square' }
4093        ];
4094        return artists.find(a => a.code === code);
4095      }
4096
4097      function getDefaultAvatar(code) {
4098        const colors = ['#e87a38', '#3b82f6', '#22c55e', '#f59e0b', '#ec4899'];
4099        const color = colors[code.length % colors.length];
4100        return `data:image/svg+xml,%3Csvg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 48
48" fill="${color}"%3E%3Ccircle cx="24" cy="24" r="20" fill="${color}"%3E%3C/circle%3E%3Ctext
x="24" y="30" text-anchor="middle" fill="white" font-size="20" font-
weight="bold"%3E${code.charAt(0).toUpperCase()}%3C/text%3E%3C/svg%3E`;
4101      }
4102
4103      function goToArtistDetail(code) {
4104        window.location.href = `/cwtj/artist.html?artist=${code}`;
```

```
4105      }
4106
4107      function goToDetail(code) {
4108        window.location.href = `/cwtj/pet-detail.html?code=${code}`;
4109      }
4110
4111      function goToIndex() {
4112        window.location.href = '/cwtj/';
4113      }
4114
```

```
4115      function showLoginModal() {
4116        document.getElementById('login-modal').style.display = 'flex';
4117      }
4118
4119      function hideLoginModal() {
4120        document.getElementById('login-modal').style.display = 'none';
4121      }
4122
4123      // Event Bindings
4124      document.getElementById('btn-back').addEventListener('click', () => {
4125        window.history.back();
4126      });
4127
4128      document.getElementById('tab-adopt').addEventListener('click', async () => {
4129        document.getElementById('tab-adopt').className = 'flex-1 py-3 bg-[var(--pet-primary)]
text-white font-bold rounded-xl';
4130        document.getElementById('tab-favorite').className = 'flex-1 py-3 bg-white/80 text-
[var(--pet-muted)] font-bold rounded-xl hover:bg-white';
4131        document.getElementById('tab-follows').className = 'flex-1 py-3 bg-white/80 text-
[var(--pet-muted)] font-bold rounded-xl hover:bg-white';
4132        document.getElementById('content-adopt').style.display = 'grid';
4133        document.getElementById('content-favorite').style.display = 'none';
4134        document.getElementById('content-follows').style.display = 'none';
4135      });
4136
4137      document.getElementById('tab-favorite').addEventListener('click', async () => {
4138        document.getElementById('tab-adopt').className = 'flex-1 py-3 bg-white/80 text-[var(-
-pet-muted)] font-bold rounded-xl hover:bg-white';
4139        document.getElementById('tab-favorite').className = 'flex-1 py-3 bg-[var(--pet-
primary)] text-white font-bold rounded-xl';
4140        document.getElementById('tab-follows').className = 'flex-1 py-3 bg-white/80 text-
[var(--pet-muted)] font-bold rounded-xl hover:bg-white';
4141        document.getElementById('content-adopt').style.display = 'none';
4142        document.getElementById('content-favorite').style.display = 'grid';
4143        document.getElementById('content-follows').style.display = 'none';
4144        await renderFavoritePets();
4145      });
4146
4147      document.getElementById('tab-follows').addEventListener('click', () => {
4148        document.getElementById('tab-adopt').className = 'flex-1 py-3 bg-white/80 text-[var(-
-pet-muted)] font-bold rounded-xl hover:bg-white';
4149        document.getElementById('tab-favorite').className = 'flex-1 py-3 bg-white/80 text-
[var(--pet-muted)] font-bold rounded-xl hover:bg-white';
4150        document.getElementById('tab-follows').className = 'flex-1 py-3 bg-[var(--pet-
primary)] text-white font-bold rounded-xl';
4151        document.getElementById('content-adopt').style.display = 'none';
4152        document.getElementById('content-favorite').style.display = 'none';
4153        document.getElementById('content-follows').style.display = 'block';
4154        renderFollowedArtists();
4155      });
4156
4157      document.getElementById('btn-close-login').addEventListener('click', hideLoginModal);
4158
4159      document.getElementById('btn-login').addEventListener('click', async () => {
4160        const username = document.getElementById('login-username').value;
4161        const password = document.getElementById('login-password').value;
4162        if (!username || !password) {
4163          alert('请输入用户名和密码');
4164          return;
```

```
4165        }
4166
4167        const res = await fetch(`${API_BASE}/users/login`, {
4168          method: 'POST',
4169          headers: { 'Content-Type': 'application/json' },
4170          body: JSON.stringify({ username, password })
4171        });
4172        const json = await res.json();
4173
4174        if (json.success) {
4175          localStorage.setItem('pet_user_id', json.data.id);
4176          localStorage.setItem('pet_username', json.data.username);
4177          hideLoginModal();
4178          loadMyPets();
4179        } else {
4180          alert(json.error || '登录失败');
4181        }
4182      });
4183
4184      document.getElementById('btn-register').addEventListener('click', async () => {
4185        const username = document.getElementById('login-username').value;
4186        const password = document.getElementById('login-password').value;
4187        if (!username || !password) {
4188          alert('请输入用户名和密码');
4189          return;
4190        }
4191
4192        const res = await fetch(`${API_BASE}/users/register`, {
4193          method: 'POST',
4194          headers: { 'Content-Type': 'application/json' },
4195          body: JSON.stringify({ username, password })
4196        });
4197        const json = await res.json();
4198
4199        if (json.success) {
4200          localStorage.setItem('pet_user_id', json.data.id);
4201          localStorage.setItem('pet_username', json.data.username);
4202          hideLoginModal();
4203          loadMyPets();
4204        } else {
4205          alert(json.error || '注册失败');
4206        }
4207      });
4208
4209      // Initialize
4210      loadMyPets();
4211    </script>
4212  </body>
4213  </html>
4214  /* ===== END OF public/my-pets.html ===== */
```