# 🌍 실시간 환율 정보 및 분석 웹 서비스 개발계획서 (API 업데이트 버전)

## 📌 프로젝트 개요

실시간 환율 정보를 제공하고, 지난 한 달간의 환율 변동을 분석하여 예측 자료까지 제공하며, 환차손을 자동으로 계산해주는 초보자 친화적 웹 서비스입니다.

\*\*⚠️ 중요 업데이트\*\*: 한국은행 Open API가 8월 말까지 사용 불가로 인해 \*\*ExchangeRate-API\*\* 등 대체 API 사용

---

## 🎯 1. 프로젝트 목표

- \*\*최신 환율 정보 바로 확인\*\*: ExchangeRate-API를 통한 실시간 환율 정보 제공

- \*\*지난 환율 흐름 파악\*\*: 지난 한 달간 환율 변동 테이블 + 간단한 라인 차트

- \*\*환율 분석 및 예측\*\*: 단순 이동평균선 기반 트렌드 분석 제공

- \*\*환차손 자동 계산\*\*: (현재환율 - 매입환율) × 환전금액 = 환차손익 공식 적용

- \*\*깔끔한 디자인\*\*: 흰색, 검정색, 레몬색만 사용한 미니멀 디자인

- \*\*정보 인쇄 기능\*\*: 환율 정보 및 분석 자료 인쇄 지원

---

## 🔧 2. 기술 스택 및 데이터 관리

### 2.1 기술 스택

- \*\*Frontend\*\*: HTML5, CSS3, Vanilla JavaScript

- \*\*데이터 소스\*\*:

- \*\*1순위\*\*: ExchangeRate-API (무료, 사용 쉬움)

- \*\*2순위\*\*: CurrencyAPI (백업용)

- \*\*3순위\*\*: Fixer.io (보조용)

- \*\*차트 라이브러리\*\*: Chart.js (경량화)

- \*\*아이콘\*\*: Font Awesome (달러 아이콘 등)

### 2.2 선택한 API 상세 정보

#### \*\*ExchangeRate-API (메인 사용) ⭐추천\*\*

```javascript

// 기본 API 호출 (무료, 제한적)

const basicURL = 'https://api.exchangerate-api.com/v4/latest/USD';

// 회원가입 후 API 키 사용 (권장)

const apiKey = 'your-api-key-here';

const premiumURL = `https://v6.exchangerate-api.com/v6/${apiKey}/latest/USD`;

// 과거 데이터 조회

const historicalURL = `https://api.exchangerate-api.com/v4/history/USD/2024-01-15`;

```

\*\*장점:\*\*

- ✅ 완전 무료 (월 1,500회 요청)

- ✅ USD 기준 통화 설정 가능

- ✅ 한국원화(KRW) 완벽 지원

- ✅ 과거 데이터 조회 가능

- ✅ CORS 문제 없음

\*\*API 응답 예시:\*\*

```json

{

"base": "USD",

"date": "2024-01-20",

"rates": {

"KRW": 1342.50,

"EUR": 0.92,

"JPY": 148.35,

"CNY": 7.23

}

}

```

### 2.3 데이터 저장 방식 ⭐\*\*추천: 브라우저 로컬스토리지\*\*

\*\*선택 이유:\*\*

- ✅ 설정 간단 (서버 불필요)

- ✅ 비용 무료

- ✅ 빠른 접근 속도

- ✅ 초보자도 쉽게 구현

```javascript

// 환율 데이터 저장 예시

localStorage.setItem('exchangeRates', JSON.stringify(rateData));

localStorage.setItem('lastUpdate', new Date().toISOString());

// 백업 API 정보도 저장

localStorage.setItem('apiSource', 'ExchangeRate-API');

localStorage.setItem('fallbackAPIs', JSON.stringify(['CurrencyAPI', 'Fixer.io']));

```

### 2.4 지원 통화

- \*\*USD\*\* (미국 달러) - 기준 통화

- \*\*EUR\*\* (유로)

- \*\*JPY\*\* (일본 엔)

- \*\*CNY\*\* (중국 위안)

---

## 🔄 3. API 연동 및 에러 처리 시스템

### 3.1 다중 API 백업 시스템

```javascript

class ExchangeRateService {

constructor() {

this.apis = [

{

name: 'ExchangeRate-API',

url: 'https://api.exchangerate-api.com/v4/latest/USD',

primary: true

},

{

name: 'CurrencyAPI',

url: 'https://api.currencyapi.com/v3/latest?apikey=YOUR\_KEY&base\_currency=USD',

primary: false

},

{

name: 'Fixer.io',

url: 'https://data.fixer.io/api/latest?access\_key=YOUR\_KEY&base=EUR',

primary: false,

needsConversion: true // EUR 기준이므로 USD로 변환 필요

}

];

this.currentAPI = 0;

}

async fetchExchangeRates() {

for (let i = 0; i < this.apis.length; i++) {

try {

const api = this.apis[this.currentAPI];

console.log(`${api.name} API 시도 중...`);

const response = await fetch(api.url);

if (!response.ok) {

throw new Error(`HTTP ${response.status}`);

}

const data = await response.json();

// API별 데이터 형식 통일

const normalizedData = this.normalizeData(data, api);

// 성공 시 로컬스토리지에 백업 저장

this.saveBackupData(normalizedData, api.name);

return normalizedData;

} catch (error) {

console.warn(`${this.apis[this.currentAPI].name} API 실패:`, error);

this.currentAPI = (this.currentAPI + 1) % this.apis.length;

if (this.currentAPI === 0) {

// 모든 API 실패 시 백업 데이터 사용

return this.getBackupData();

}

}

}

}

normalizeData(data, api) {

// API별로 다른 응답 형식을 통일된 형식으로 변환

if (api.name === 'ExchangeRate-API') {

return {

base: 'USD',

date: data.date,

rates: data.rates,

source: api.name

};

} else if (api.name === 'Fixer.io' && api.needsConversion) {

// EUR 기준을 USD 기준으로 변환

const eurToUsd = data.rates.USD;

const convertedRates = {};

for (const [currency, rate] of Object.entries(data.rates)) {

if (currency !== 'USD') {

convertedRates[currency] = rate / eurToUsd;

}

}

convertedRates.EUR = 1 / eurToUsd;

return {

base: 'USD',

date: data.date,

rates: convertedRates,

source: api.name + ' (converted)'

};

}

return data;

}

saveBackupData(data, source) {

const backup = {

data: data,

timestamp: Date.now(),

source: source

};

localStorage.setItem('exchange-rate-backup', JSON.stringify(backup));

}

getBackupData() {

const backup = localStorage.getItem('exchange-rate-backup');

if (backup) {

const parsed = JSON.parse(backup);

// 백업 데이터가 24시간 이내인 경우만 사용

if (Date.now() - parsed.timestamp < 24 \* 60 \* 60 \* 1000) {

console.log('백업 데이터 사용:', parsed.source);

return parsed.data;

}

}

// 백업 데이터도 없으면 기본값 반환

return this.getDefaultRates();

}

getDefaultRates() {

return {

base: 'USD',

date: new Date().toISOString().split('T')[0],

rates: {

KRW: 1342.50,

EUR: 0.92,

JPY: 148.35,

CNY: 7.23

},

source: 'Default (Offline)',

isDefault: true

};

}

}

// 사용 예시

const exchangeService = new ExchangeRateService();

```

---

## 📅 4. 개발 일정 (4주 계획) - API 변경 반영

### 1주차: 기반 구조 + API 연동 테스트

- \*\*Day 1-2\*\*: HTML 구조 설계 및 구현, ExchangeRate-API 테스트

- \*\*Day 3-4\*\*: CSS 기본 스타일링, API 응답 데이터 화면 표시

- \*\*Day 5-7\*\*: 반응형 디자인, CORS 해결, API 에러 처리

### 2주차: JavaScript 기능 + 데이터 관리

- \*\*Day 1-2\*\*: 환차손 계산, 입력 검증, 로컬스토리지 시스템

- \*\*Day 3-4\*\*: 과거 데이터 수집, 백업 및 복원 기능

- \*\*Day 5-7\*\*: 에러 처리, 로딩 인디케이터, 접근성 기능

### 3주차: 차트 구현 + 고급 기능

- \*\*Day 1-2\*\*: Chart.js 환율 차트, 과거 데이터 시각화

- \*\*Day 3-4\*\*: 이동평균선 분석, 트렌드 예측 로직

- \*\*Day 5-7\*\*: 인쇄 기능, 성능 최적화

### 4주차: 테스트 및 배포 준비

- \*\*Day 1-2\*\*: 단위 테스트, 통합 테스트, 다중 API 테스트

- \*\*Day 3-4\*\*: 사용자 테스트, 피드백 반영

- \*\*Day 5-7\*\*: 배포 환경 설정, 문서화, 최종 QA

---

## 📋 5. 최종 체크리스트

### API 관련 확인사항

- [ ] ExchangeRate-API 계정 생성 완료

- [ ] API 키 발급 및 테스트 완료

- [ ] 백업 API 계정 준비

- [ ] CORS 문제 해결 방법 확정

- [ ] API 호출 제한 모니터링 시스템 구현

### 개발 환경 확인사항

- [ ] 로컬 서버 설정 완료

- [ ] 개발자 도구에서 API 호출 테스트 성공

- [ ] 네트워크 오류 시 백업 시스템 작동 확인

- [ ] 오프라인 상황에서 로컬 데이터 표시 확인

---

\*\*🎉 API 변경에 따른 개발계획서 업데이트 완료!\*\*

\*\*주요 변경사항:\*\*

- ✅ 한국은행 API → ExchangeRate-API로 변경

- ✅ 다중 API 백업 시스템 추가

- ✅ CORS 문제 해결 방안 제시

- ✅ API별 데이터 형식 통일 시스템

- ✅ 에러 처리 및 백업 데이터 시스템 강화

\*최종 업데이트: 2024년 1월 (API 변경 반영)\*