핀테크 산업

2022 핀테크 인력 양성 교육

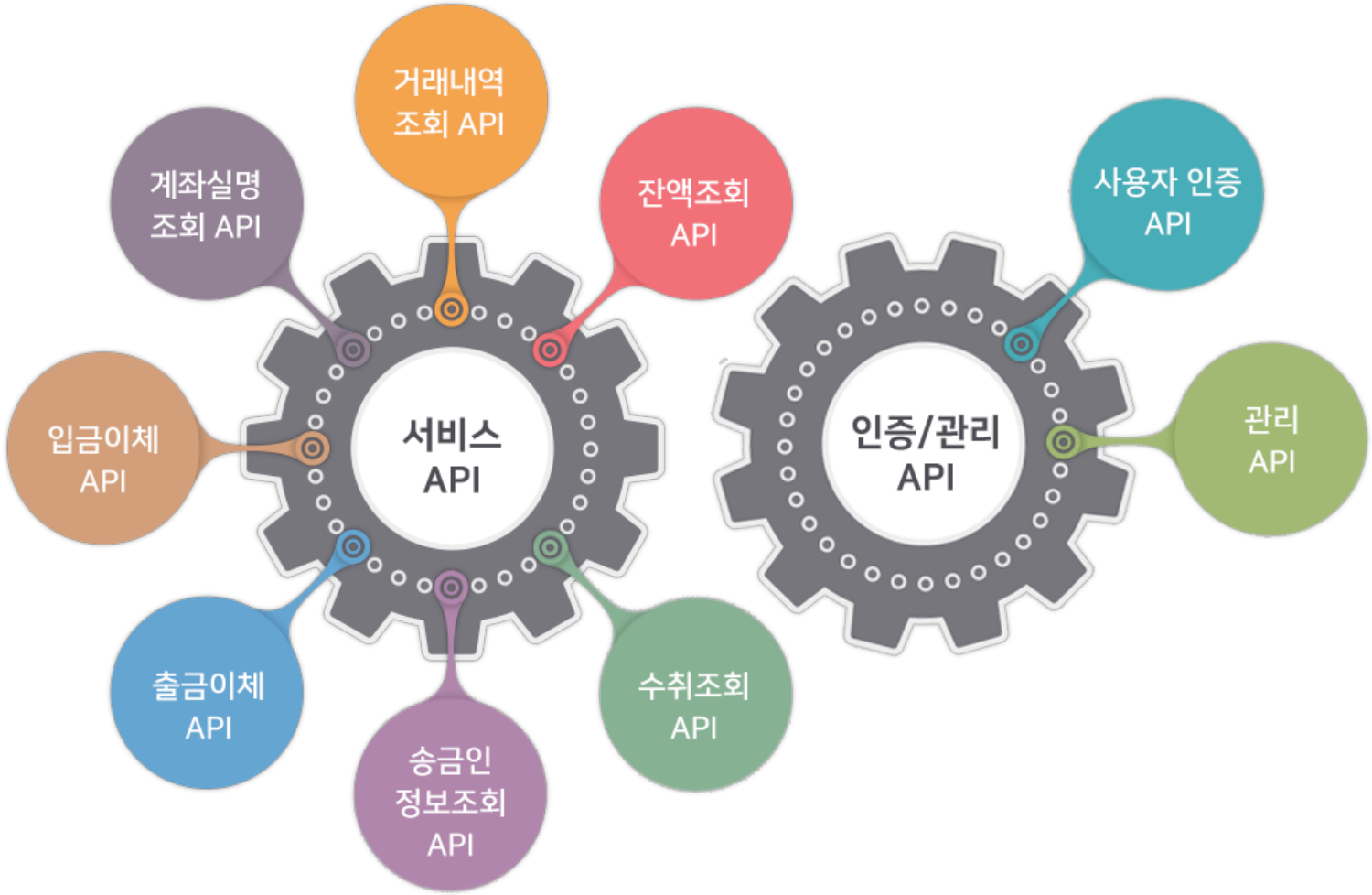
오픈뱅킹 API의 이해

오픈뱅킹(Open Banking)

- 핀테크기업이 금융서비스를 편리하게 개발할 수 있도록 은행 등 참가기관의 금융서비스를 표준화된 형태로 제공하는 인프라
- 오픈 API(Application Programming Interface) : 핀테크기업이 직접 응용프로그램과 서비스를 개발할 수 있도록 공개된 프로그램 도구로서, 7개의 서비스 API와 인증/관리 API 제공
- 테스트베드 : 개발된 서비스가 금융전산망에서 정상적으로 작동하는지 시험해 볼 수 있는 인프라
- 오픈 API와 테스트베드를 활용하여 기존 금융서비스에 새로운 IT기술을 접목시킨 다양한 핀테크 서비스를 빠르고 편리하게 출시가능


이용기관
(핀테크기업)
간편결제
간편송금
P2P사업
자산관리
오픈API
오픈뱅킹센터
(금융결제원)
잔액조회
거래내역조회
계좌실명조회
송금인 정보조회
수취조회
입금이체
출금이체
금융공동망
참가기관
A은행
B은행
○○은행

제공 API



https://www.openbanking.or.kr/apt/content/openapi

관련 법률

- 금융실명법, 전자금융거래법, 개인정보보호법
- 잔액조회, 거래내역조회 API: 금융실명법상 전자서명, ARS등을 통한 고객의 정보제공 동의 필요
- 출금이체 API: 전자금융거래법에 의거 고객의 출금동의 필요(공인인증, ARS 동의 채널 제공)

오픈뱅킹 동향(21.12.20 기준)

- 순가입자 3천만명, 순등록계좌수 1억개 돌파(경제활동인구 대비 105%)
- 누적 거래량: 83억 8천만건, 매일 2천만건, 1조원의 거래
- 잔액조회(68%), 출금이체(21%), 거래내역 조회(6%) 등 계좌 관련 기능 이용 비중 높음
- 카드, 선불정보 관련 기능의 이용량 꾸준히 증가 중
- 참여기관(총 120개 사)
    - 금융회사: 은행(19), 상호금융(7), 금투사(18), 카드사(8)
    - 핀테크기업: 대형(51), 중소형(17)

서비스 성과

- 금융 인프라 전면 개방: 폐쇄적 인프라 개방을 통한 제도적 혁신
- 지속적인 고도화: 확장성, 형평성, 안정성 관점의 고도화 실시
- 소비자 편익: 새로운 금융서비스와 금융소비자 편익의 증가
    - 금융소비자-새롭고 다양한 금융서비스 이용
    - 금융회사-간편송금, 자산관리 등 종합금융 서비스 제공
    - 핀테크기업-서비스 확장, 아이디어를 접목한 특화서비스 출시

향후 계획

- 오픈 파이낸스(Open Finance)로의 발전을 위한 향후 추진 방향 마련
- 여타업권(보험, 대출, ISA 등), 상품 추가 등을 통한 포괄적 확장
- PaaS(Platform as a Service) 지향
- 보안성 강화를 위한 Zero Trust 개념 도입
- 사전 및 사후 보안점검 체계화
- 백업센터 운영 등을 통한 데이터 복원력 확보
- 이용자 편익 증진을 위한 시스템 개선 추진
    - 출금이체 전 잔여이체한도 확인 기능 등

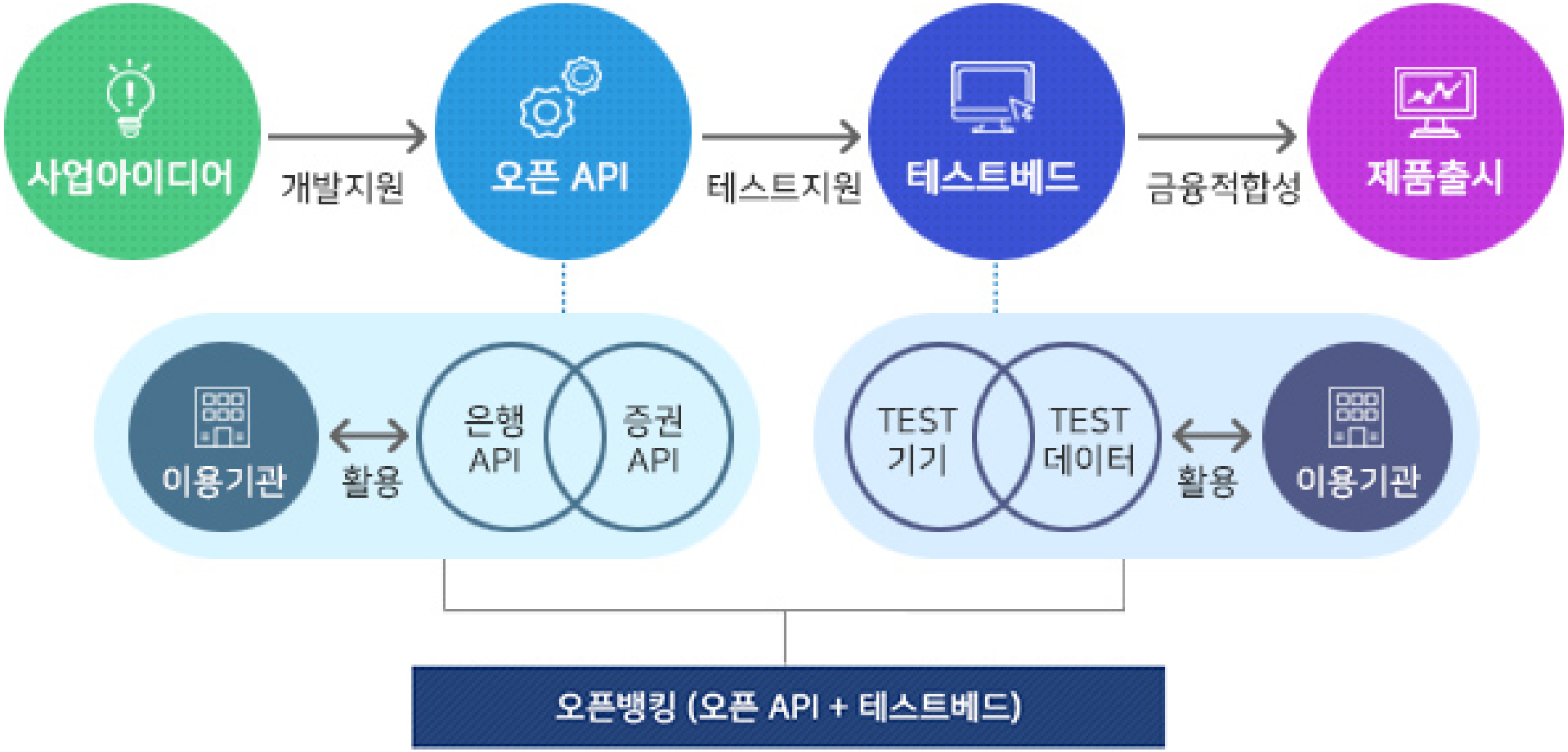
오픈뱅킹 API 적용 사례

- 해외 소액송금, 간편결제, 교환권 거래, 본인확인 서비스 등 다양한 분야에서 활용 중
- https://www.openbanking.or.kr/apt/content/showcase

오픈뱅킹 이용 절차

1. 금융결제원에 오픈뱅킹 이용신청
2. 이용적합성 심사와 승인
3. 서비스 개발과 시험
4. 금융보안원 등의 이용기관 보안점검과 취약점 점검 실시
5. 이용계약 체결과 오픈뱅킹 이용

오픈뱅킹 API를 활용한 개발 프로세스



출처 : 금융위원회 발표자료(2015. 7월)

HTTP

HyperText Transfer Protocol

- www상에서 정보를 주고받는 프로토콜
- TCP, UDP를 활용함
- HTTP method: GET, POST, PUT, DELETE

HTTP over SSL

- HTTP의 보안 강화 버전
- 소켓 통신 시 SSL or TLS Protocol로 세션 데이터를 암호화
- default port: 443

Web Programming

웹 개발 패턴의 변화

- 1991 ~ 1999: Sir Timothy John "Tim" Berners-Lee가 하이퍼텍스트 기반의 프로젝트를 제안한 이후 정적인 컨텐츠들을 중심으로 한 웹 기술이 발달
- 1999 ~ 2009: Linux, Apache, Mysql, Php 중심의 동적인 서버, 정적인 클라이언트 모델이 지속됨
- 2010 ~ 현재: javaScript!! (Dynamic Web Client)

웹 개발 패턴의 변화

```
<html>
<head></head>
<body>
<h1>Static Header</h1>
<div>Static Contents</div>
</body>
</html>
```

- 1991 ~ 1999: Sir Timothy John "Tim" Berners-Lee가 하이퍼텍스트 기반의 프로젝트를 제안한 이후 정적인 컨텐츠들을 중심으로 한 웹 기술이 발달

웹 개발 패턴의 변화

```
<html>
<head></head>
<body>
<h1>{% Dynamic Header %}</h1>
<div>{% Dynamic Contents %}</div>
</body>
</html>
```

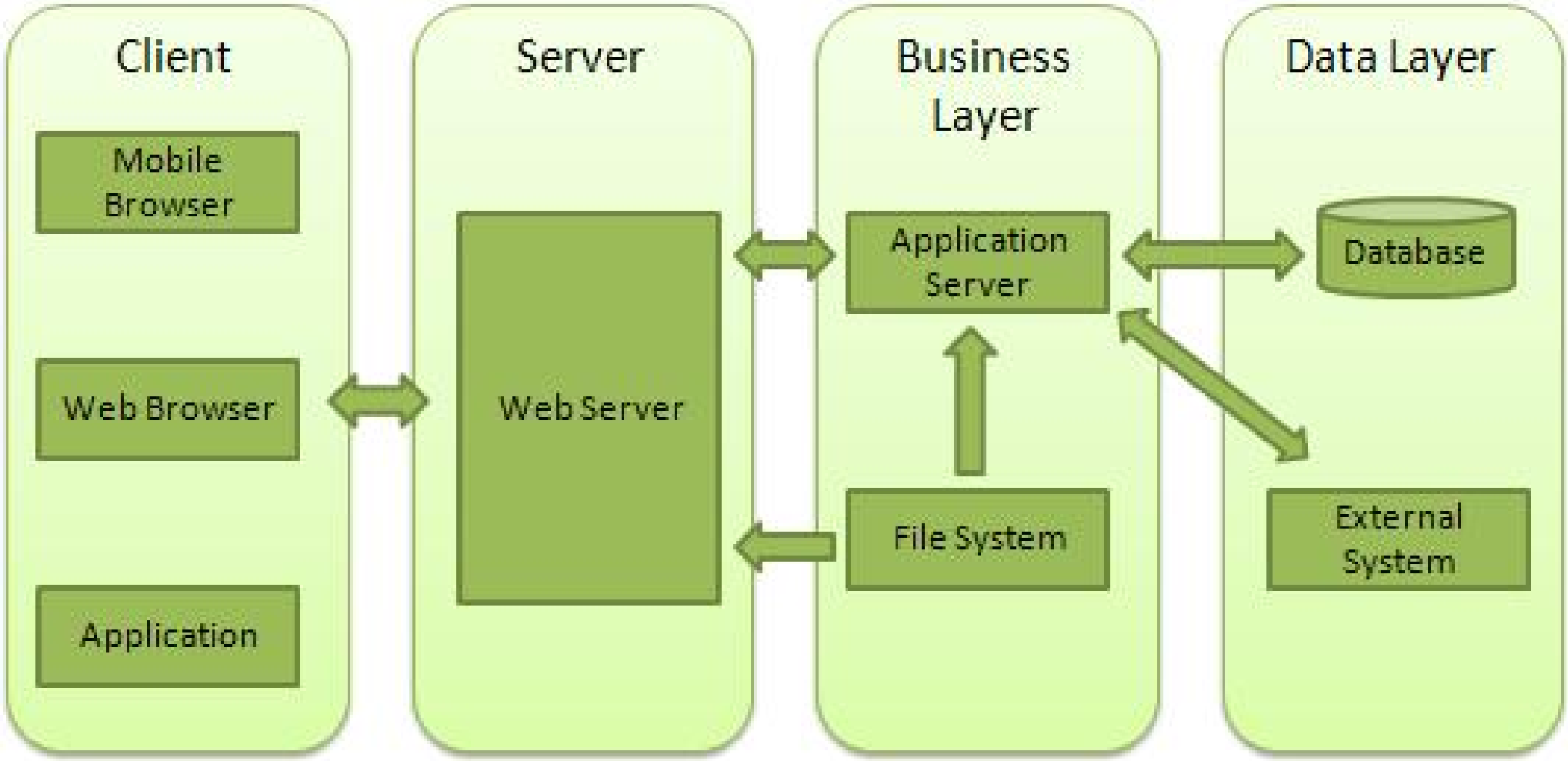
- 1999 ~ 2009: Linux, Apache, Mysql, Php 중심의 동적인 서버, 정적인 클라이언트 모델이 지속됨

웹 개발 패턴의 변화

```
<html>
<head>
<script src="https://unpkg.com/vue"></script>
</head>
<body>
<h1>{{ header }}</h1>
<div id="app">
  {{ message }}
</div>
<script>
var app = new Vue({
  el: '#app',
  data: {
    message: '안녕하세요 Vue!'
  }
})
</script>
</body>
</html>
```

- 2010 ~현재: javaScript!!(Dynamic Web Client)

Web architecture



Client-side

- HTML/CSS, javaScript
- jQuery, AJAX
- Front-end Web Framework
  - Angular
  - React.js
  - Vue.js
- CSS Framework
  - Bootstrap
  - Foundation

Server-side

- Depends on Language
  - PHP: Laravel
  - javaScript: Node.js(Express.js)
  - Java: Spring
  - C++, C#: ASP.net
  - Python: Django, Flask
  - Golang: itself
  - Ruby: Ruby on Rails

Database

- RDBMS
  - MySQL
  - PostgreSQL
  - MariaDB
- noSQL
  - MongoDB
  - CouchDB
  - Redis

etc

- celery (for Distributed Task Queue)
- github, Bitbucket, gitlab (for SCM)
- travis CI or jenkins (for Continuous Integration)
- slack, trello
- Sentry(for Exception monitoring)