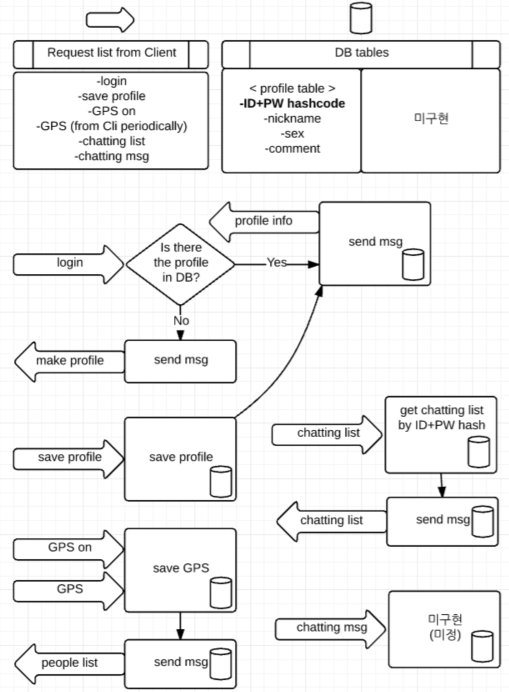
모바일 클라우드 최신기술

상세 설계 계획서

20103374 이창우

<Server>



서버쪽에서는 클라이언트의 요구가 무엇인지 받는다. 오른쪽으로 향하는 화살표가 클라이언트에게서 온 요구사항이고 왼쪽으로 향하는 화살표는 서버가 클라이언트에게 보내는 응답이다. DB를 사용하는 부분에서는 그래프 안에 DB 아이콘이 포함되어 있다.

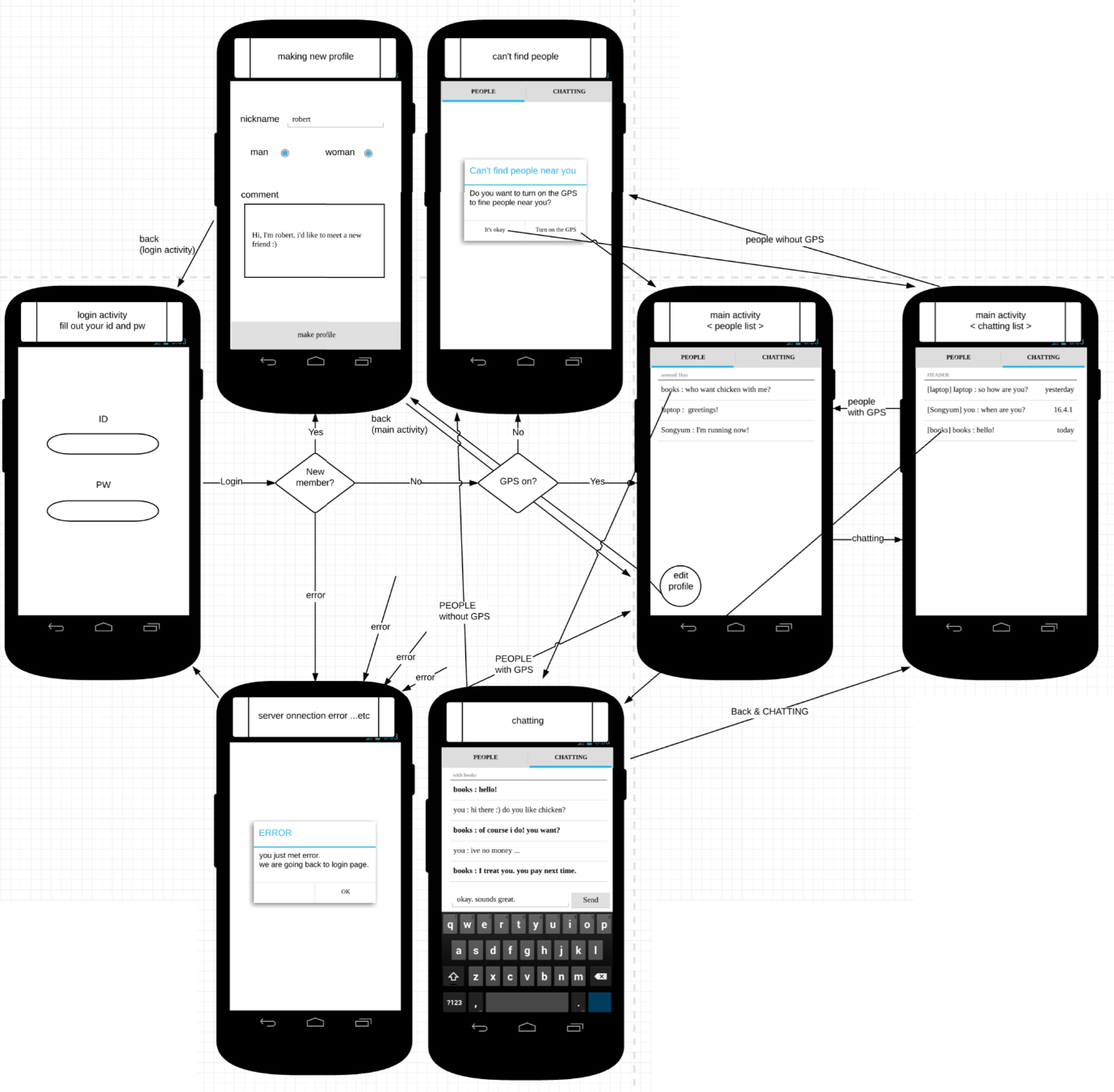
서버는 아마존 ec2 인스턴스에 코드 이그나이터를 설치해 사용하며 mySQL을 기본 DB로 사용한다. Php 페이지마다 각각 다른 종류의 서비스를 제공한다. 클라이언트쪽에선 필요로 하는 서비스를 제공하는 서버 php 페이지로 접속하면 된다.

~~채팅에 관한 모든 정보는 서버쪽에서 저장하지 않는다. 서버는 단지 메시지를 서로에게 보내줄 뿐이다.~~ 생각해보니 클라이언트에서 ID,PW가 다르면 채팅 내용도 달라져야 하잖아 ?? 그러므로 서버는 저장 기능을 가져야 한다. 저장할 때는 채팅 한 사람의 ID/PW의 hash 값을 기준으로, String 타입으로 저장한다. 클라이언트에서 저장된 것을 꺼낼 때는 String 그대로 주며 클라이언트는 파싱하여 리스트에 뿌려주어 채팅 내역을 회복한다.

깃허브에는 DB 정보가 존재하지 않는 php 페이지만 업로드 한다.

젠킨스 서버는 코드 이그나이터와 별개로 따로 만든다.

< Client >



**loginActivity.class** // 로그인 화면

-성공하면 다음 화면으로 넘어감

-실패하면 프로필 새로 만드는 화면으로 넘어감

-에러나면 에러 화면으로 넘김

**errorActivity.calss** // 에러 화면

-에러 화면. 경고창을 띄운 후 전 액티비티로 돌아가거나 로그인 화면으로 강제 변경

-대개는 서버 에러때문에 오게 됨.

**newProfileActivity.class** // 프로필 새로 만드는 화면

-새로운 유저일 때

-save 없이 back 누르면 그 전 액티비티로 돌아감

-스태틱매니저에서 관리되어짐.

**networkThread.class** // 네트워크 연결 쓰레드

-네트워크 컨트롤 옵션들을 담은 쓰레드. 이 쓰레드를 통해 http 통신을 함

-스태틱매니저에서 관리되어짐.

**viewHandlerThread.class** // GUI를 변경하기 위한 쓰레드

-뷰를 수정하거나 바꾸기 위함.

-스태틱매니저에서 관리되어짐.

**staticManager.class** // 스태틱 매니저

-저장하거나 유지되고 있어야 할 정보들을 담고 있음.

-껏다 켜져도 유지될 수 있도록 저장함.

-이를 테면, 로그인 여부나 네트워크, UI 핸들러, 저장, 기타 쓰레드 등

**mainActivity.class** // 피플 리스트와 채팅 리스트를 담고 있는 탭 액티비티

**peopleListActivity.class** // 사람들 보여주는 화면

-서버로부터 사람들 정보를 받아서 보여줌.

-저장하는 정보 없음.

-화면이 reload 되면 서버에서 무조건 다시 받아옴.

**chattingListActivity.class** // 채팅 내역을 보여주는 화면

-채팅한 내역을 보여줌.

-채팅 내역은 어플이 꺼져도 살아있을 수 있도록 저장함.

-각 채팅방마다 각각의 쓰레드가 돌아가도록 함.

**chattingActivity.class** // 채팅하는 화면

-스크롤뷰 위에 리스트와 에딧텍스트, 버튼 세 가지만 놓여있음

-받고 보내는 이벤트가 일어날 때 마다 서버와 http 연결함.

-채팅 내용은 어플이 꺼져도 살아있을 수 있도록 저장함.

안드로이드에서 테스트 해야 하는 것

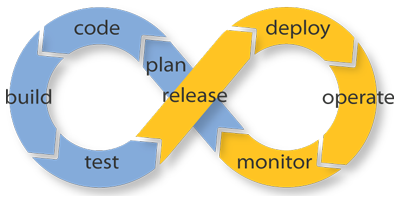
-액티비티 생명 주기 이벤트. onPause나 onCreate 등이 불렸을 때를 임의로 가정하여 상태 변경이 정확히 이루어지는지 테스트  
-Configuation. 현재 액티비티를 다시 생성하는 configuration 변경 이벤트.  
-DB와 FileSystem 연산. 정확하게 수행되는지 반드시 테스트.  
-디바이스의 물리적 특성. 어플 배포 전 모든 디바이스에서 잘 동작하는지, 현재 디바이스 환경을 알아내서 그에 맞게 동작하는지 확인.  
 네트워크 환경, 화면 밀도, 화면 해상도, 화면 크기, 센서 여부, 키보드와 입력 장치, GPS, 외장 메모리 등  
-TDD 공부해서 적용하기. TDD를 하려면 OOP가 기본이고, 액티비티의 메소드에는 UI 없이 오로지 논리적인 기능만 있어야 함.  
-mock Object 를 통해서 네트워크에 대한 유닛테스트를 할 수 있음.

-어? 테스트 하는 클래스를 하나 만들어?

상세 일정표

|  |  |  |
| --- | --- | --- |
| 4.25-31 | 각 액티비티 별 프로토 타입 구축 MainActivity의 프래그먼트 구조 먼저 이해. 기본적인 GUI 프로토타입 만듦. 서버의 DB안에 데이터(ID/PW hash value, profile data, 그 외에 저장이 필요한 데이터들) 저장하는 부분 구현(테스트 해가며 구현) | 젠킨스 서버 구축 및 깃허브와 연동 윤교수님께 면담 및 코드 유닛 테스트 |
| 5.1-6 | 액티비티 내용 구현과 그에 따른 서버 구현 어플 내에서 프로필 만들고 저장하는 부분 구현-로그인 정보 네트워크 쓰레드 클래스 구현-쓰레드 StaticManager 클래스 만들어서 ‘로그인 정보’와 ‘쓰레드’ 공유 (로그인 및 프로필) | ~~JIRA 활용하여 프로젝트 관리(학교 재정지원 받기)~~ 깃허브의 issue 기능을 대신 사용 OOP 개념 도입하여 구축 TDD는 이걸 해야할지 말아야할지 아직 모르겠으나 되도록 해보는 것이 경험상 좋을 것임 윤교수님께 면담 및 코드 유닛 테스트 |
| 5.9-13 | 액티비티 내용 구현과 그에 따른 서버 구현 위치 기반 서비스를 이용하여 서버에 내 위치 저장하는 부분 구현. 위치 기반 서비스와 그에 따른 소스들 이해하고 메커니즘, 사용법 공부하기 내 위치 기반으로 원거리에 있는 사람들 정보를 받아서 리스트로 만듦. 리스트 GUI 프로토타입 만들기 (사람들 리스트) | 윤교수님께 면담 및 코드 유닛 테스트 |
| 5.16-20 | 액티비티 내용 구현과 그에 따른 서버 구현 채팅 쓰레드 클래스 만들고 StaticManager에 넣어 유지 채팅 기능 구현하기(뷰, alarm, 서버와의 프로토콜 등) 서버에서 채팅 로그 기록하는 부분 구현 더불어 양쪽 클라이언트에게 서로의 메시지를 보내주는 프로토콜 구현 (채팅 리스트) | 윤교수님께 면담 및 코드 유닛 테스트 |
| 5.23-27 | 액티비티 내용 구현과 그에 따른 서버 구현 액티비티 간 이동, 지저분한 구현 구조 개선 어플의 전반적인 안정성 향상을 위해 사용해보면서 튕기거나 버그가 있는 부분 찾아서 수정  (채팅 시스템 및 버그 수정) | 사람들에게 피드백 받음 윤교수님께 면담 및 코드 유닛 테스트 |
| 5.30-6.3 | 버그 수정 및 피드백 결과 반영 및 UI 수정  사용자 편의에 맞게 GUI 수정하기 | 윤교수님께 면담 및 코드 유닛 테스트 |

continuous integration을 위한 사이클



앞으로 해야 할 것:  
-안드로이드 명명 규칙 따라 구현하기  
-Junit 사용해야지. Junit 사용법 공부하고 적용하자! TDD.

※github : https://github.com/eyeballss/161KMUMobileCloudClass.git

※각각의 액티비티 기능별로 유닛 테스트를 모두 함. 테스트 한 후에야 기능을 합치도록 함. 그래서 유닛 테스트를 위한 별도의 프로젝트와 통합 프로젝트를 구분하여 진행해 나가는 것도 괜찮은 방법일 듯.

※Atlassian JIRA : 버그 트랙킹 시스템에서 시작해서, issue나 Issue 기반의 전체적인 프로젝트 관리를 할 수 있게 하는 툴. 요금은 학교에 청구했으면 함.

※윤교수님께 면담 한 내용은 그때 그때 반영함.

※각각의 액티비티는 어떤 액티비티에서 왔는지 알고 있어야 함.

※모든 그래프는 lucidchart([www.lucidchart.com](http://www.lucidchart.com))를 이용하여 그리고 있음. 아래는 내가 설계서를 만들기 위해 사용중인 문서의 공유 링크.

<https://www.lucidchart.com/invitations/accept/fdb8a2f0-74e4-4e55-bbc1-2c473004d69e>

※계속해서 업데이트 됩니다. 구현을 계속 진행해 나가면, 수정되는 부분이 많을 겁니다.