# Architektur

Als erstes wird analysiert welche Kommunikationsmethode für das System verwendet werden sollte. Anschließend wird ermittelt welche Komponenten benötigt werden und das Ganze in einem Diagramm zur besseren Veranschaulichung dargestellt.

## Abwägung der Kommunikationsmethode

**Peer-to-Peer**

Bei Peer-to-Peer werden die Systemnutzer direkt miteinander verbunden, ohne Umwege über einen Server. Hierdurch werden kosten für die Bereitstellung und Wartung des Servers reduziert, da jedes Gerät im Netzwerk seine Ressourcen zur Verfügung stellen und teilen kann. Diese Methode ist jedoch nur für eine geringe Anzahl von Nutzern ausgelegt, die sich oder die Adresse des Geräts auch noch kennen und gezielt miteinander verbinden müssen. Diese Methode wäre für unser System nicht nützlich, da Inserate immer verfügbar sein sollen, selbst wenn sich der Anbieter nicht verbunden hat. Außerdem fehlen eine zentrale Verarbeitung und Speicherung von Daten. Unser angestrebtes System wäre durch die Verwendung von Peer-to-Peer nicht realisierbar.

**Client/Server**

Die Verwendung der Client/Server Methode bietet unserem System die Möglichkeit einer flexiblen Systemstrukturierung und ist für eine beliebig große Menge an Nutzern anwendbar. Durch die zentrale Verwaltung von Ressourcen wird es uns ebenfalls ermöglicht die gewohnten Sicherheitsstandards umzusetzen und den gesetzlichen Datenschutz einzuhalten. Serversysteme sind in der Regel leicht erweiterbar und ermöglichen das System jederzeit zu vergrößern und mit Funktionen zu erweitern, was es zukunftssicher und anpassbar an den Markt macht. Negativ anzumerken sind vor allem der hohe Aufwand für die Einrichtung der zusätzlichen Hardware, zudem könnte ein Angriff auf die Komponente das gesamte System gefährden. Trotzdem überwiegen die positiven Aspekte dieser Methode, vor allem die flexible Möglichkeit zur Erweiterung und der Sicherheitsstandards.

# System-Komponenten

### Server

Der Server wird primär für die Speicherung von Nutzerdaten und Inseraten verwendet. Nutzer sollen jederzeit zeitgleich mit ihm interagieren können, weshalb er möglichst alle Anfragen gleichzeitig und in Echtzeit bearbeiten sollte. Erstellte Anzeigen und die dazugehörigen Daten sollen in einer Datenbank gespeichert und mit weiteren Daten automatisch angereichert werden. Um zu gewährleisten, dass der Server trotz steigender Nutzeranzahl weiterhin einwandfrei funktioniert, muss auf eine gute Skalierbarkeit geachtet werden. Er stellt zusätzlich Verbindungen zu externen Diensten her, um Daten zu empfangen und zu verarbeiten oder zu verteilen. Um diese Funktionalität zu realisieren wurde sich auf die Verwendung von NodeJS geeinigt. Diese auf Javascript basierende Plattform bietet eine umfangreiche Dokumentation und ebenfalls eine große Auswahl an Funktionen und Diensten, die die Entwicklungszeit des Systems beschleunigen sollte. Außerdem wurde mit NodeJS bereits Erfahrung gesammelt, was zu einer schnelleren Entwicklungszeit führen könnte.

### Clients

Das System soll von zwei Kerngruppen genutzt werden, die jedoch unterschiedliche Ansprüche an das System und deren Aufgaben haben. Diese Clients werden in Anbieter und Suchender aufgeteilt. Für dieses Projekt wird sich lediglich auf ein Endprodukt fokussiert - eine mobile Anwendung, um eine einfache und effiziente Nutzung des Systems zu ermöglichen. Nutzer sollen ihre Rolle beliebig ändern können, dazu müssen sie sich registrieren, um die volle Funktion des Systems nutzen und Inserate erstellen zu können. Weitere Vorteile durch die Verwendung einer mobilen Anwendung sind die Einbindung der Standortdaten und die Funktion der Kamera, hierdurch wird erleichtert attraktive Angebote zu erstellen.

### Middleware

Um den Nutzer weiteren Komfort zu bieten, sollen externe Dienste eingebunden werden. Hier muss jedoch entschieden werden über welche Schnittstelle dies passieren soll. Es wurde sich hierfür auf die Verwendung von REST geeinigt. Die REST-Schnittstelle ist auf HTTP beschränkt, bietet jedoch im Datenformat eine freiere Auswahl als andere Schnittstellen. Durch die Verwendung von NodeJS auf dem Server lassen sich externe Dienste leicht über das Framework "express" einbinden. Zudem wurden mit der Implementierung von REST bereits Erfahrung gemacht, womit die Einarbeitungs- und Entwicklungszeit verkürzt werden kann, was sich positiv auf das Projekt auswirkt.

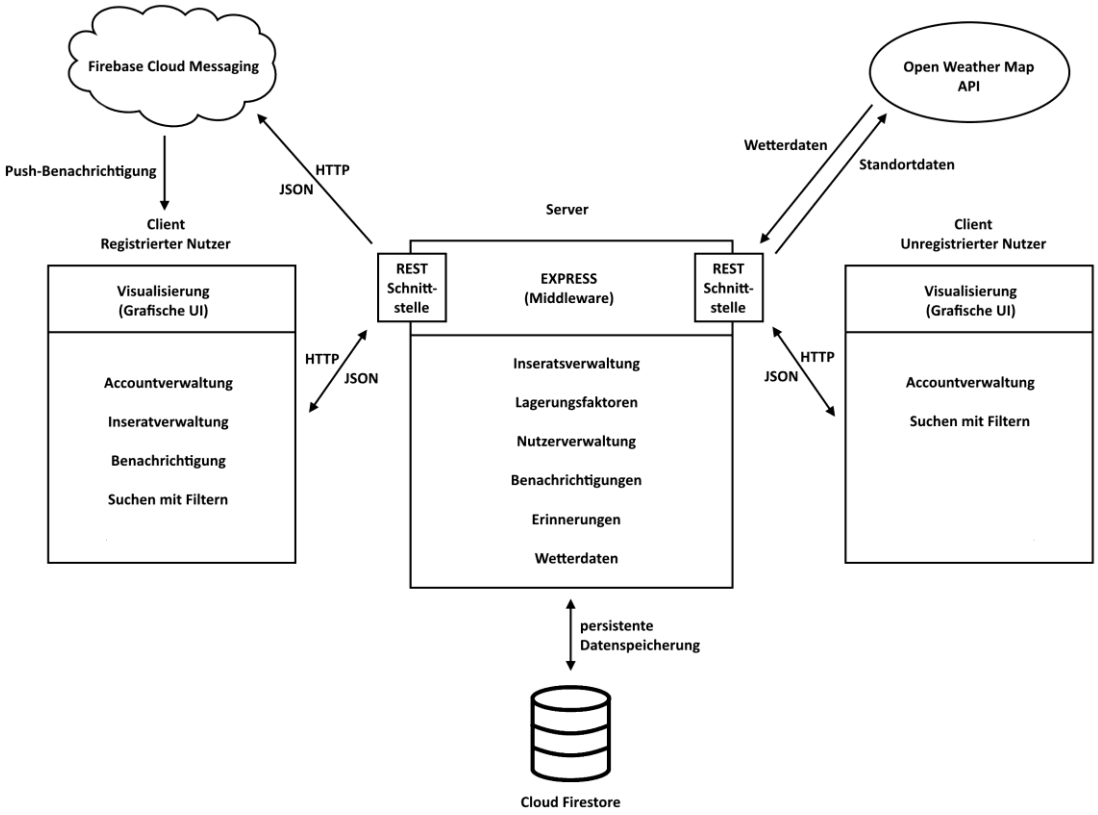
### Weitere Dienste

Im geplanten System ist angestrebt dem Nutzer die Möglichkeit zu geben, bestimmte Gruppen zu abonnieren, um über aktuelle Inserate benachrichtigt zu werden - also eine Art Publish/Subscribe-System. Es wurde sich hier für Firebase Cloud Messaging von Google entschieden. Es ermöglicht eine einfache Möglichkeit zur Versendung von Nachrichten und ist zudem noch performant und kann im Hintergrund arbeiten - sollte der Nutzer dies wünschen. Negativ anzumerken ist hierbei jedoch der Datenschutz. Es ist nicht offensichtlich wie viele und welche Daten Google analysiert und speichert. Ebenfalls wird ein Google-Konto benötigt, was die meisten Nutzer jedoch durch die Verwendung eines Android-Geräts beim Start bereits erstellt haben. Eine weitere Funktion im System soll die Einbindung von Wetterdaten sein. Der Nutzer wird bei stürmischem oder heißem Wetter gewarnt und nähere Inserate bevorzugt. Es wurde sich hier für die API von OpenWeatherMap entschieden, sie ist kostenlos nutzbar und bietet eine schnelle und einfache Möglichkeit Daten auszulesen. Für die Ermittlung der lokalen Inserate müssen Standortdaten erhoben werden. Das Android-System bietet auch hierfür ausreichende Mittel, um Standorte zu ermitteln, diese mit Stadtkoordinaten abzugleichen und zu berechnen.

## Datenformat und Datenbank

Inserate sollen mit all den zugehörigen Informationen auf dem Server in einer Datenbank gespeichert werden. Es soll sich ebenfalls vollständig wieder löschen lassen, sollte ein Abnehmer gefunden worden sein oder die Frist abgelaufen sein. Sie sollten ebenfalls editierbar sein, sollte sich der Anbieter entschließen mehr Informationen anzugeben, oder bestimmte Daten zu ändern. Es wird also angestrebt eine Datenbank zu realisieren die flexibel und schnell arbeitet. Dazu wurden NoSQL und SQL Datenbanken verglichen. NoSQL ist deutlich performanter, da hier Vorgänge angepasst werden können und nicht wie bei regulären SQL Datenbanken vorgeschriebene Abläufe stattfinden müssen. Weiterer wichtiger Punkt ist die Kompatibilität von Datenformaten. Da der geplante Server bereits NodeJS und Javascript verwenden soll, bietet es sich an JSON zu nutzen. Zuletzt muss nur noch ermittelt werden welche Datenbank implementiert werden sollte, die all diese Kriterien erfüllt. Als Lösung wurde der Cloud Firestore von Google gefunden. Da bereits für die Pub/Sub Dienste Firebase benutzt werden soll, bietet sich diese Datenbank an. Ebenfalls ist sie sehr flexibel und vor allem skalierbar.

# Architekturdiagramm




### Drei-Schichten-Architektur

Um das System möglich effizient zu gestalten, liegt die Idee nahe eine Mehrschichtenarchitektur zu implementieren. Dadurch ergibt sich eine übersichtliche Struktur und eine flexible Wartbarkeit. Die einzelnen Schichten können individuell bearbeitet werden, ohne die Funktion der anderen zu beeinträchtigen. Um eine sinnvolle Unterteilung dieser Schichten im System zu realisieren, können diese nach der Three-Tier Architecture gegliedert werden: Darstellung, Anwendung und Daten. Die Datenschicht wird genutzt für den Zugriff auf lokal gespeicherte Daten und den Zugriff auf die verwendete Datenbank, die Anwendungsschicht wird für die Nutzung, Verarbeitung und Anreicherung von Daten verwendet. Die Darstellungsschicht stellt die grafische Visualisierung dar und ist jene mit der der Nutzer interagieren soll. Diese befindet sich jedoch nur bei den Clients, da der Server diese nicht benötigt.