Software Architecture

For our project 3, we decided on a Client-Server model for our software architecture. We decided on this architecture in particular because our program made sense to be split up into a client side that handled all of the logic and user interface functionality, and a server side that contained a database and handled the storing and sharing of data for all users. As for other possible software architectures, in our case a 3-tier or N-tier architecture wouldn't really make sense, because the data that will be sent in between the client and server won't really need to be processed in any way before it reaches its destination. Also, a peer-to-peer architecture also wouldn't really make sense, as our program will rely on a single set of data that will need to be available to all systems when needed. Finally, a pipes-and-filters type architecture also didn't really make sense, as the data once again won't really need to be processed in between the client and server. Thus, a client-server architecture was the only one that really made sense, and it would work well with our chosen design paradigm, which was an object-oriented approach. With an object-oriented approach, the client and server could communicate with discrete objects that would encapsulate all the data that corresponds to a single entity, which in our case would be a lost pet. Meanwhile, the components of the client and server, such as the UI and database respectively, could also be their own objects. Overall, an object-oriented approach felt like a very natural and intuitive way to create our website, and it seemed to lend itself to a client-server architecture quite well.