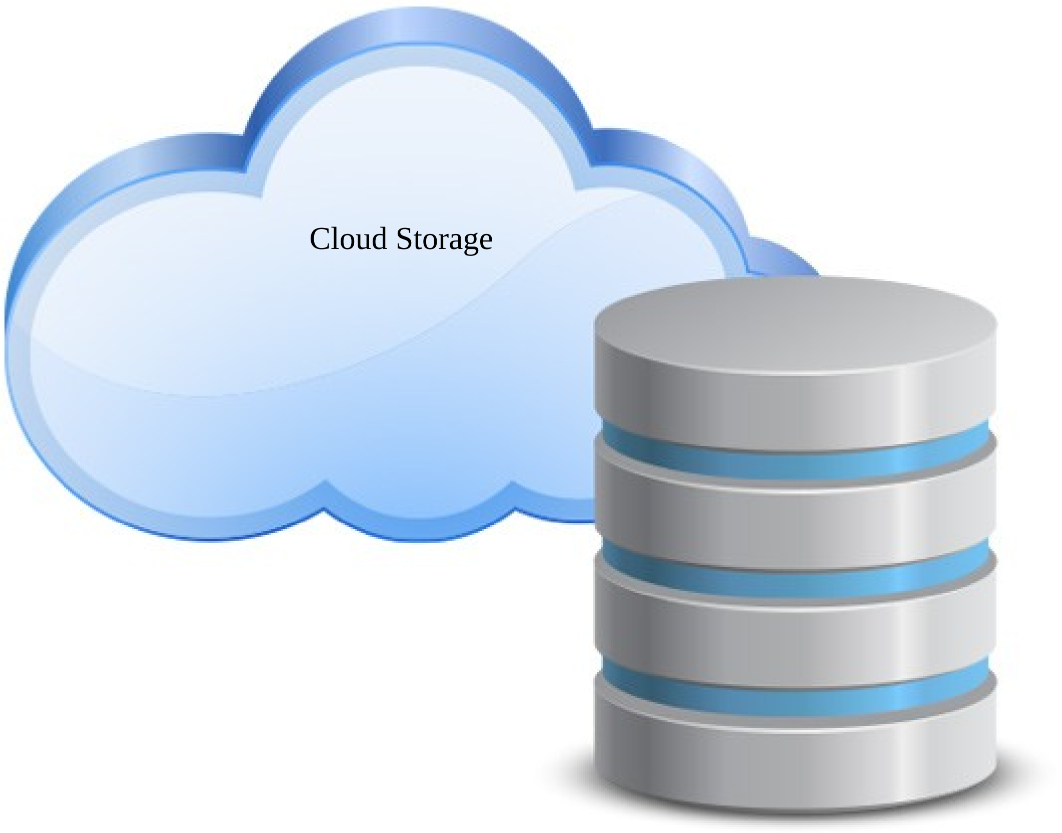
STORAGE

Base A Storage:
Have an Offline Database System.
Follow PCI DSS (Payment Card Industry Data Security Standard), encrypt the sensitive information and store them in a databases (PostgreSQL or MySQL) using the best and most secure encryption algorithms currently known (AES and Twofish). For example, PostgreSQL offers encryption at several levels, and provides flexibility in protecting data from disclosure due to database server theft, unscrupulous administrators, and insecure networks.

PostgreSQL encryptions include:
i. Password Encryption, ii. Encryption For Specific Columns, iii. Data Partition Encryption, iv. Encrypting Data Across A Network, v. SSL Host Authentication, vi. Client-Side Encryption.

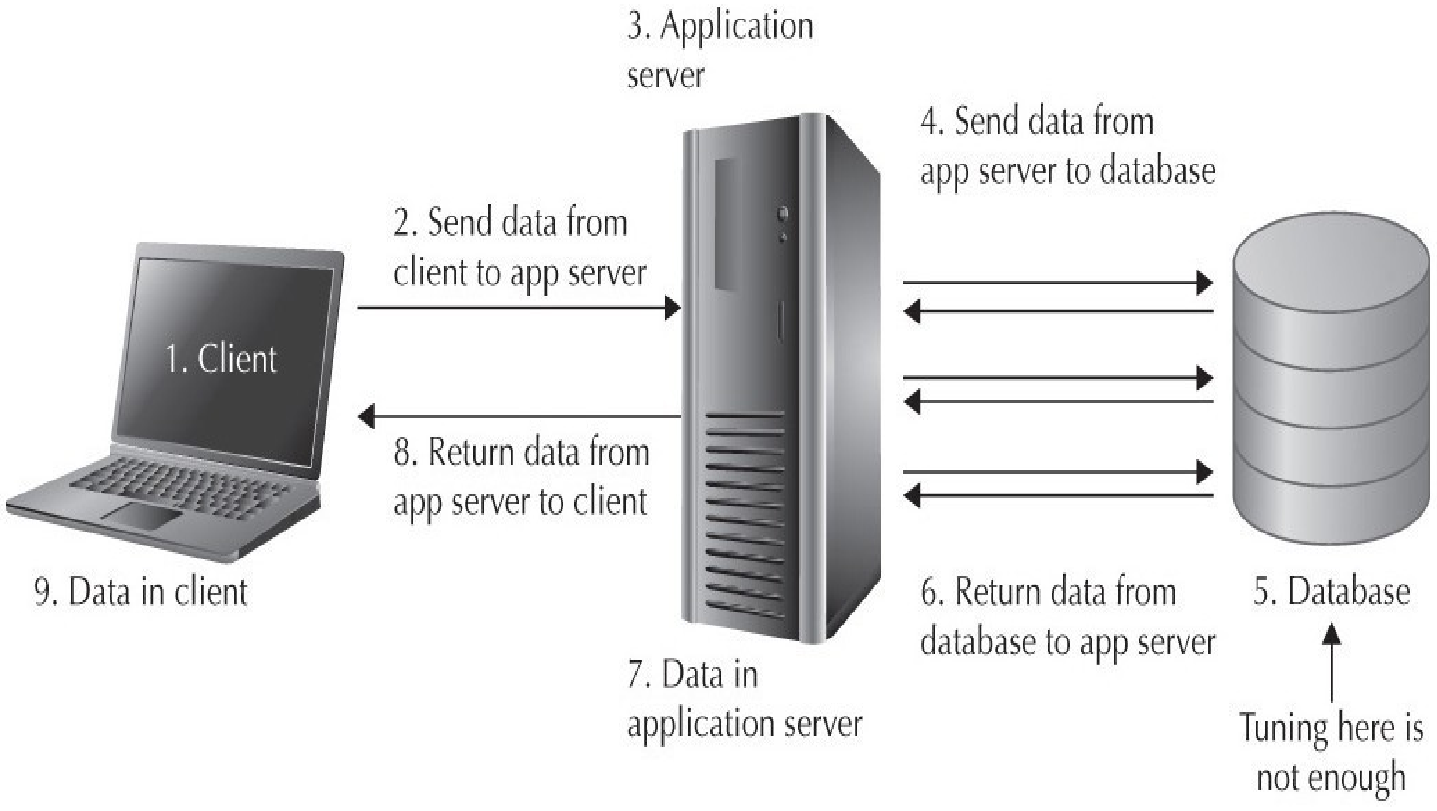
Also, the database system(s) should not be connected to the internet. If necessary, it should use intranet. Restrict physical access to the database system and to cardholder data to only authorize person.

Base B Storage:
Use third party services.
Third party services like AWS CLOUD STORAGE offers highly affordable long-term storage. Cloud storage is typically more reliable, scalable, and secure than traditional on-premises storage systems. Depending on customers need, integrating MasterCard services and APIs or third party validated PCI DSS compliant service provider might be helpful too.



Base C Storage:
Encrypt all the sensitive information, store them in a database(PostgreSQl or MySQL database), and follow the PCI DSS 12 requirements for implementation.

PCI DSS 12 Steps
1. Install and maintain a firewall configuration to protect data.
2. Do not use vendor-supplied defaults for system passwords and other security parameters.
3. Protect stored data.
4. Encrypt the transmission of cardholder data and sensitive information across public networks.
5. Use and regularly update antivirus software.
6. Develop and maintain secure systems and applications.
7. Restrict access to data by business need-to-know.
8. Assign a unique ID to each person with computer access.
9. Restrict physical access to cardholder data.
10. Track and monitor all access to network resources and cardholder data.
11. Regularly test security systems and processes.
12. Maintain a policy that addresses information security.


3. Application server
4. Send data from app server to database
2. Send data from client to app server
1. Client
8. Return data from app server to client
9. Data in client
6. Return data from database to app server
5. Database
7. Data in application server
Tuning here is not enough

PROCESSING

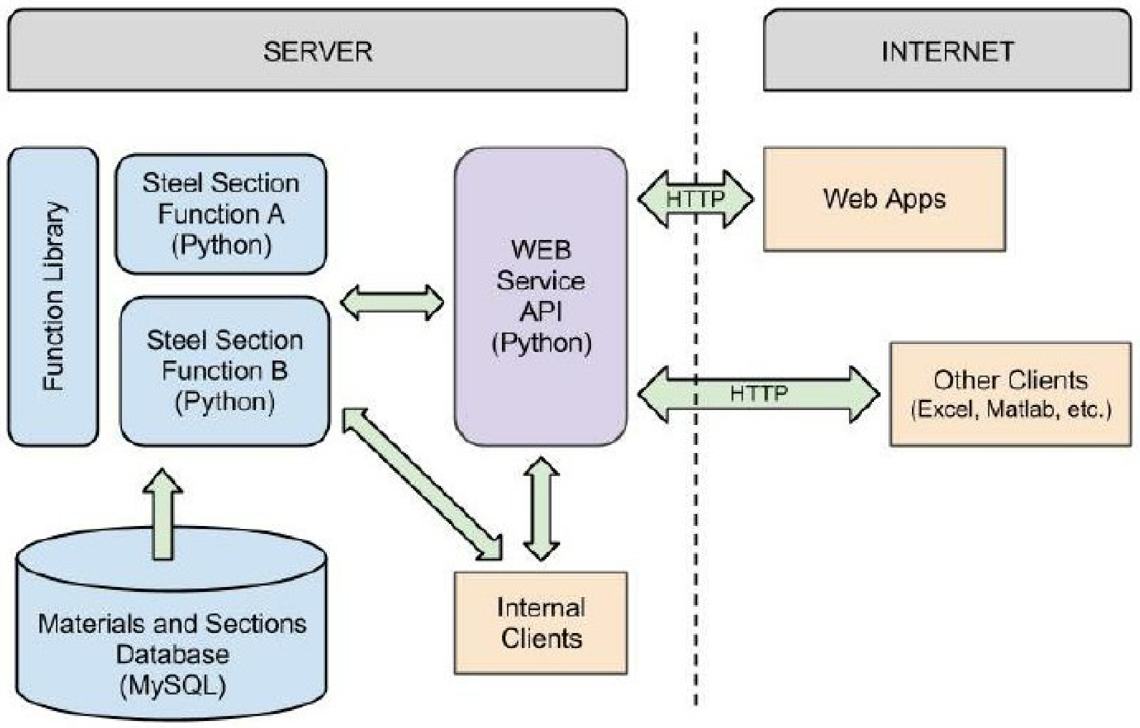
Use Python to process the datas. We can also implement APIs using Python, Django Rest Framework. Depending on the necessity we can use Django for backend. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design, and it takes security seriously.

It's important to note that, we also need to implement ElasticSearch for easy search and retrievval of information. Python Requests will be useful here for APIs calls.

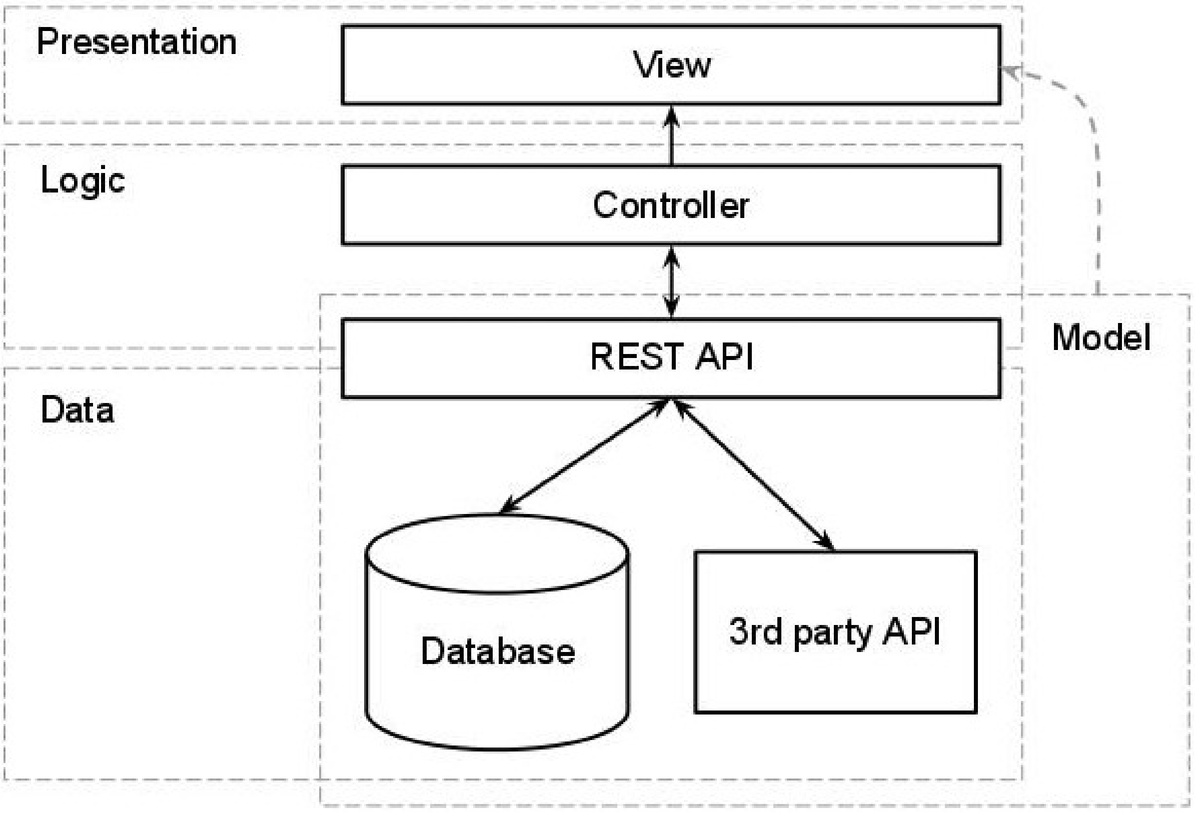
Technology:
1. Python
2. SQL and NoSQL databases (PostgreSQL, MySQL, Redis etc)
3. AWS Cloud Storage
4. ElasticSearch
5. Django Rest Framework
6. Django
7. Requests

ARCHITETURES

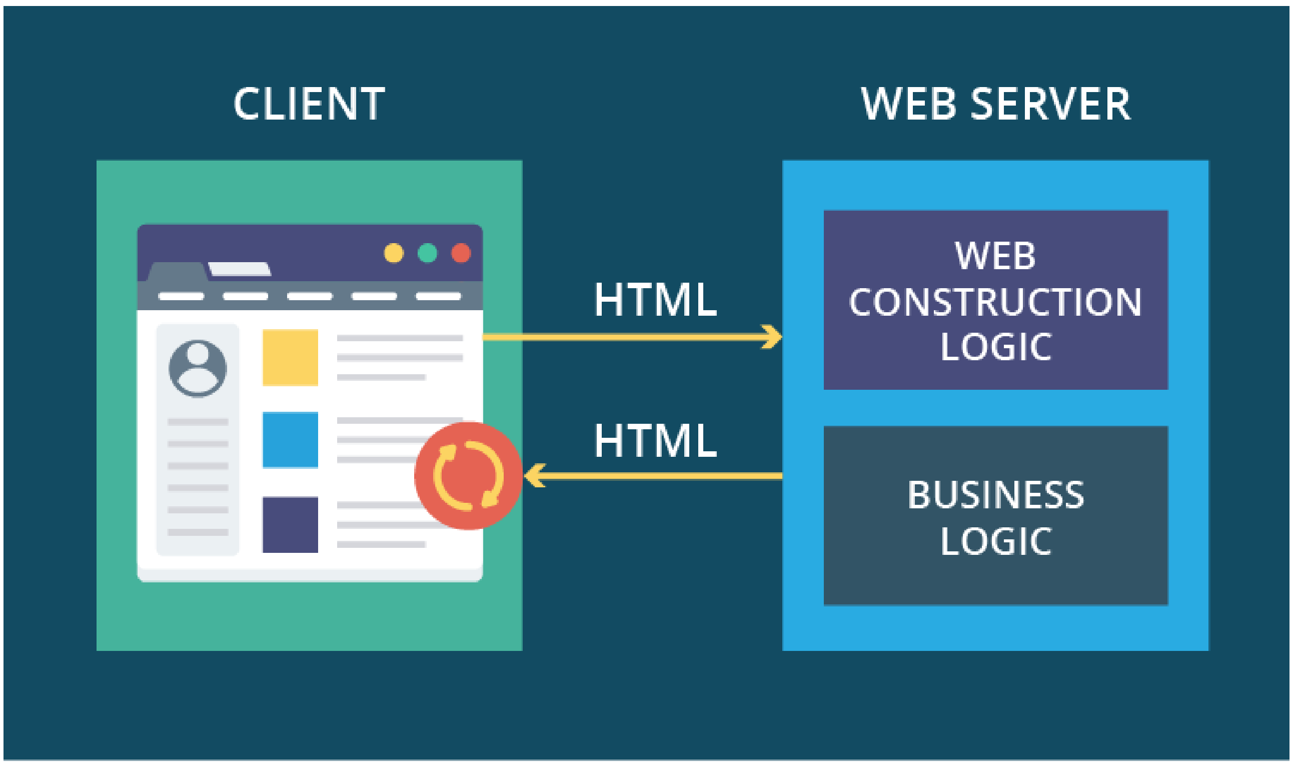
Base A Architeture


SERVER
INTERNET
Function Library
Steel Section Function A (Python)
Steel Section Function B (Python)
WEB Service API (Python)
HTTP
Web Apps
HTTP
Other Clients (Excel, Matlab, etc.)
Materials and Sections Database (MySQL)
Internal Clients

Base B Architeture

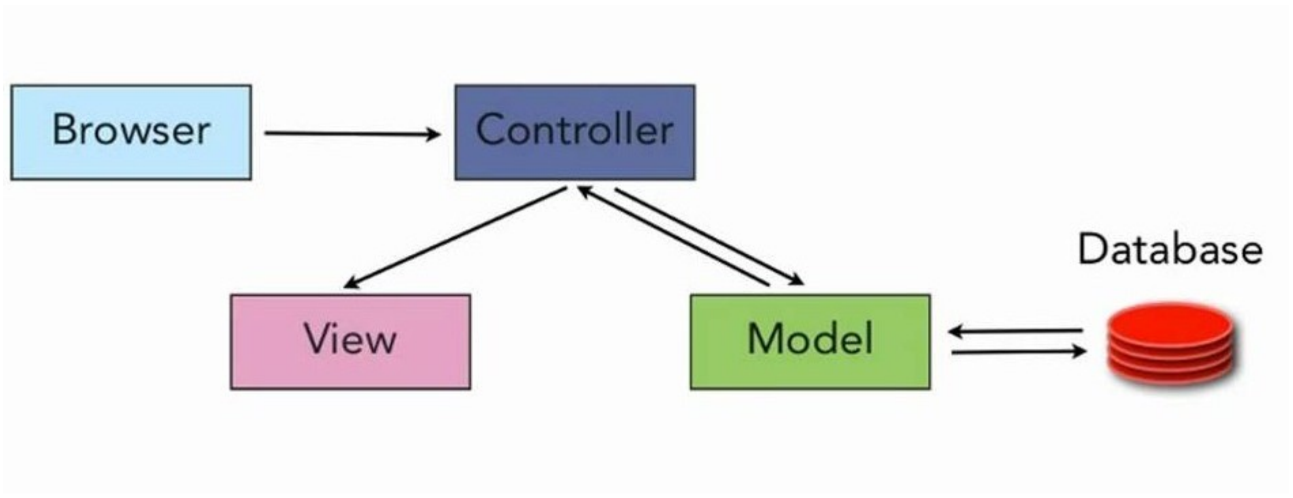


Base C Architeture



DESIGN PATTERN

MVC/MVP (Model View Controller or Model View Presenter)



AVAILABILITY

Base A
The database system(s) should not be connected to the internet. If necessary, it should use intranet. Restrict physical access to the database system and to cardholder data to only authorize person.
It should have Internal access only. If for any reason, an external access is allowed, it should be restricted to only some certain IP addresses.

Base B
Access can be granted through API calls with public and private keys or tokens only.

Base C
Access can be granted through WebAPP using queries. The WebAPP will contain a form, where a user can type in a CPF and retrieve related information.
In some cases, 2FACTOR Authentication can also be implemented. CAPTCHA implementation can also be helpful.