# PROJECT PLAN AVOC

**INTRODUZIONE**:

**Avoc** è un web app realizzata per la concessionario Boninsegna s.r.l. che consenta ai clienti di richiedere un preventivo per comprendere la convenienza tra un noleggio e un leasing.

All' interno dell'applicativo sono presenti due interfacce:

Una consultabile dai clienti ed una riservata ai dipendenti della concessionaria

Il pannello di amministrazione prevede la possibilità di inserire o rimuovere automezzi e di modificare i parametri di quelle già esistenti.

L' area riservata ai clienti deve consentire la visualizzazione delle informazioni dei veicoli e la possibilità di richiedere un preventivo per gli stessi, descrivendo, anche con l'ausilio di grafici esplicativi, l'andamento del canone mensile.

Il servizio deve essere raggiungibile da qualsiasi dispositivo, prevedendo l'adattabilità su qualsiasi schermo.

**PROCESS MODEL:**

Per l'organizzazione del progetto si è deciso di adattare la metodologia AGILE SCRUM.

Nello specifico si è stabilito di dividere il progetto in sprint da due settimane, dove giornalmente ci si riunisce su Google meet per informare il gruppo sugli sviluppi individuali effettuati nella giornata precedente, includendo, se presenti, le problematiche riscontrate ed infine comunicare i task che si intendono svolgere.

**ORGANIZZAZIONE DEL PROGETTO:**

Abbiamo deciso di optare per la metodologia Adhocracy, in quanto, giornalmente si condividono le problematiche riscontrate singolarmente e, per evitare di incorrere in task bloccanti, si dedica del tempo al pair programming per la risoluzione delle stesse.

**STANDARD :**

| | |
|---|---|
| **Visual Studio, PhpStorm**: | IDE per sviluppo php e linguaggi di markup; |
| **Datagrip:** | Web server control per visualizzazione dei dati del db in forma tabellare; |
| **PhpMyAdmin:** | Web app per amministrazione database MySQL; |
| **Apache:** | Web server; |

**MDB:** Framework con material design per Bootstrap;







**ATTIVITÁ DI MANAGEMENT:**

Come già anticipato precedentemente, si è deciso di suddividere il progetto in sprint da 2 settimane dove alla, dove alla fine di ognuno ci si riunisce, anche con il cliente, e viene mostrato lo stato di avanzamento del progetto.
Si prosegue poi, senza la presenza del cliente, con l' attività di planning, ossia definire quali task si intendono inserire nello sprint successivo, tenendo anche conto dei feedback riportati nell' incontro precedente, e si esegue la retrospective e lo sprint review dove si valuta complessivamente lo sprint appena concluso e si elenca rispettivamente:

- Cosa si può migliorare;
- Cosa è stato fatto bene;
- Cosa non è piaciuto;

**RISCHI**:
Siccome alcuni elementi del gruppo sono impegnati anche in altri progetti universitari, nonchè occupati dalla vita lavorativa, risulta difficoltoso trovare dei momenti in cui tutti i componenti del gruppo risultino liberi, tralasciando i daily meeting, comportando un potenziale problema a rispettare i tempi di consegna.

**STAFFING:**

Silviu Mihaita Filote Pandelea -> Back-end developer

Jonathan Bommarito: Front-end developer

Nicolò Carissimi: DevOps

Simone Ronzoni-> Full stack developer

**METODI E TECNICHE:**

Per la documentazione in fase di requirement-engineering e design utilizzeremo UML:

1) Use Case Diagram, per descrivere le funzioni e i servizi forniti dal sistema e come gli attori che interagiscono con il sistema li percepiscono e li usano;

2) Class Diagram per rappresentare la struttura statica del sistema descrivendo quindi entità e le loro relazioni;

3) Sequence Diagram che descrive in che ordine devono essere scambiate sequenze di messaggi per completare una task

4) Activity Diagram, utilizzato per descrivere l'attività e il suo processo;

5) State Machine Diagram che descrive gli stati possibili in cui un oggetto può trovarsi e le possibili transizioni tra questi stati.

Utilizzo di GitHub per il versioning del progetto.

Si ritaglierà una porzione di tempo per ogni sprint dedicata ai test di features.

**RISORSE:**

Per lo sviluppo del progetto sarà richiesto:

- l' utilizzo di un server fornito da Altervista;

- Per l' installazione delle dipendenze sul server sarà necessario l' ausilio di npm.

**QUALITÁ:**

Per garantire la qualità nel progetto si sono seguite le best practise per evitare problematiche già note (SQL injection...) e ad aumentare la complessiva solidità del sistema.

**BUDGET:**

Il budget nella versione iniziale non prevederà alcun tipo di costo legato ai sistemi utilizzati, infatti il server su cui verrà hostato è gratuito in quanto il provider del servizio è Altervista.

Il costo del personale che ha lavorato al progetto è nullo.

**CAMBIAMENTI:**

É utilizzata la tecnica del versioning attraverso Git.

Ogni cambiamento viene trasmesso alla repository in un branch personale(solitamente dettato da un task), successivamente viene aperta una pull-request o merge-request assegnata ad un reviewer che provvederà ad esaminare la commit effettuata ed accettare o meno la richiesta.

In seguito, in caso di approvazione della pull request verrà effettuato un merge sul branch Develop (il branch master verrà utilizzato per il deploy sul server in produzione).

**DELIVERY:**

Per il deploy in produzione utilizziamo come strategia quella di

"Basic Deployment" attraverso il branch master:

Questa strategia prevede che tutti i nodi all' interno dell' ambiente di produzione vengano aggiornati contemporaneamente con la nuova versione del servizio.