JAKE WHARTON

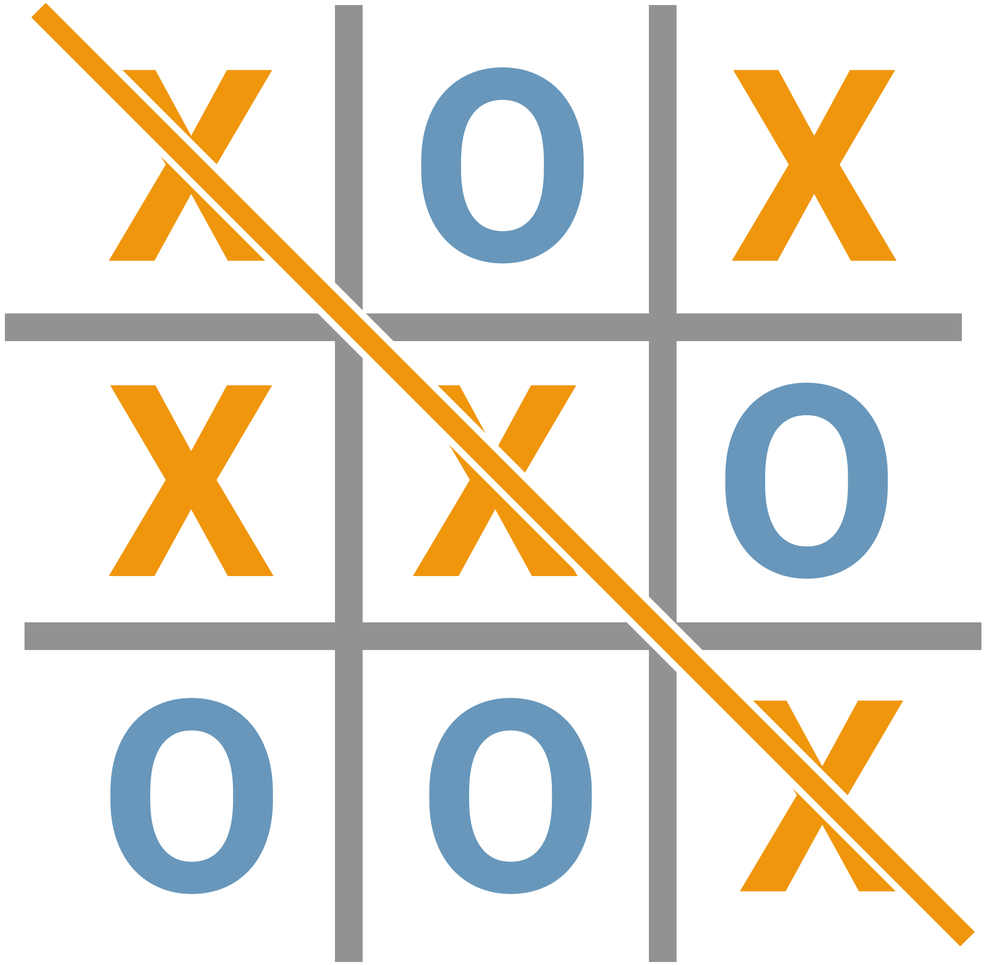
PRESENTS

IT'S A KOTLIN, KOTLIN, KOTLIN WORLD!

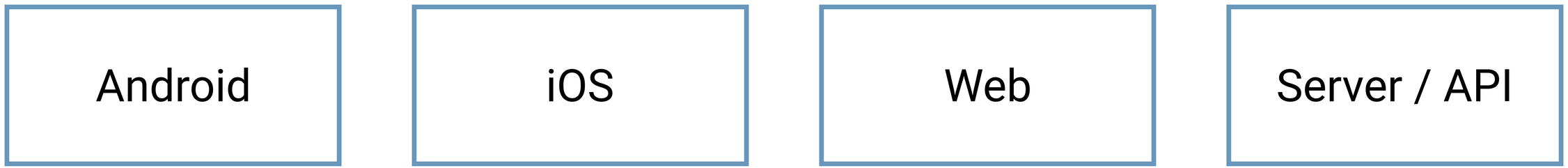
K

X

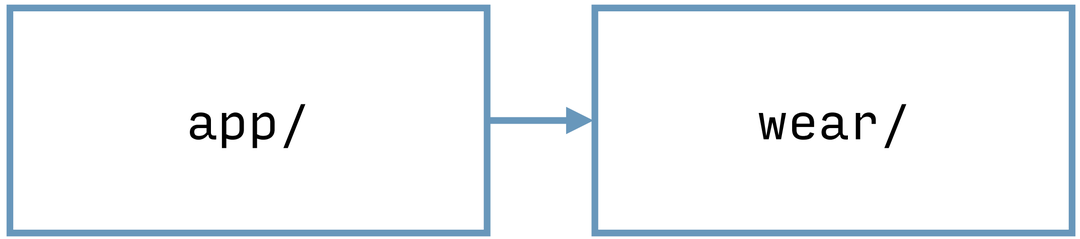
X O

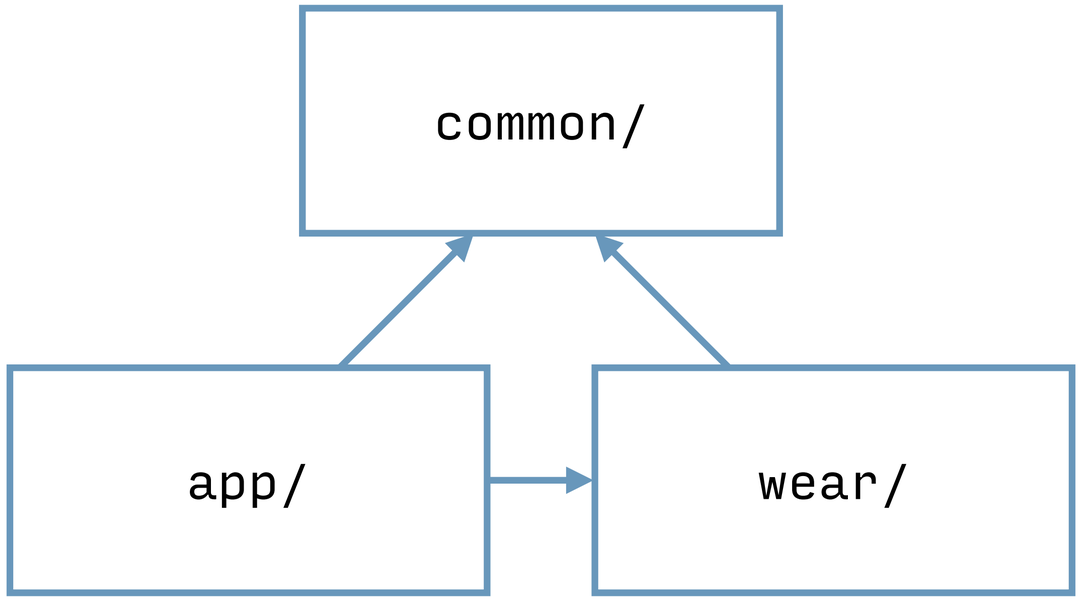

X O X
X X O
O O X

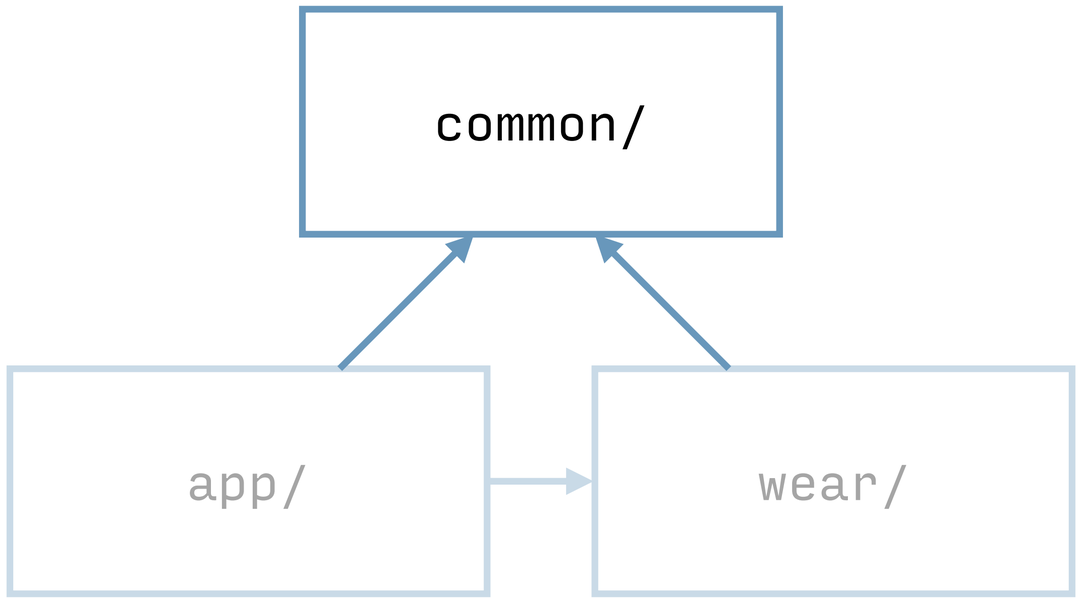
XOPlayer

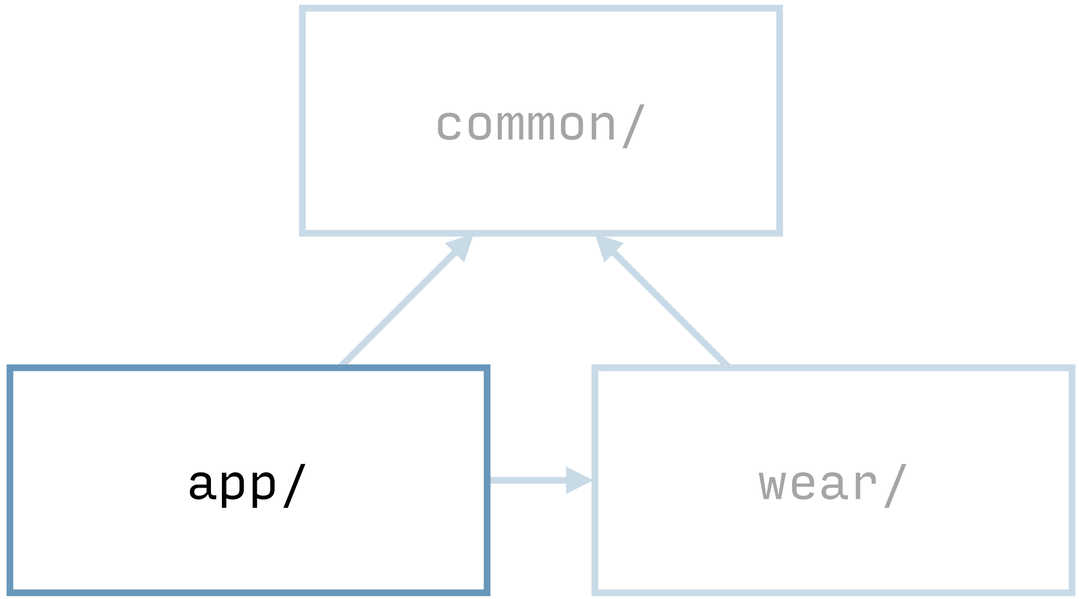

Android
iOS
Web
Server / API

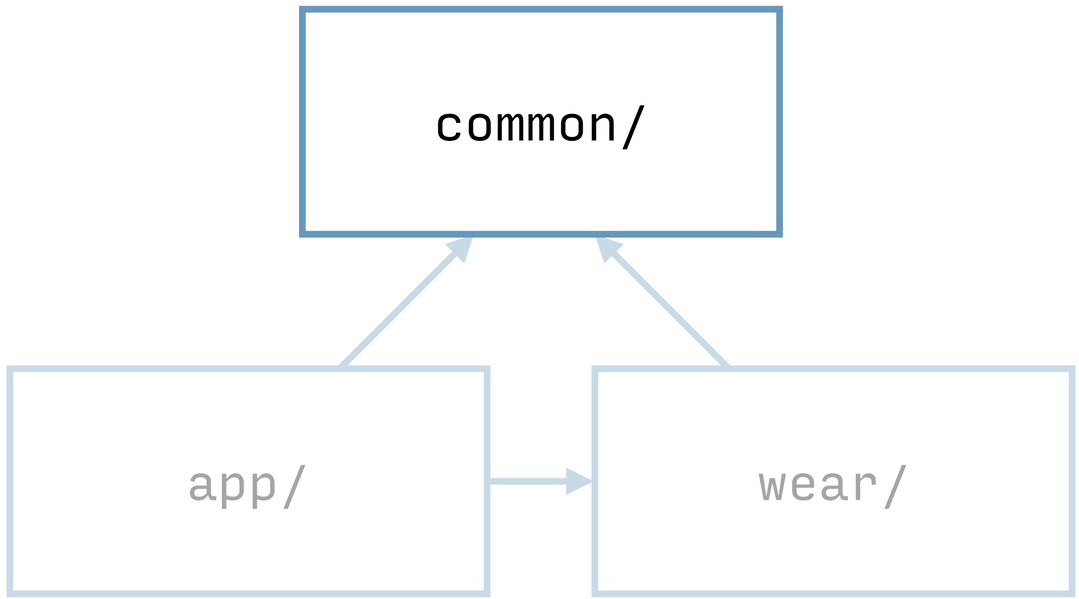

app/

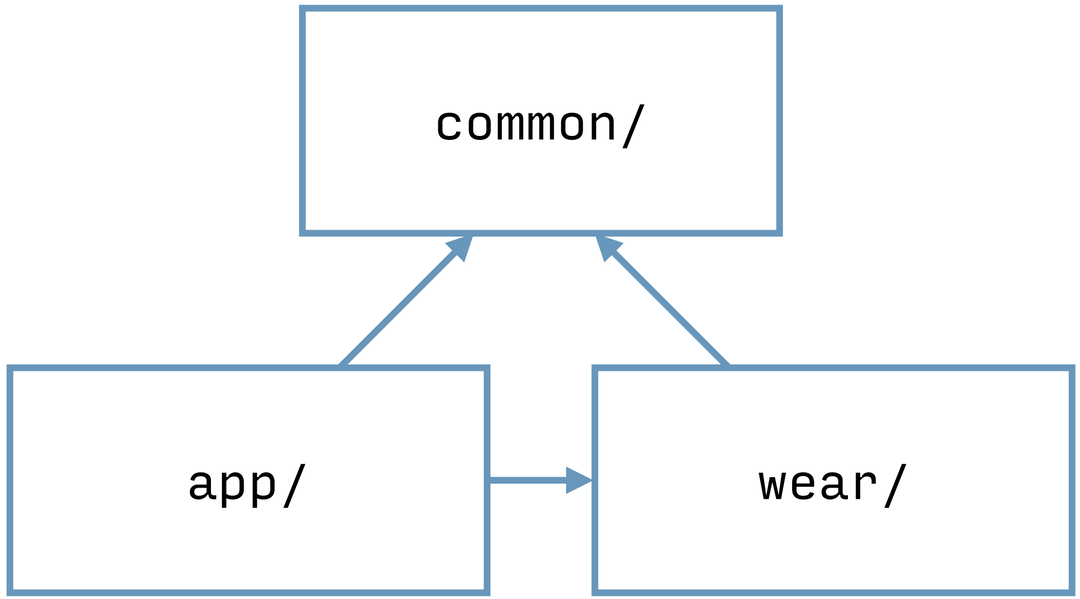

app/
wear/

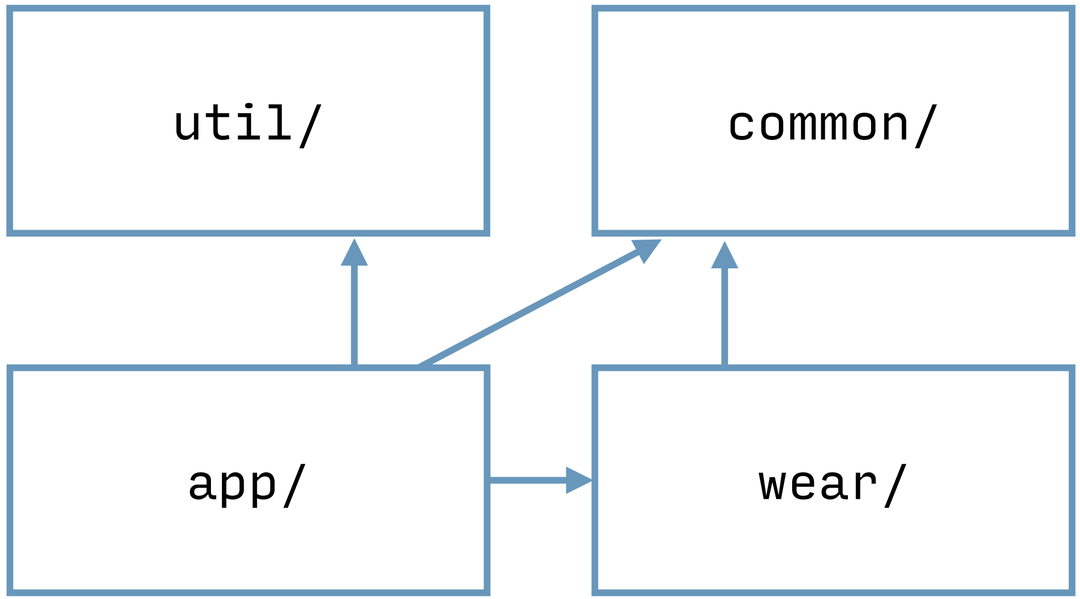

common/
app/
wear/

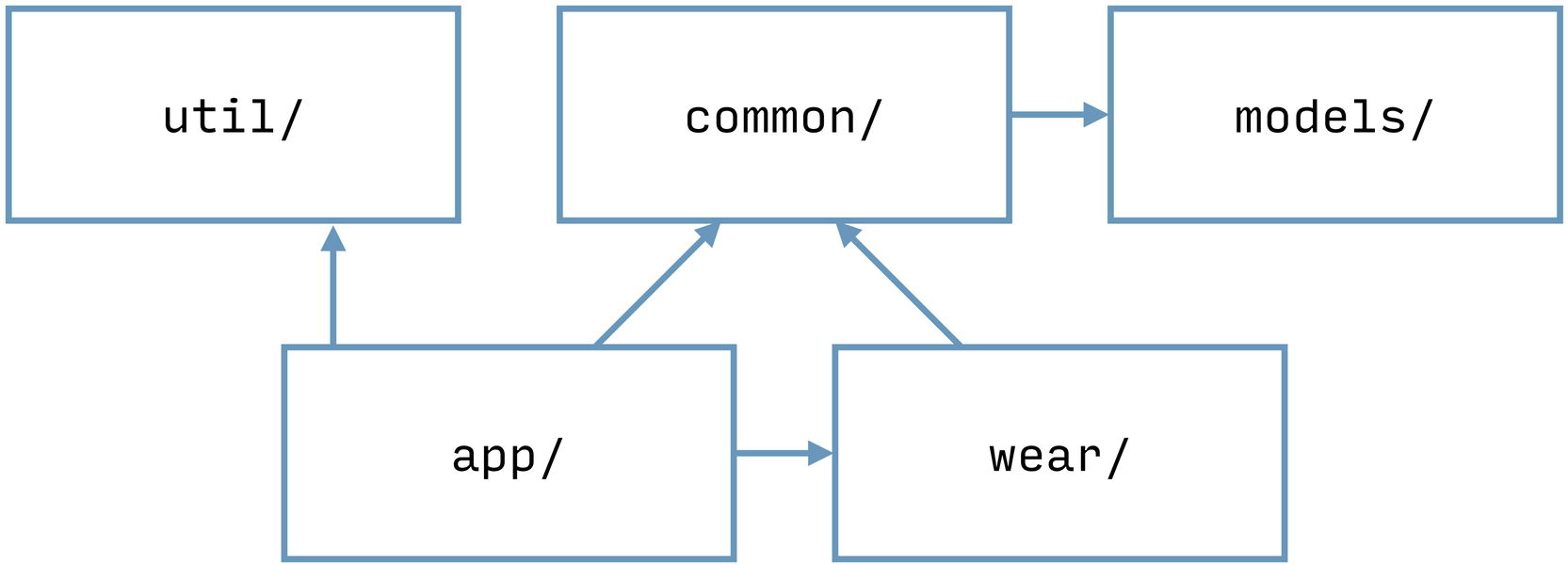

common/
app/
wear/

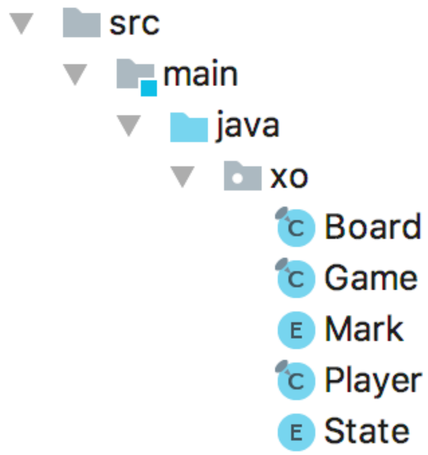

common/
app/
wear/

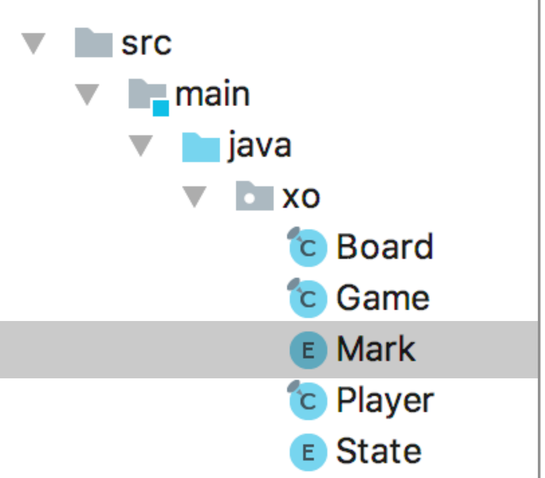

common/
app/
wear/


common/
app/
wear/

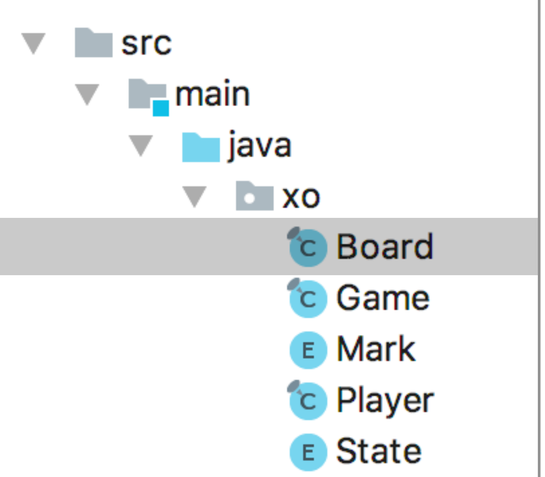

util/
common/
app/
wear/


util/
common/
models/
app/
wear/

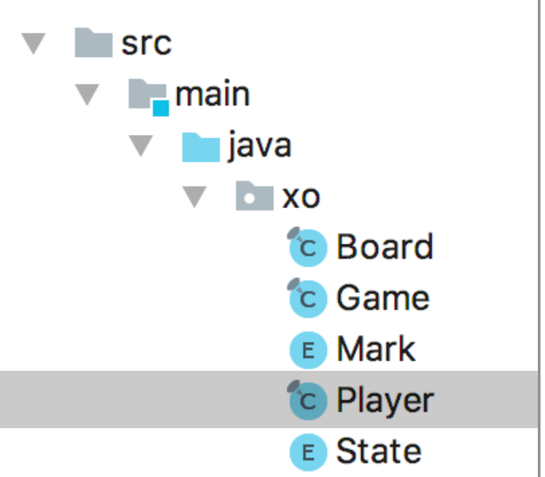

src
main
java
xo
Board
Game
Mark
Player
State



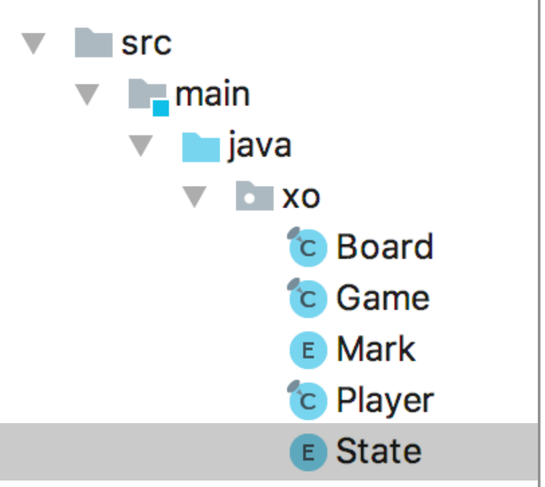

```java
package xo;

public enum Mark {
  X, O;
}
```



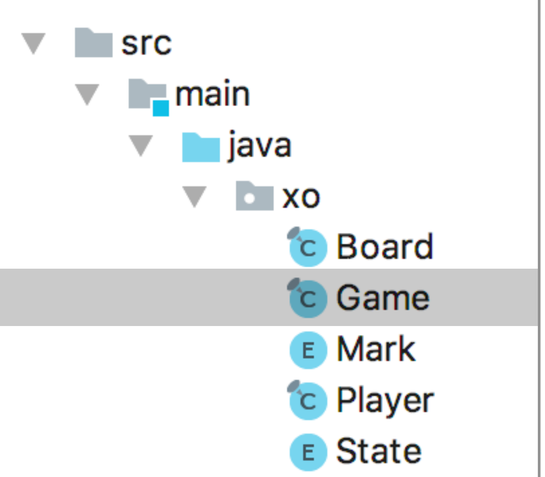

```java
package xo;

import java.util.Arrays;

public final class Board {
  private static final int SIZE = 3;

  private final Mark[][] cells;

  public Board() {
    this.cells = new Mark[3][3];
  }

  // TODO mutator methods...

  @Override public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Board)) return false;
    Board other = (Board) o;
    return Arrays.deepEquals(cells, other.cells);
  }

  @Override public int hashCode() {
    return Arrays.deepHashCode(cells);
```



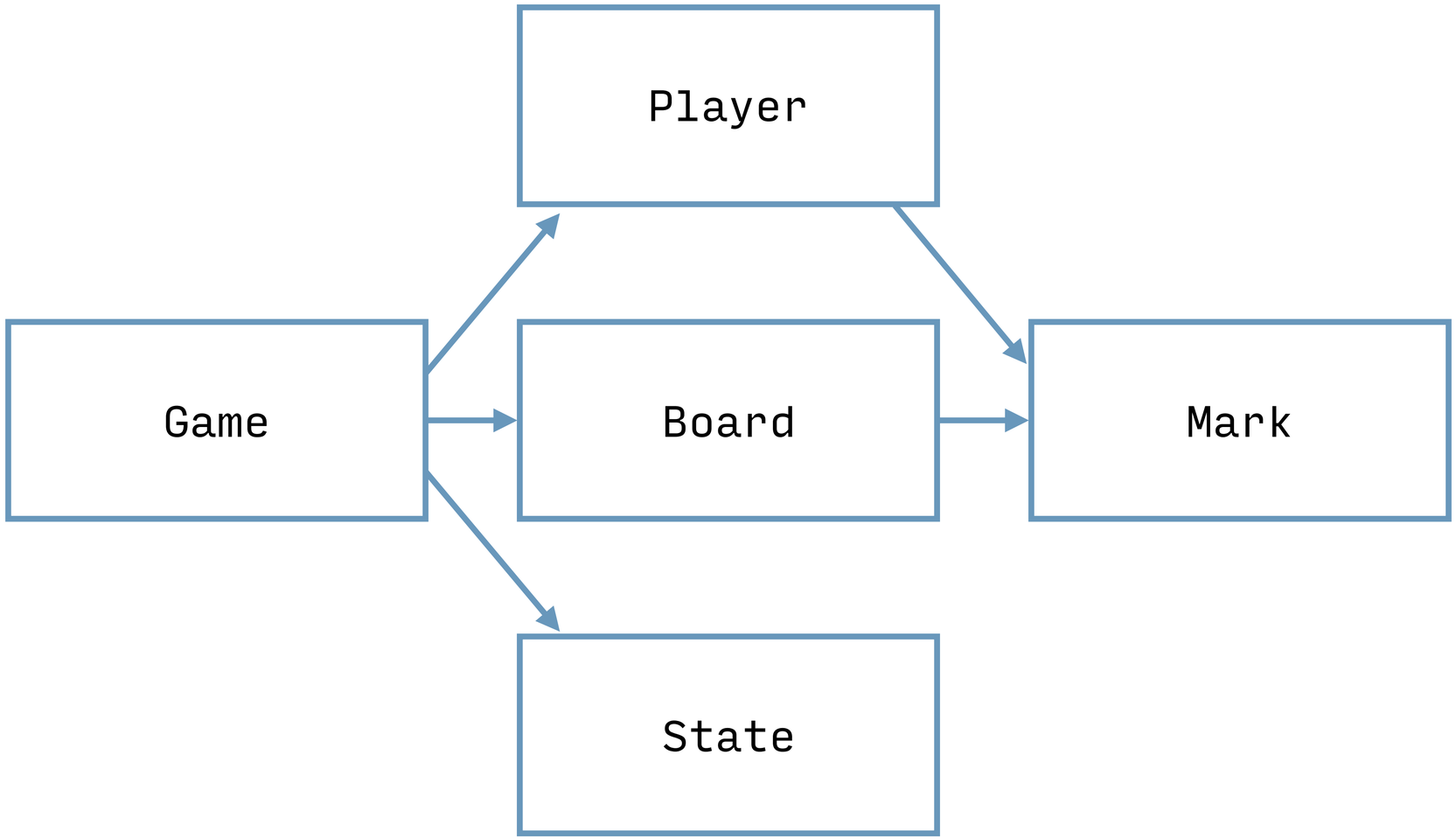

```java
package xo;

import static java.util.Objects.requireNonNull;

public final class Player {
  public final String name;
  public final Mark mark;

  public Player(String name, Mark mark) {
    this.name = requireNonNull(name, "name == null");
    this.mark = requireNonNull(mark, "mark == null");
  }

  @Override public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Player)) return false;
    Player other = (Player) o;
    return name.equals(other.name) && mark == other.mark;
  }

  @Override public int hashCode() {
    return 31 * name.hashCode() + mark.hashCode();
  }
```




```java
package xo;

public enum State {
  PLAYER_1_MOVE,
  PLAYER_2_MOVE,
  PLAYER_1_WIN,
  PLAYER_2_WIN,
  DRAW,
}
```

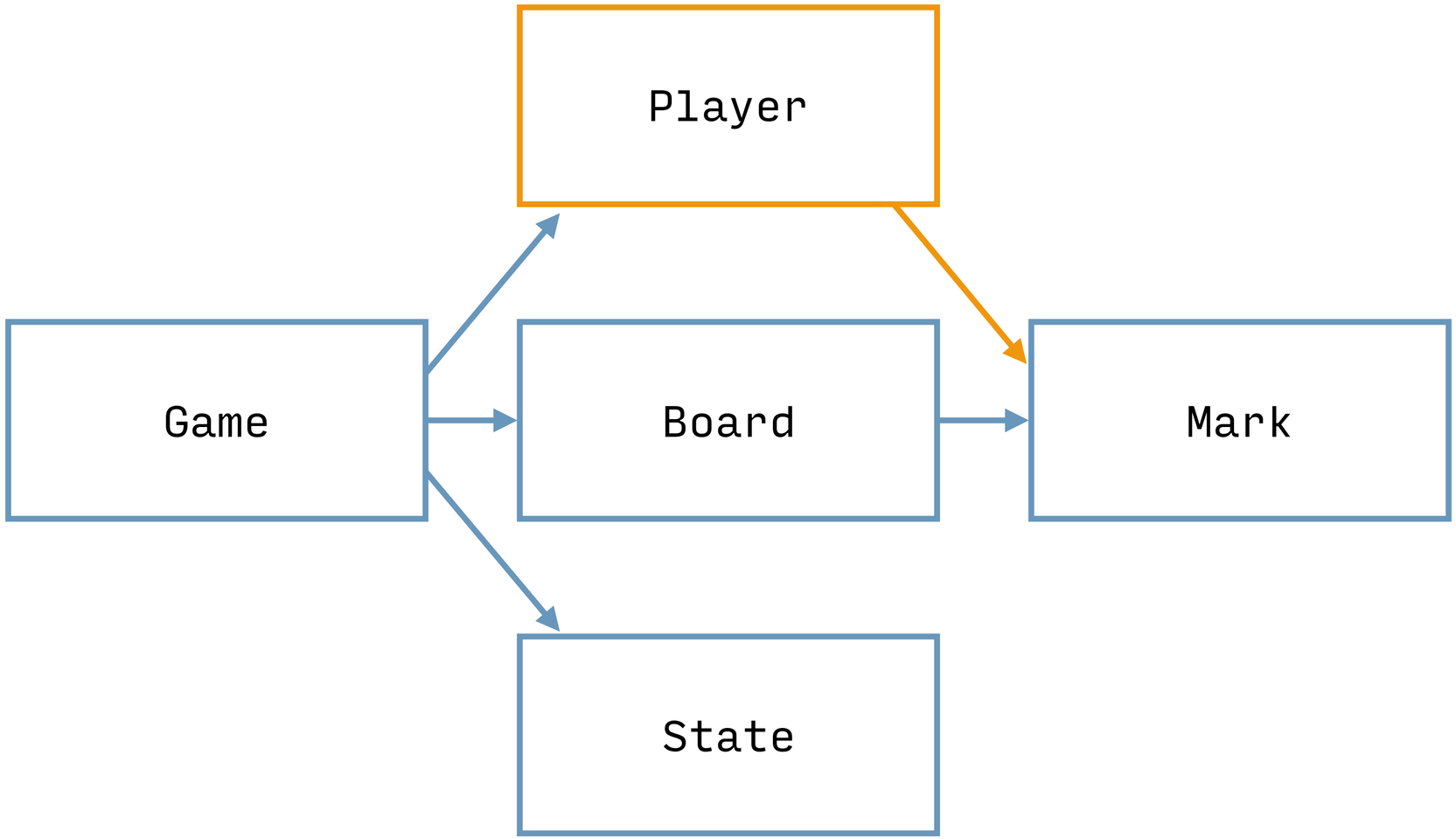


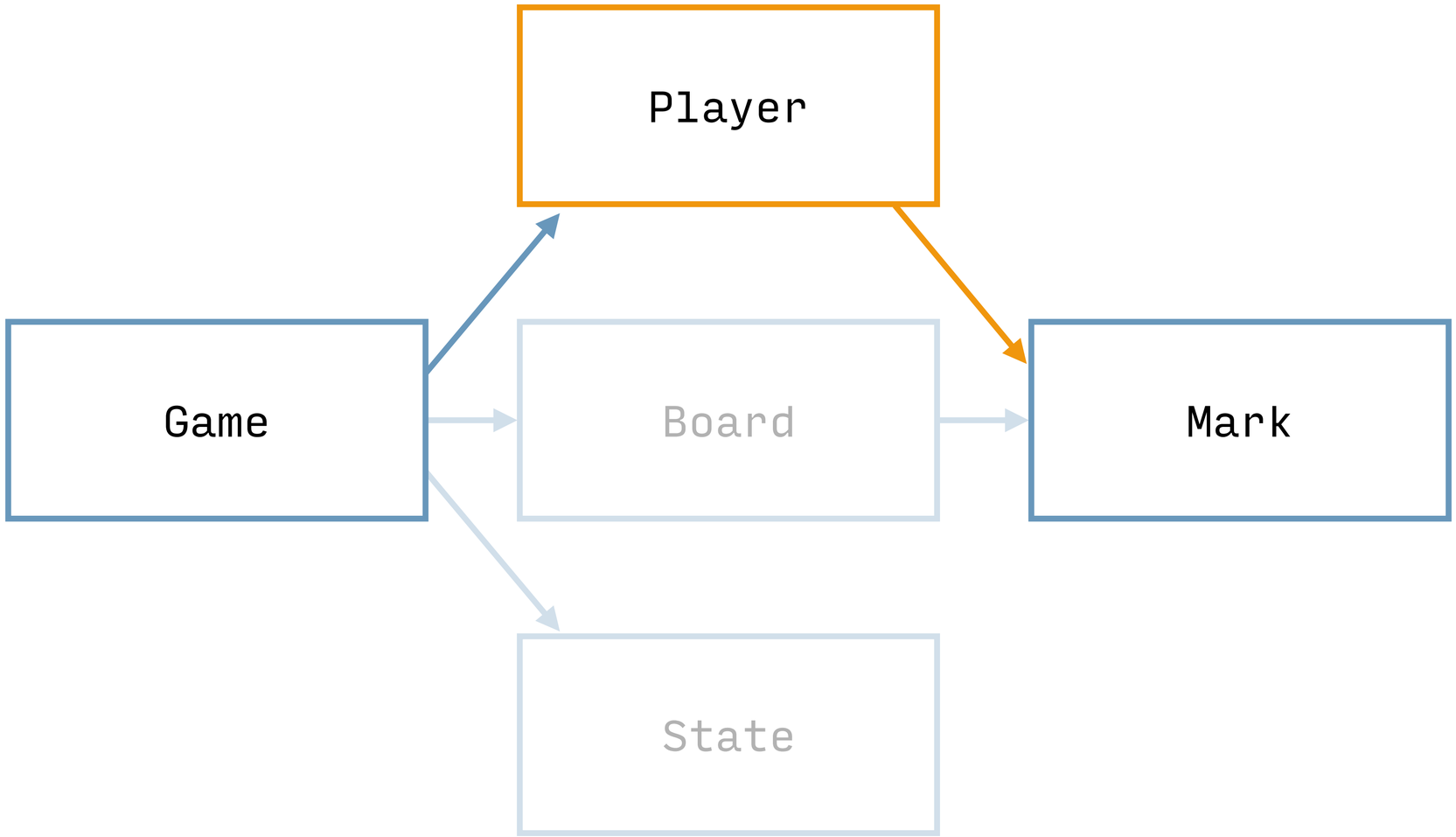

```java
package xo;

import static java.util.Objects.requireNonNull;

public final class Game {
  private final Board board;
  private final Player player1;
  private final Player player2;
  private State state = State.PLAYER_1_MOVE;

  public Game(Board board, Player player1, Player player2) {
    this.board = requireNonNull(board, "board == null");
    this.player1 = requireNonNull(player1, "player1 == null");
    this.player2 = requireNonNull(player2, "player2 == null");
  }

  // TODO mutator methods...

  @Override public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Game)) return false;
    Game other = (Game) o;
    return board.equals(other.board)
        && player1.equals(other.player1)
```

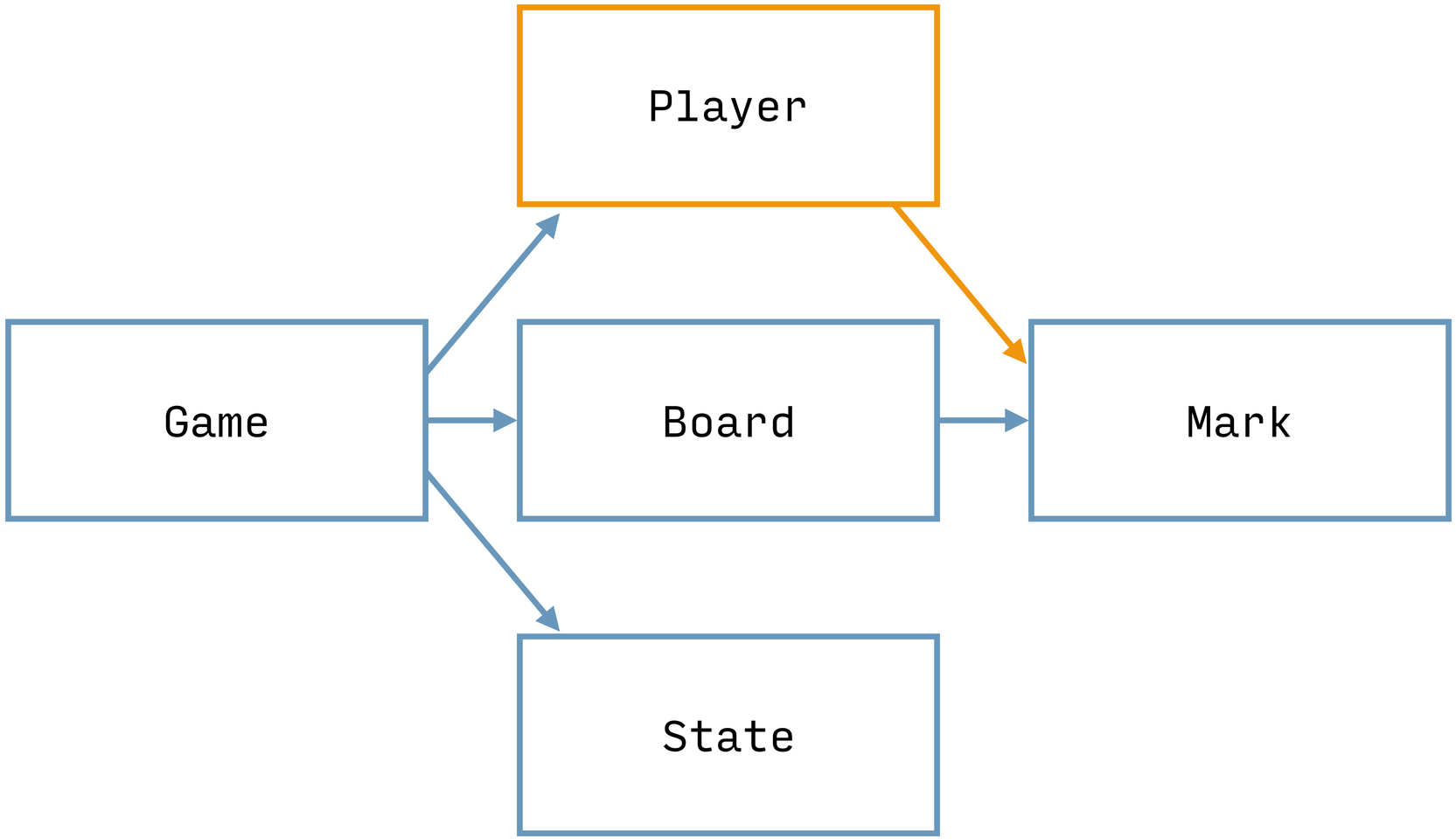

Player
Game
Board
Mark
State

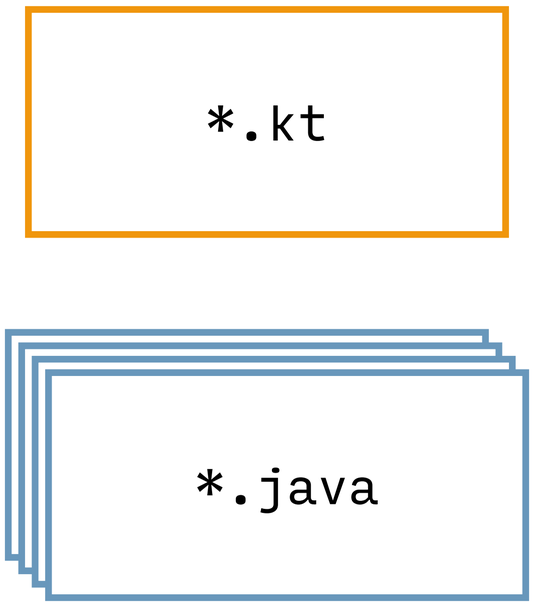
```java
public final class Player {
  public final String name;
  public final Mark mark;

  public Player(String name, Mark mark) {
    this.name = requireNonNull(name, "name == null");
    this.mark = requireNonNull(mark, "mark == null");
  }

  @Override public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Player)) return false;
    Player other = (Player) o;
    return name.equals(other.name) && mark == other.mark;
  }

  @Override public int hashCode() {
    return 31 * name.hashCode() + mark.hashCode();
  }

  @Override public String toString() {
    return "Player{name='" + name + ", mark=" + mark + '}';
  }
}
```

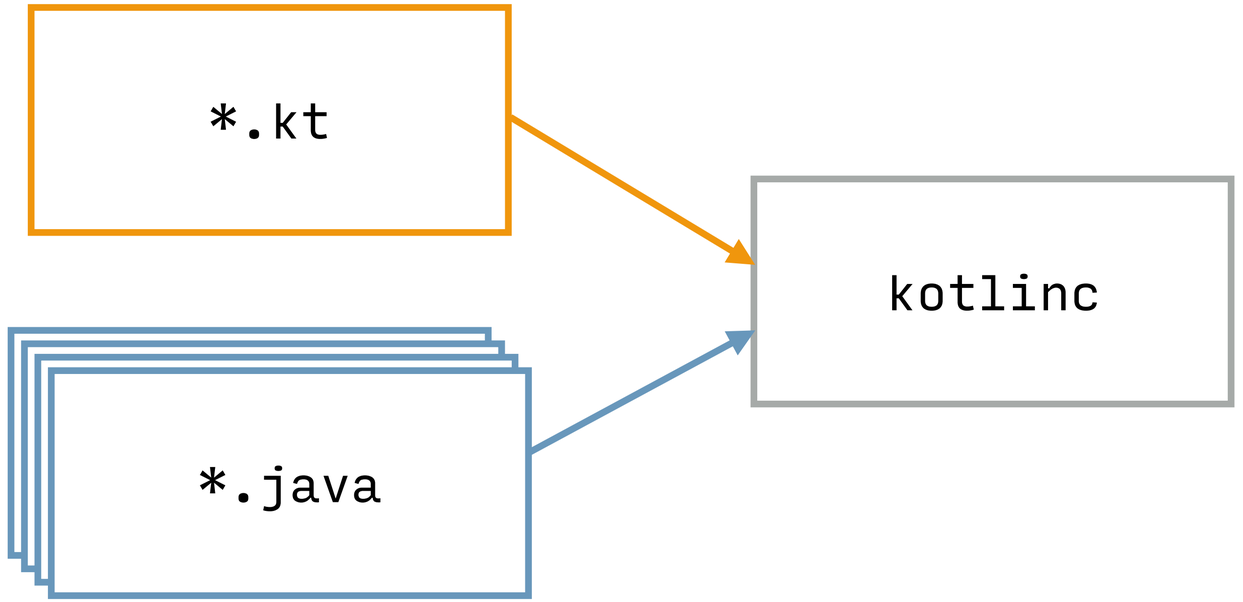
```
data class Player(val name: String, val mark: Mark)
```

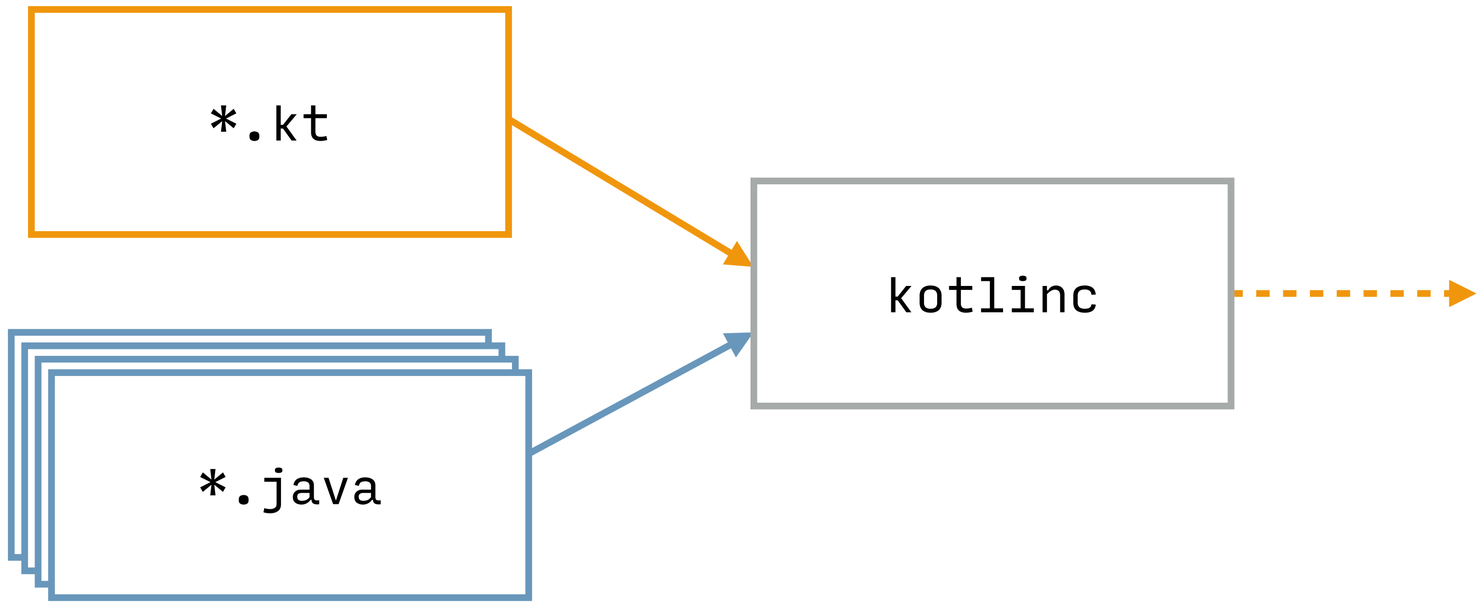

Player
Game
Board
Mark
State

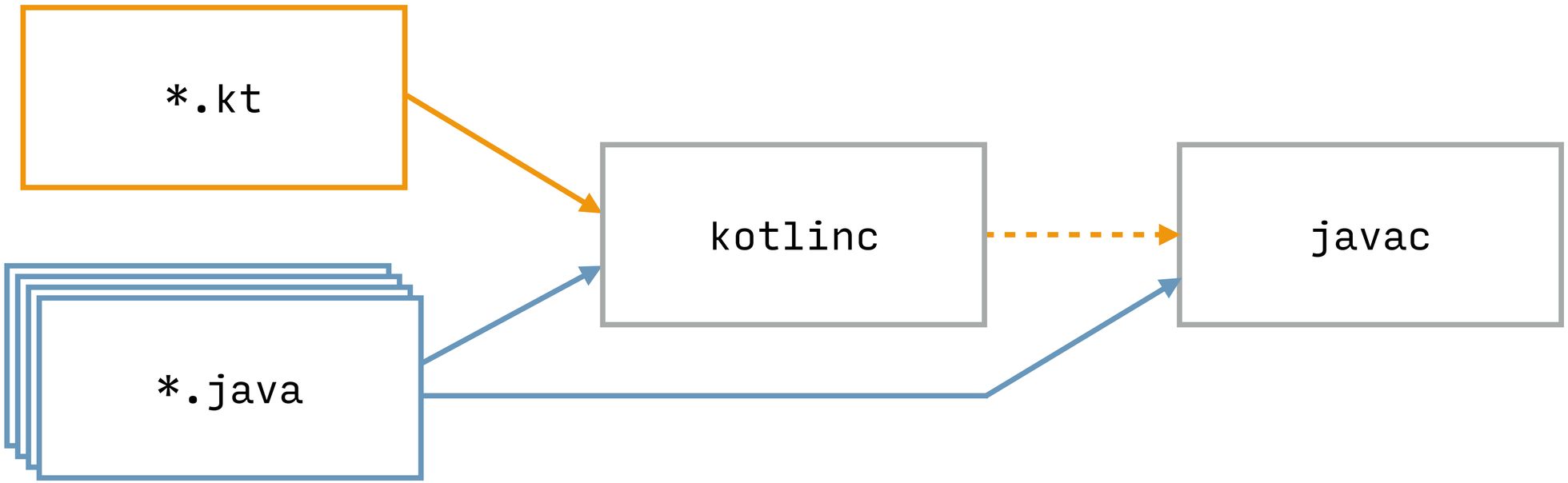

Player
Game
Board
Mark
State

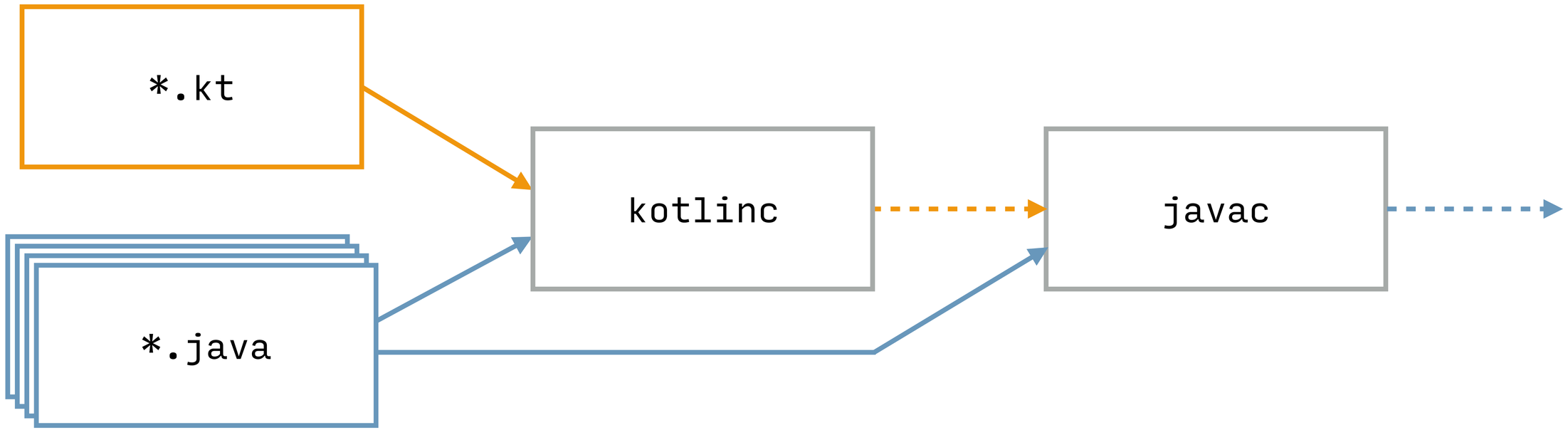

Player
Game
Board
Mark
State

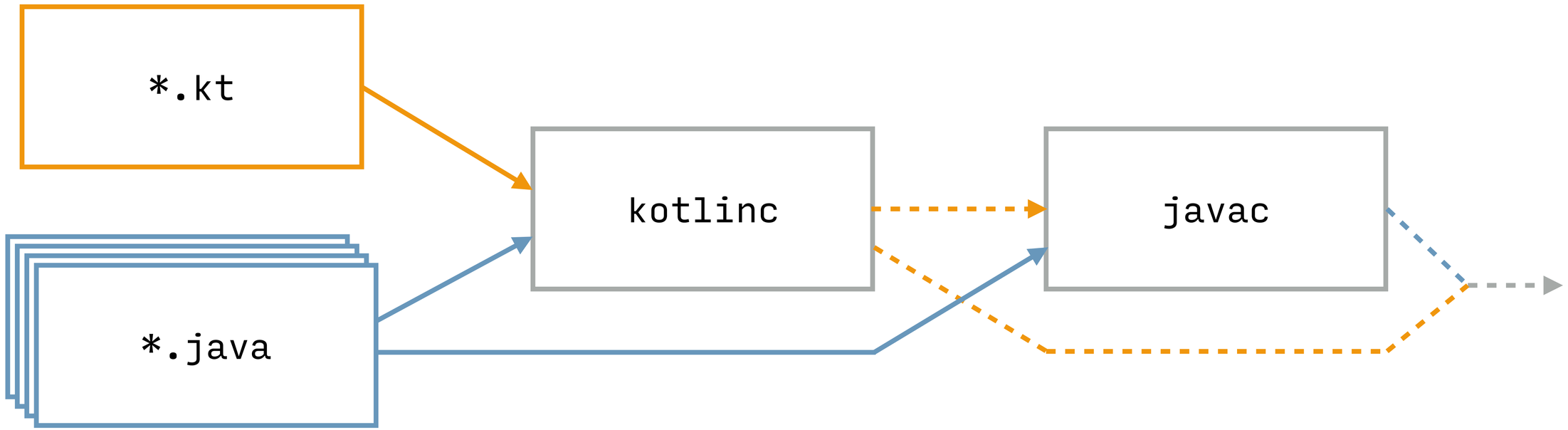

*.kt
*.java

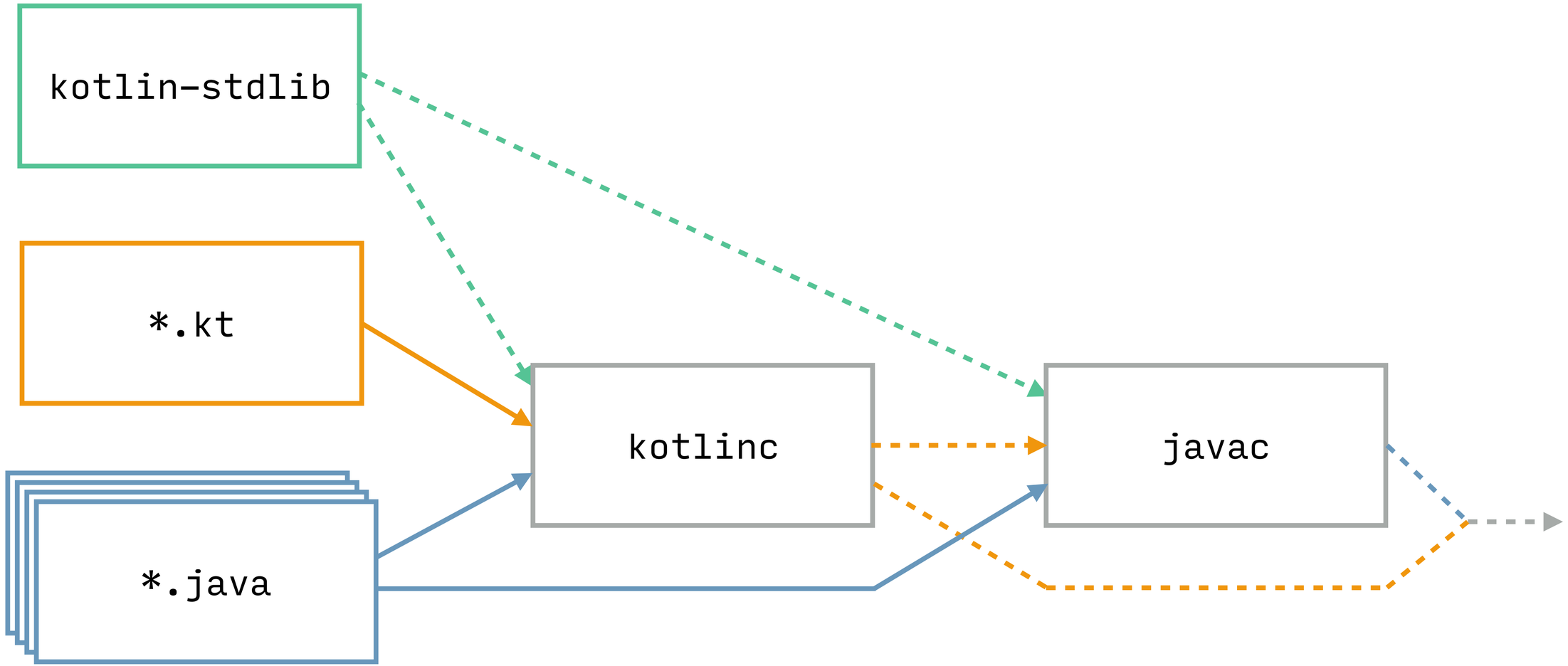

*.kt
*.java
kotlinc

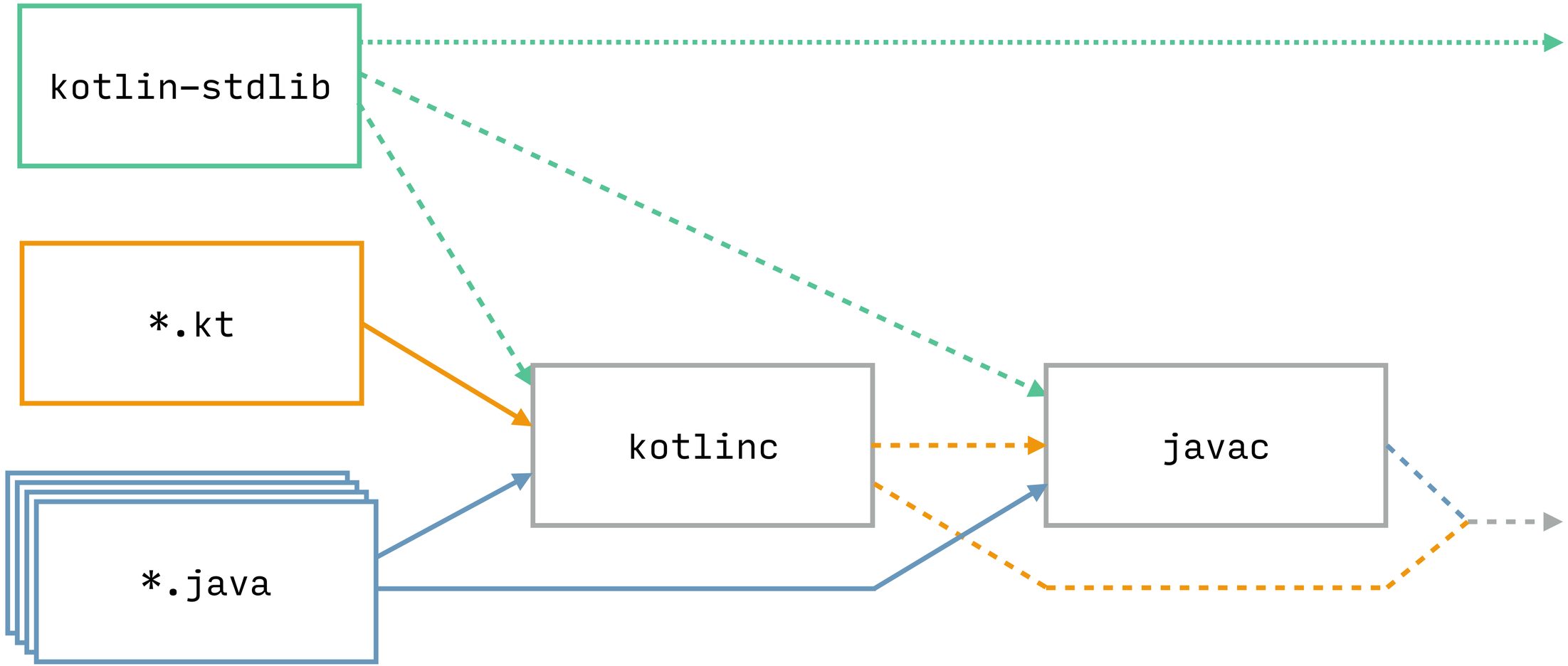

*.kt
*.java
kotlinc

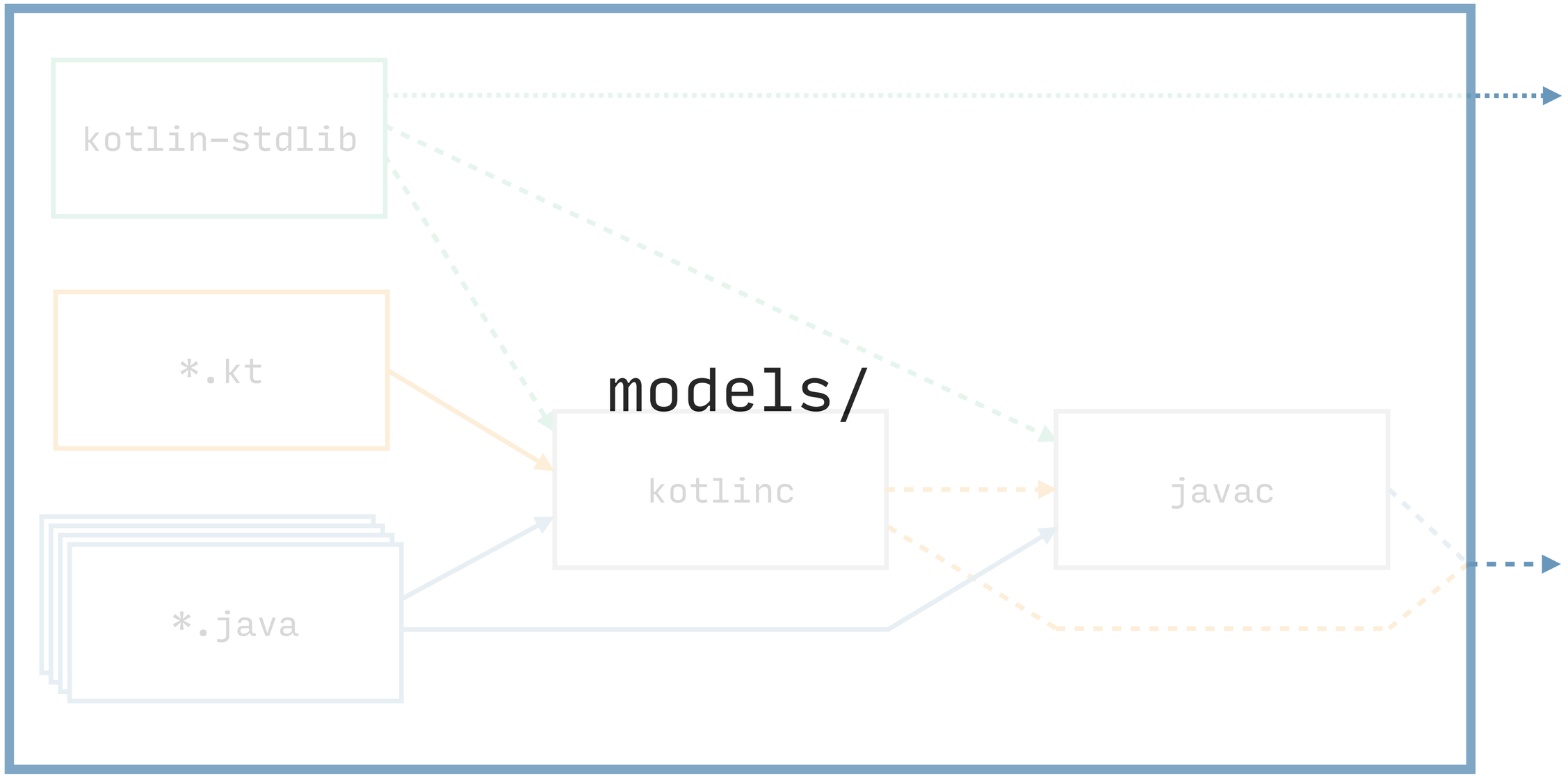

*.kt
*.java
kotlinc
javac

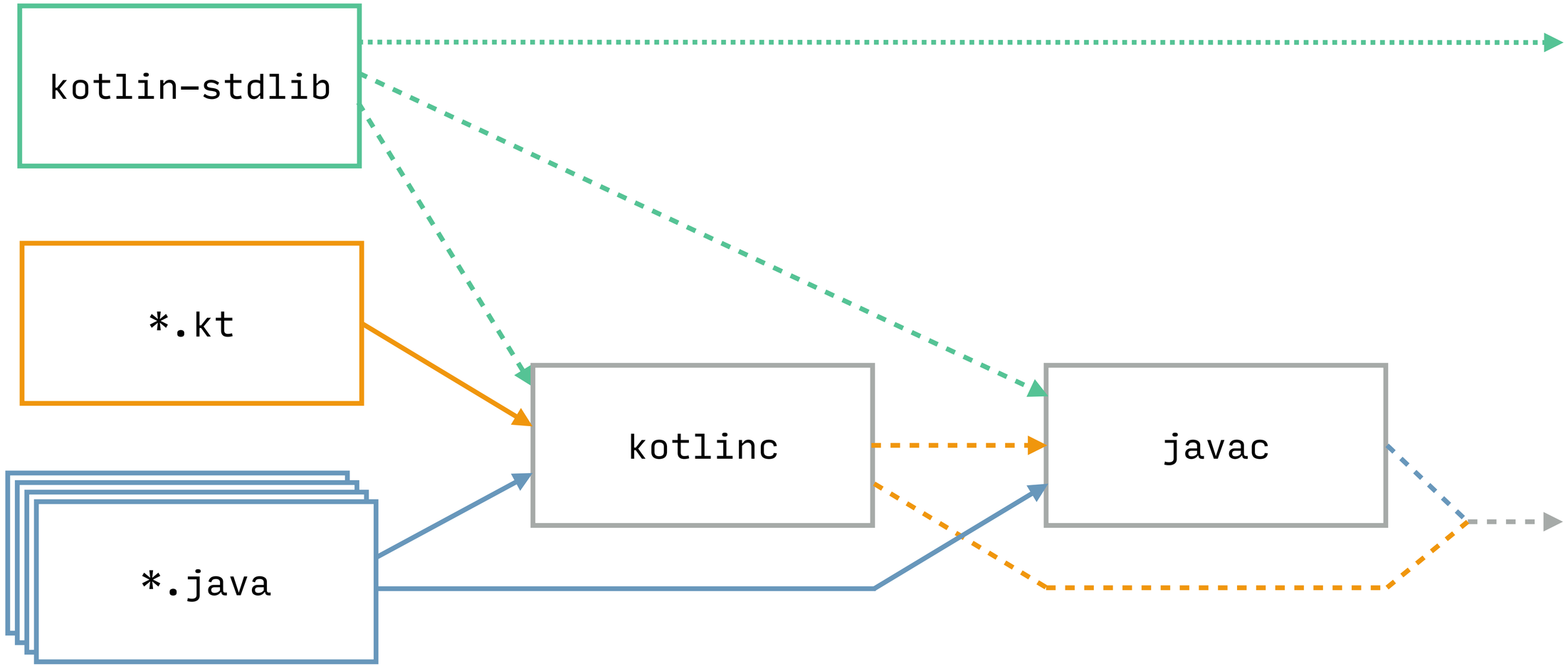

*.kt
*.java
kotlinc
javac

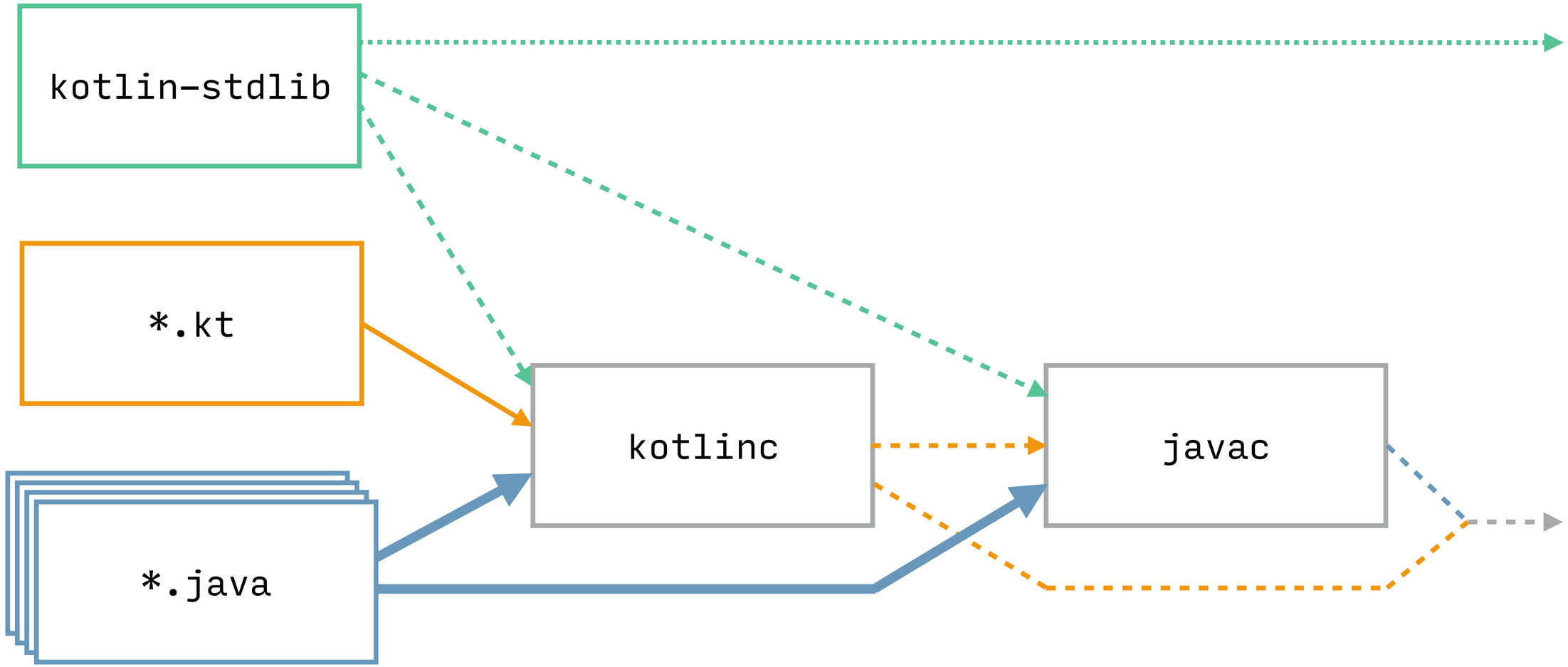

*.kt
*.java
kotlinc
javac

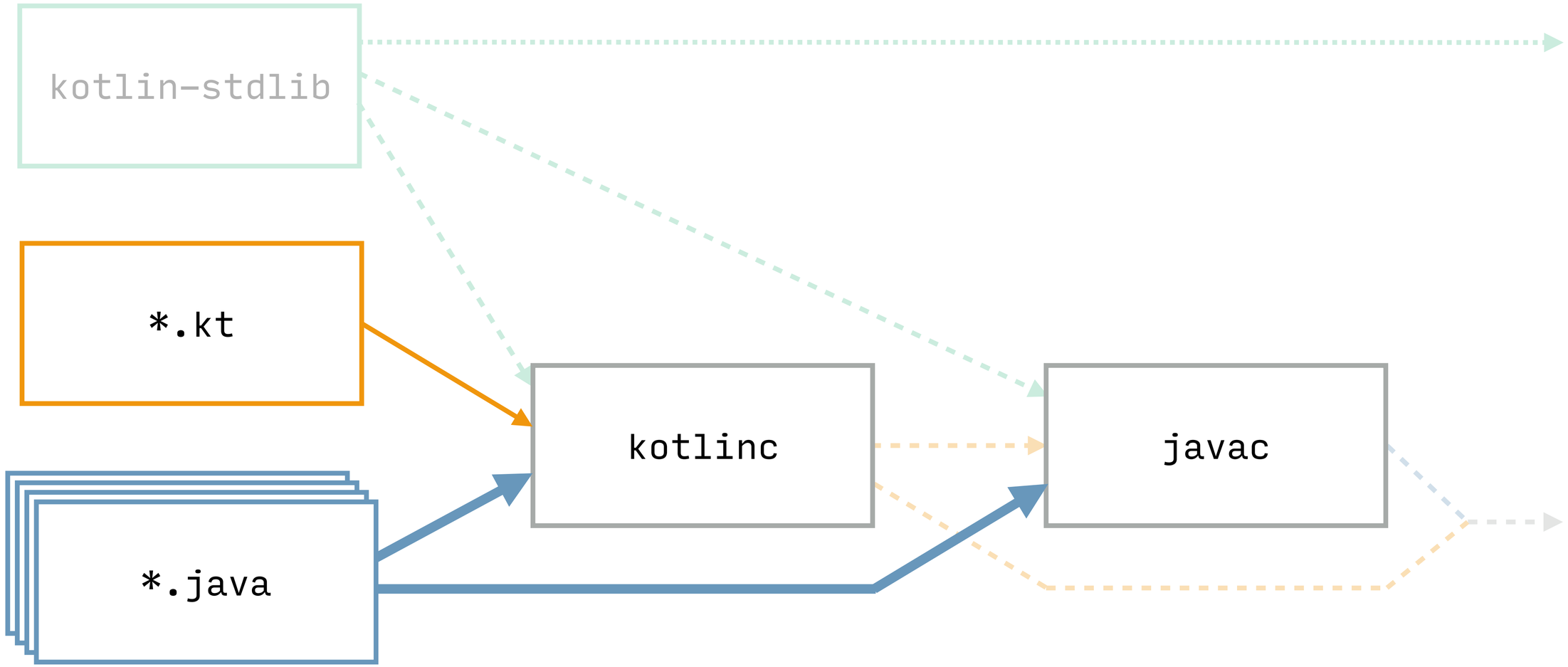

kotlin-stdlib
*.kt
*.java
kotlinc
javac

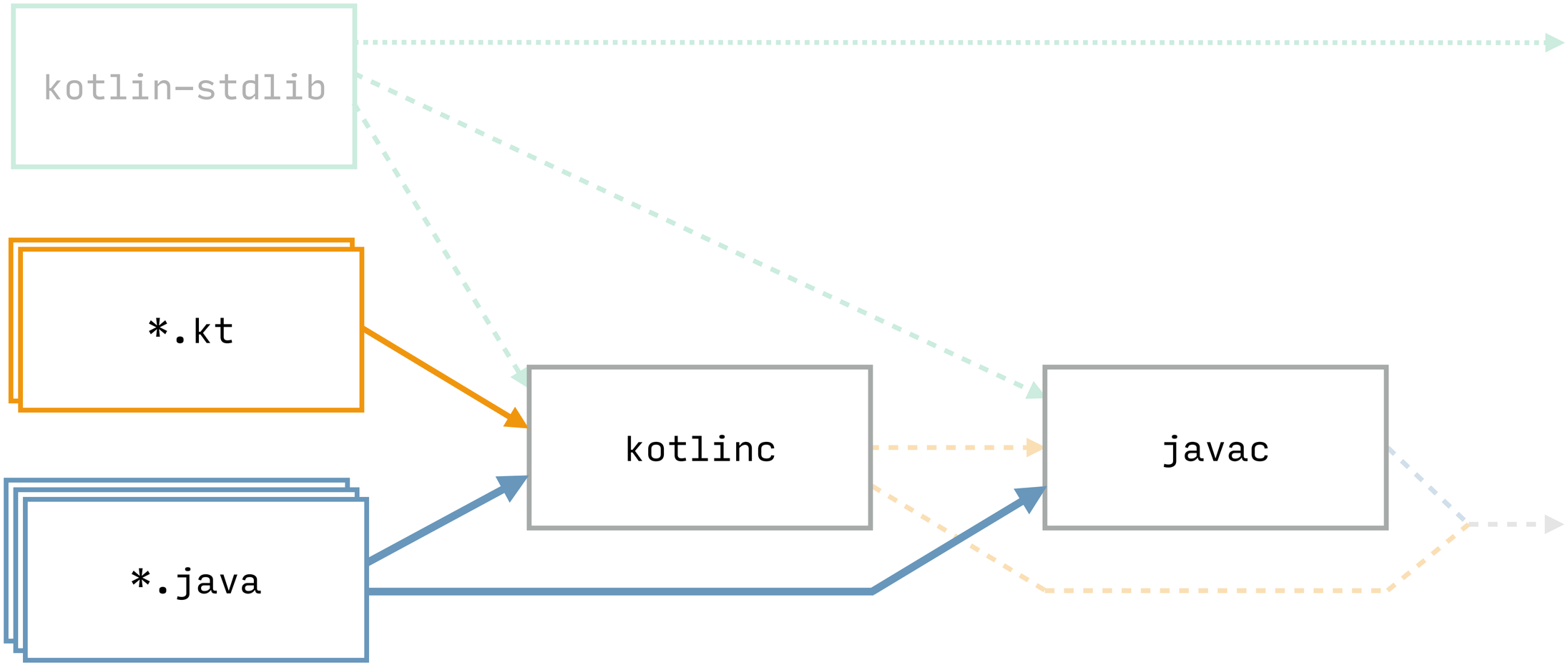

kotlin-stdlib
*.kt
*.java
kotlinc
javac

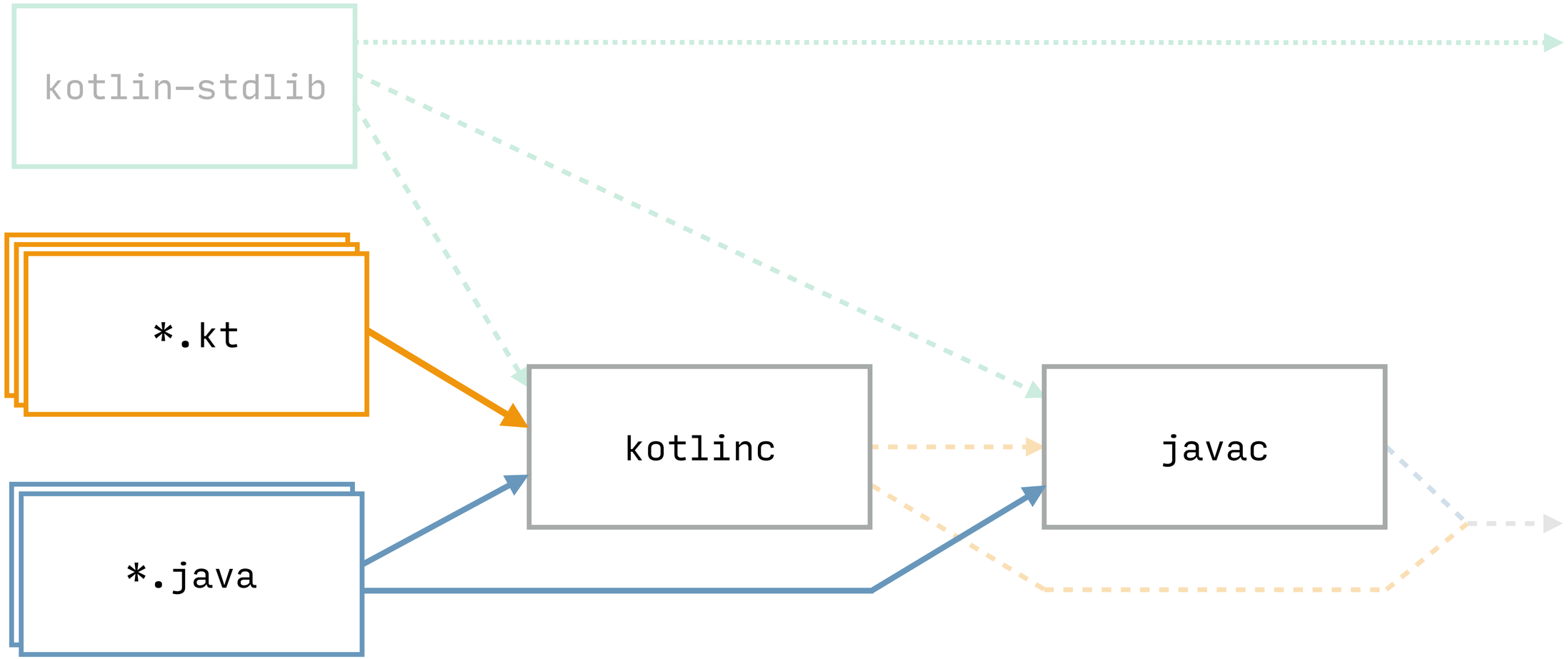

kotlin-stdlib
*.kt
*.java
models/
kotlinc
javac

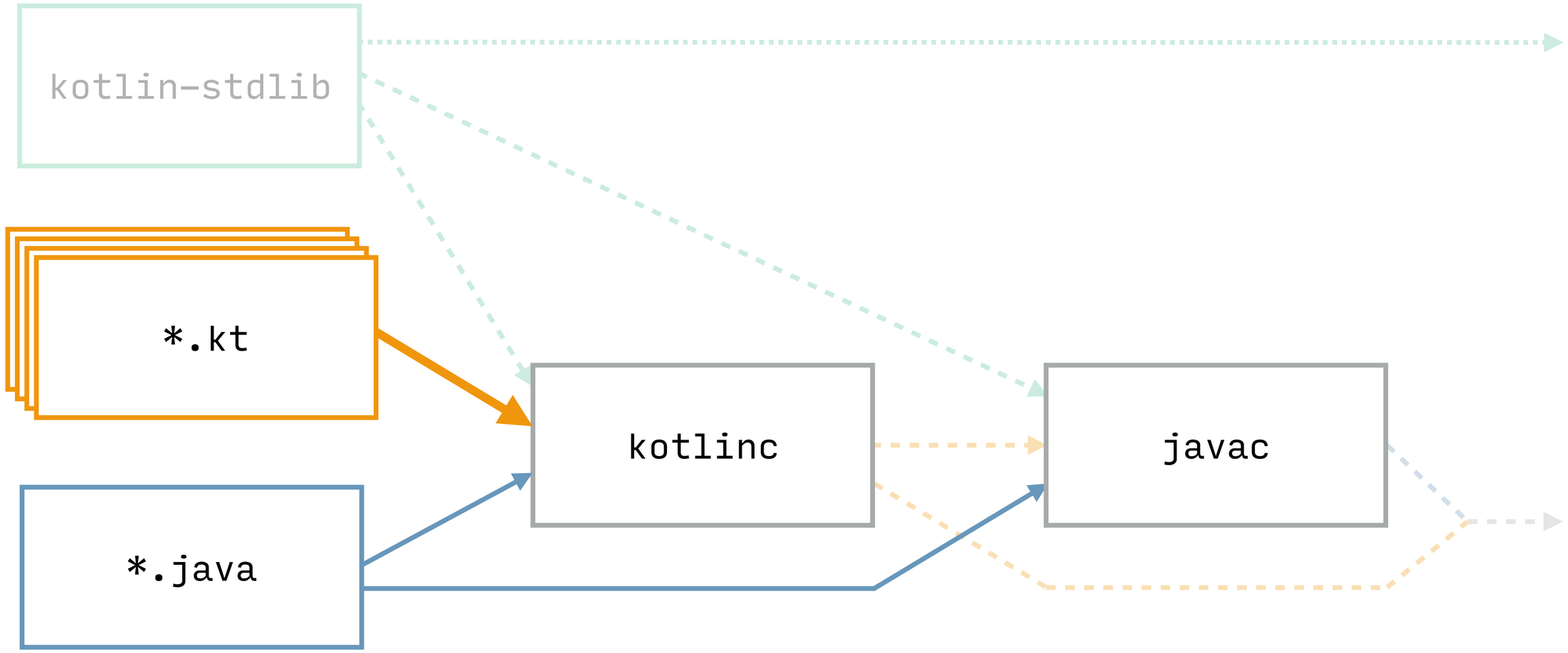

kotlin-stdlib
*.kt
*.java
kotlinc
javac

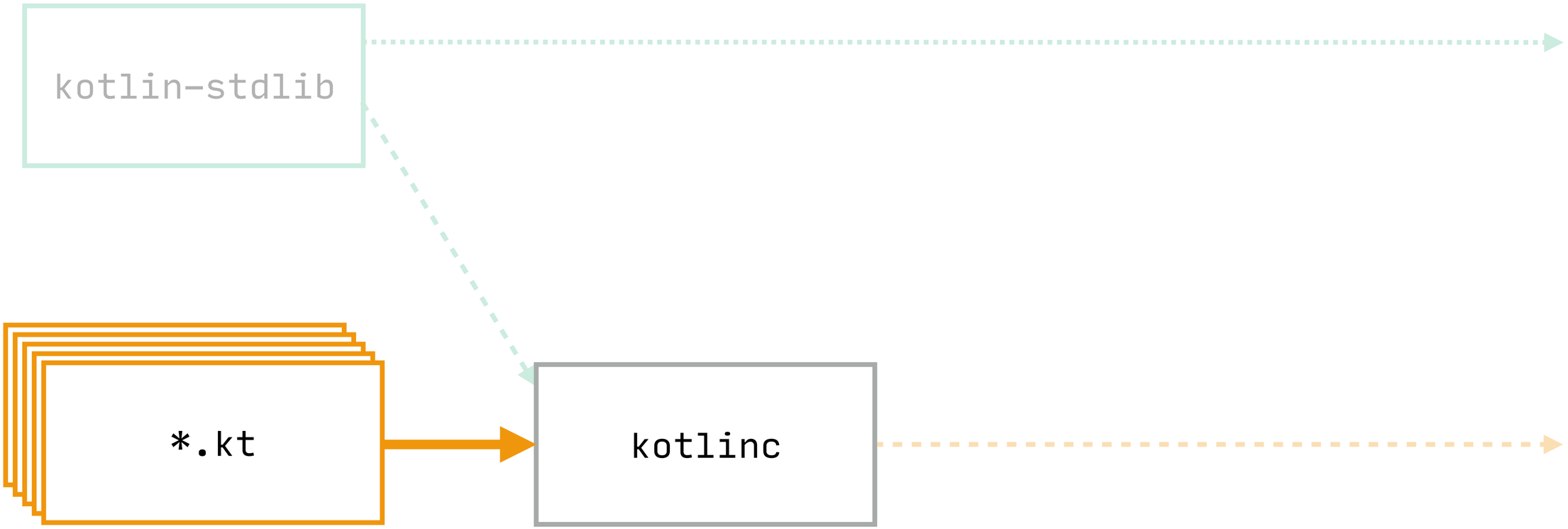

kotlin-stdlib
*.kt
*.java
kotlinc
javac

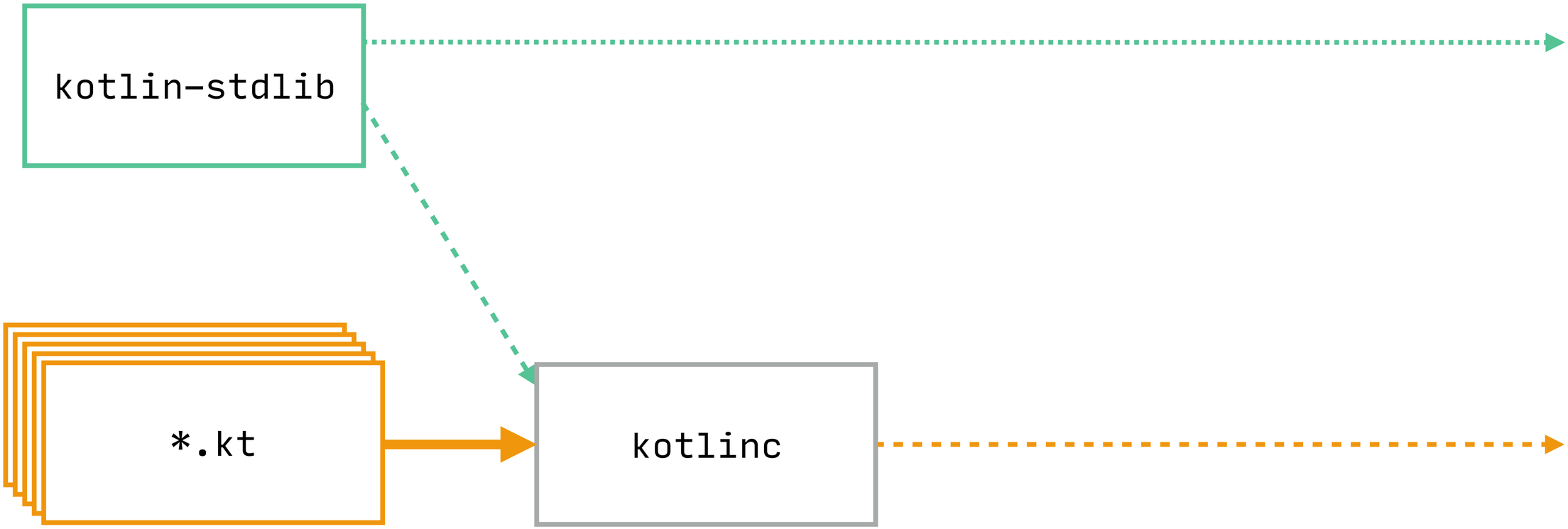

kotlin-stdlib
*.kt
*.java
kotlinc
javac

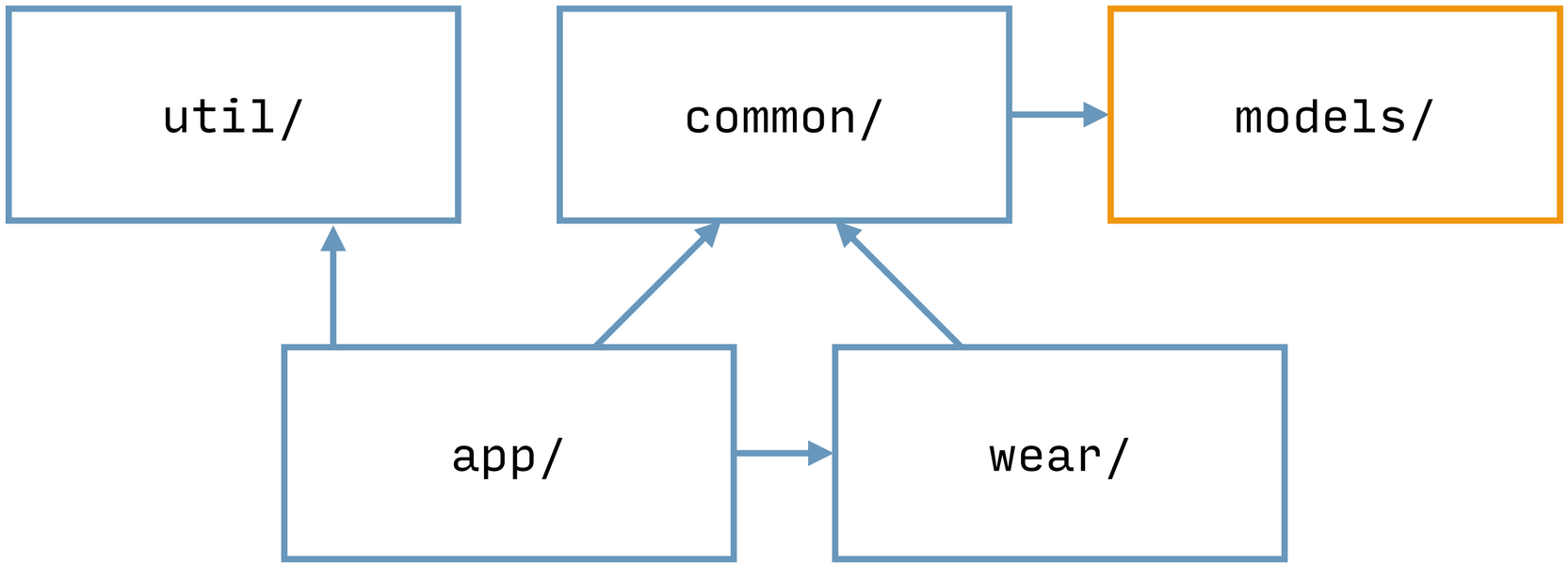

kotlin-stdlib
*.kt
*.java
kotlinc
javac

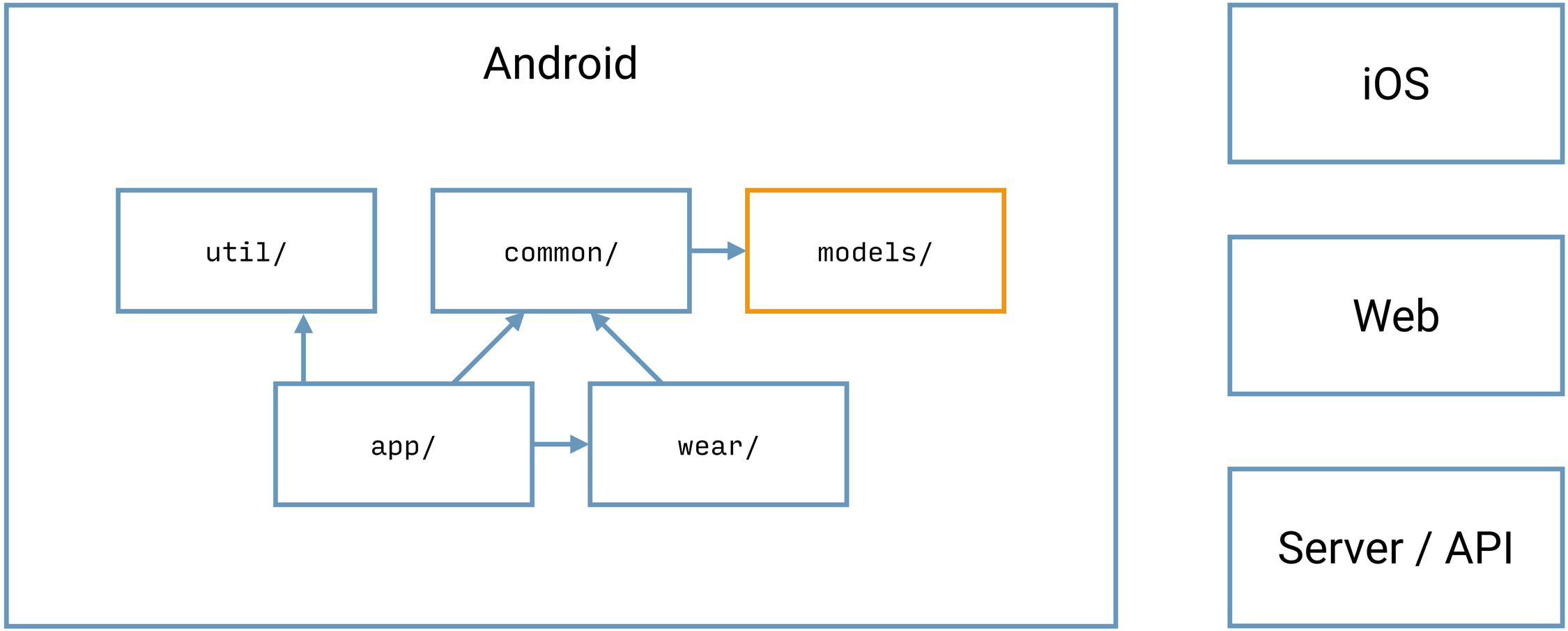

kotlin-stdlib
*.kt
*.java
kotlinc
javac

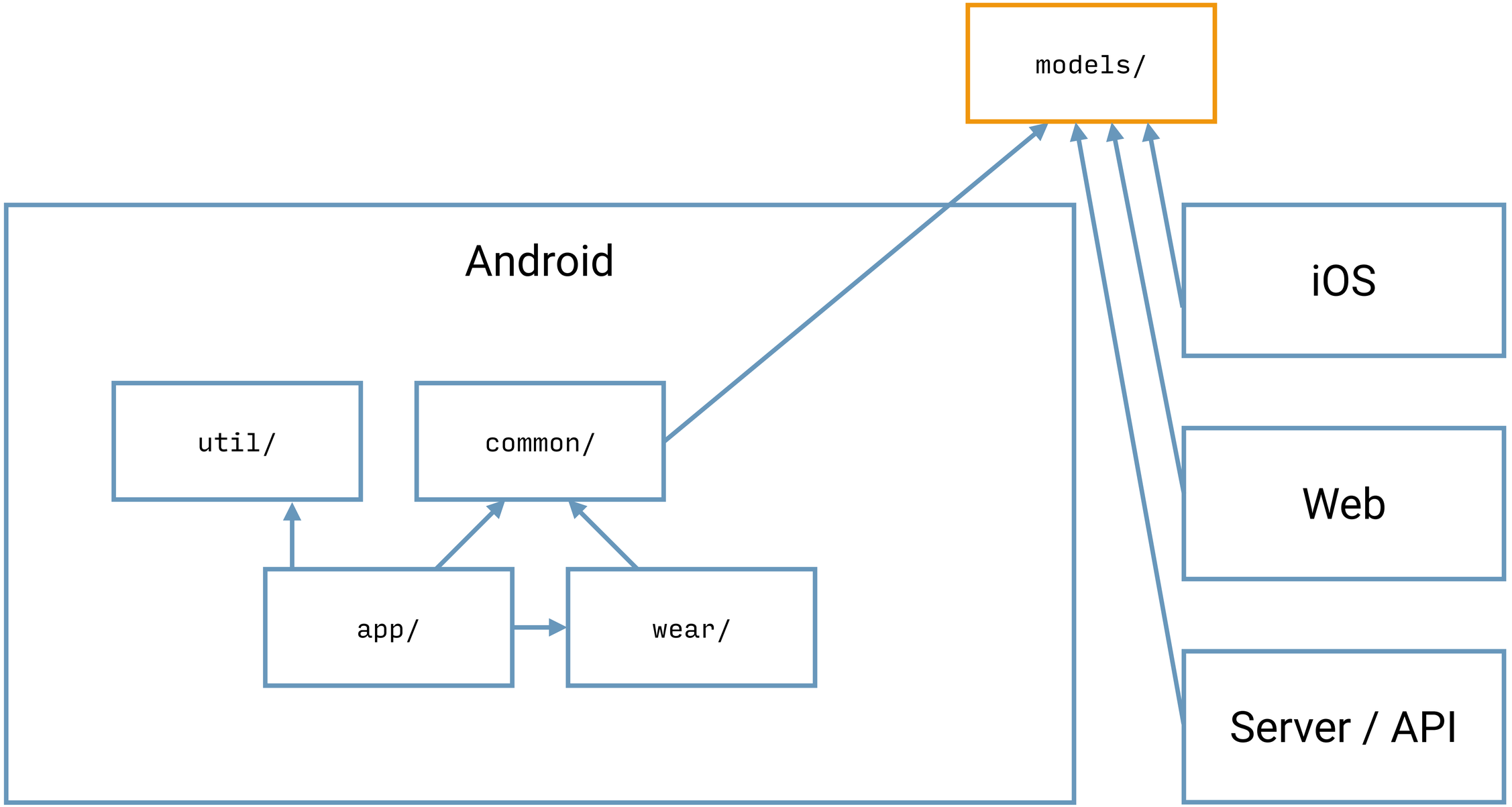

kotlin-stdlib
*.kt
*.java
kotlinc
javac

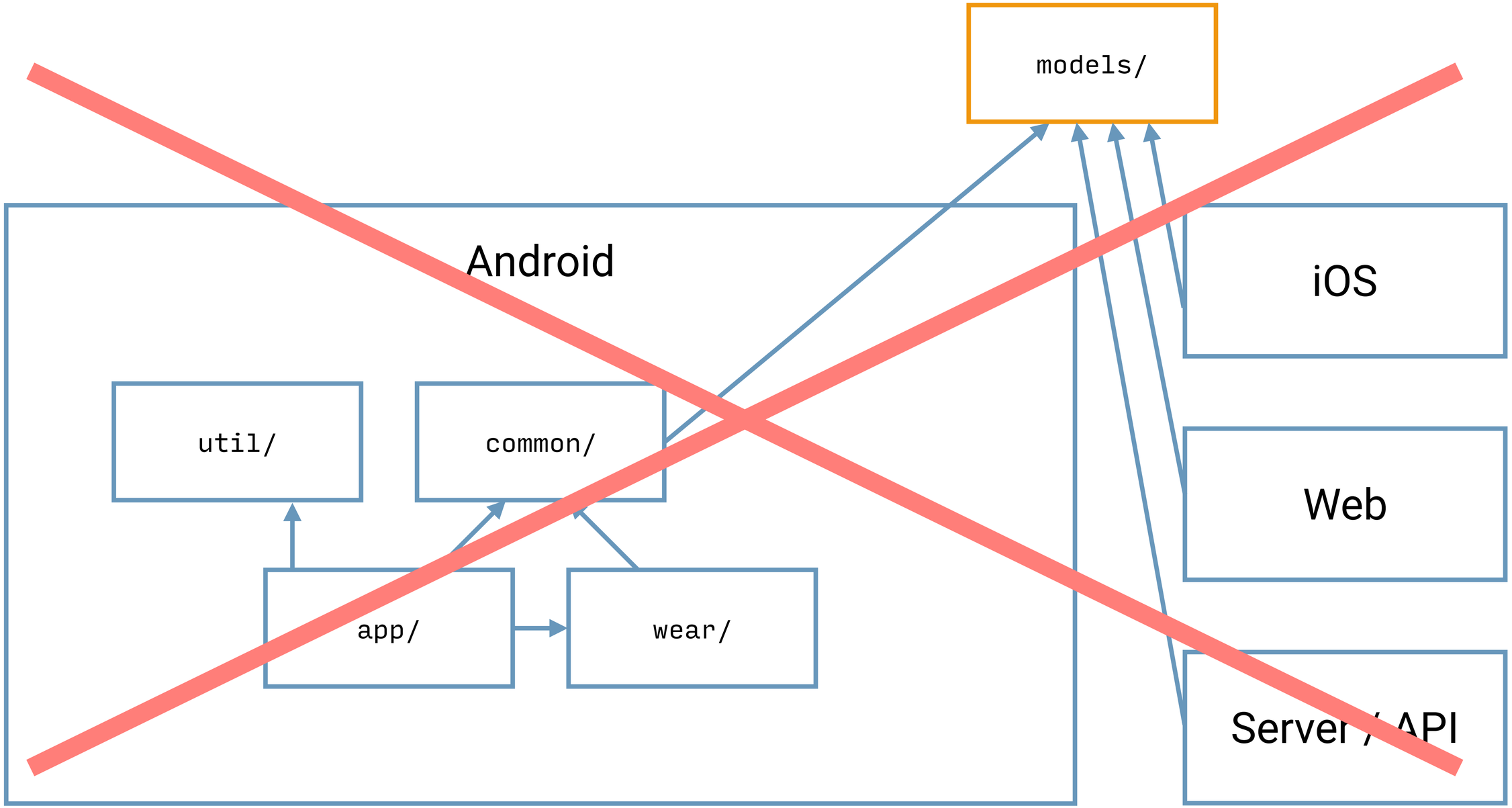

kotlin-stdlib
*.kt
kotlinc

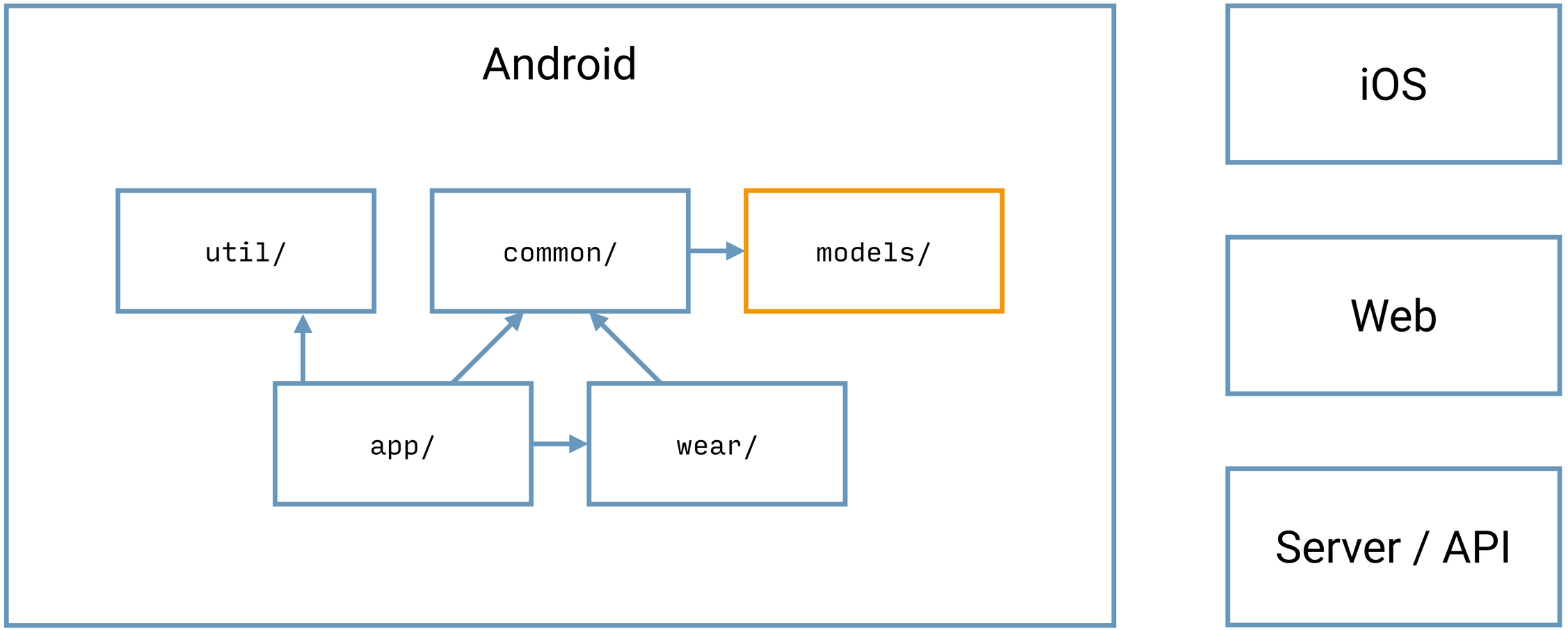

kotlin-stdlib
*.kt
kotlinc

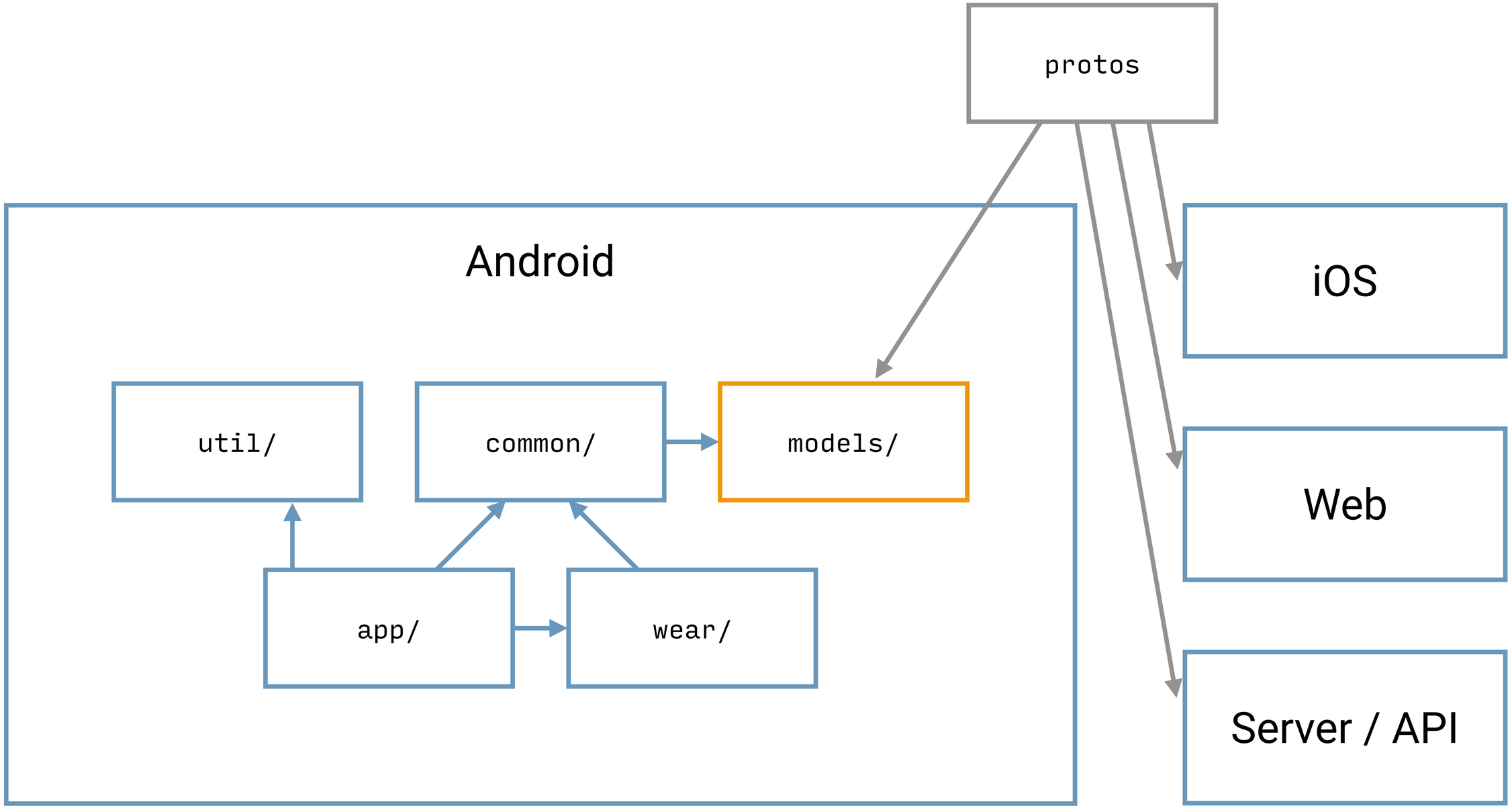

util/
common/
models/
app/
wear/

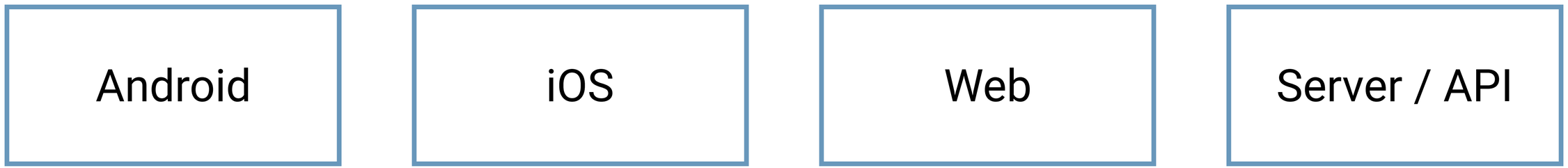

Android
util/
common/
models/
app/
wear/
iOS
Web
Server / API

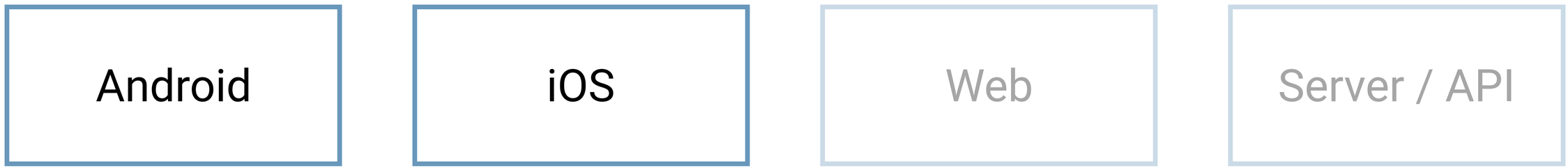

models/
Android
util/
common/
app/
wear/
iOS
Web
Server / API

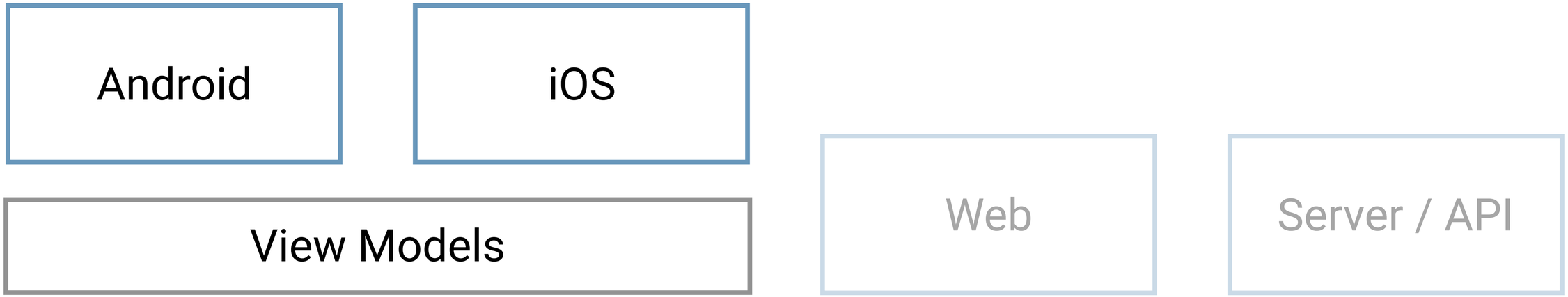

models/
Android
util/
common/
app/
wear/
iOS
Web
Server / API


Android
util/
common/
models/
app/
wear/
iOS
Web
Server / API


protos
Android
util/
common/
models/
app/
wear/
iOS
Web
Server / API

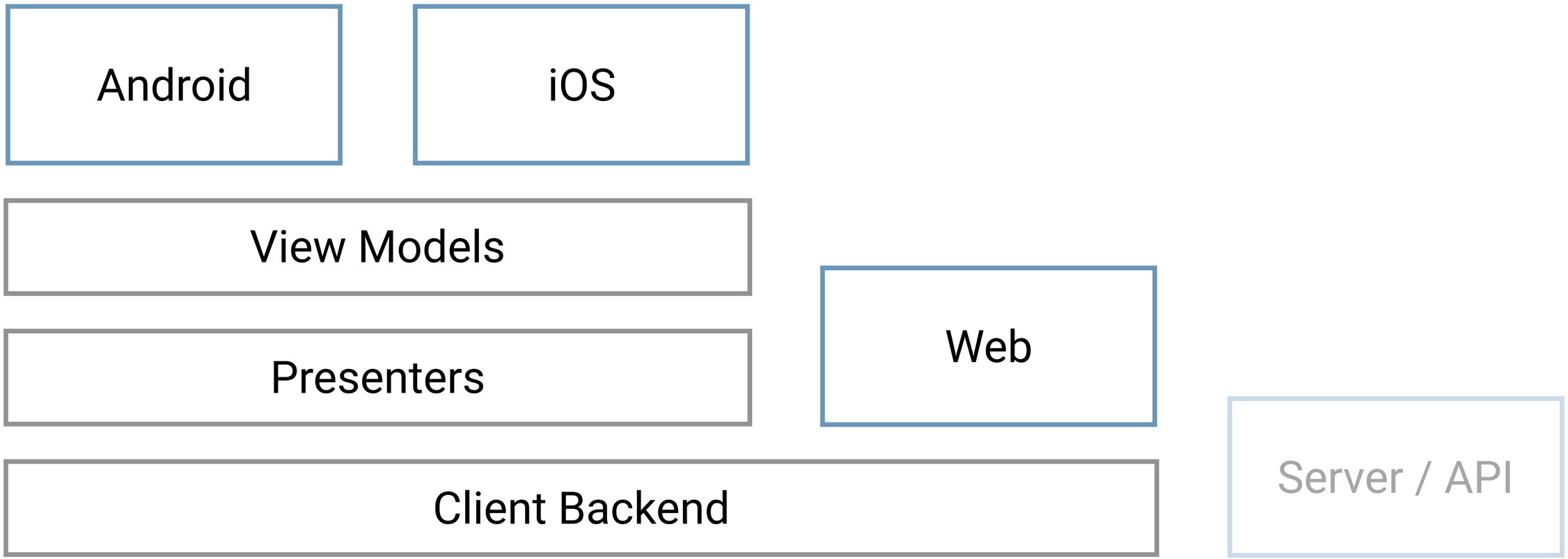

Android
iOS
Web
Server / API

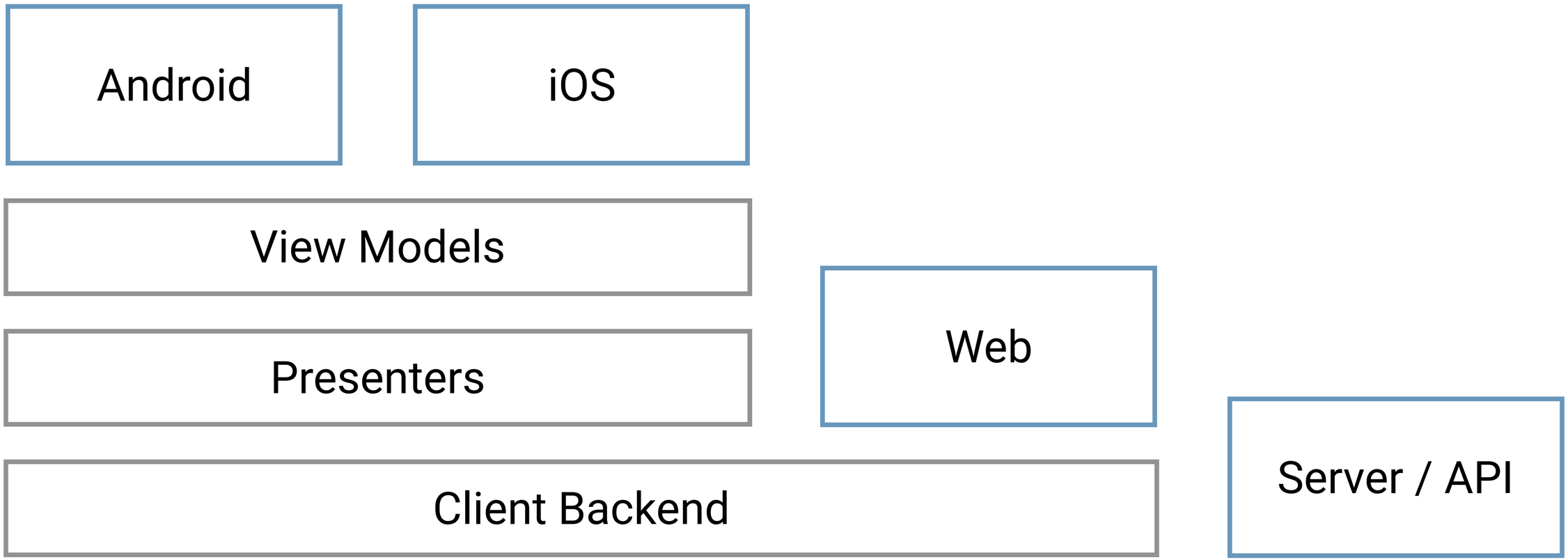

Android
iOS
Web
Server / API

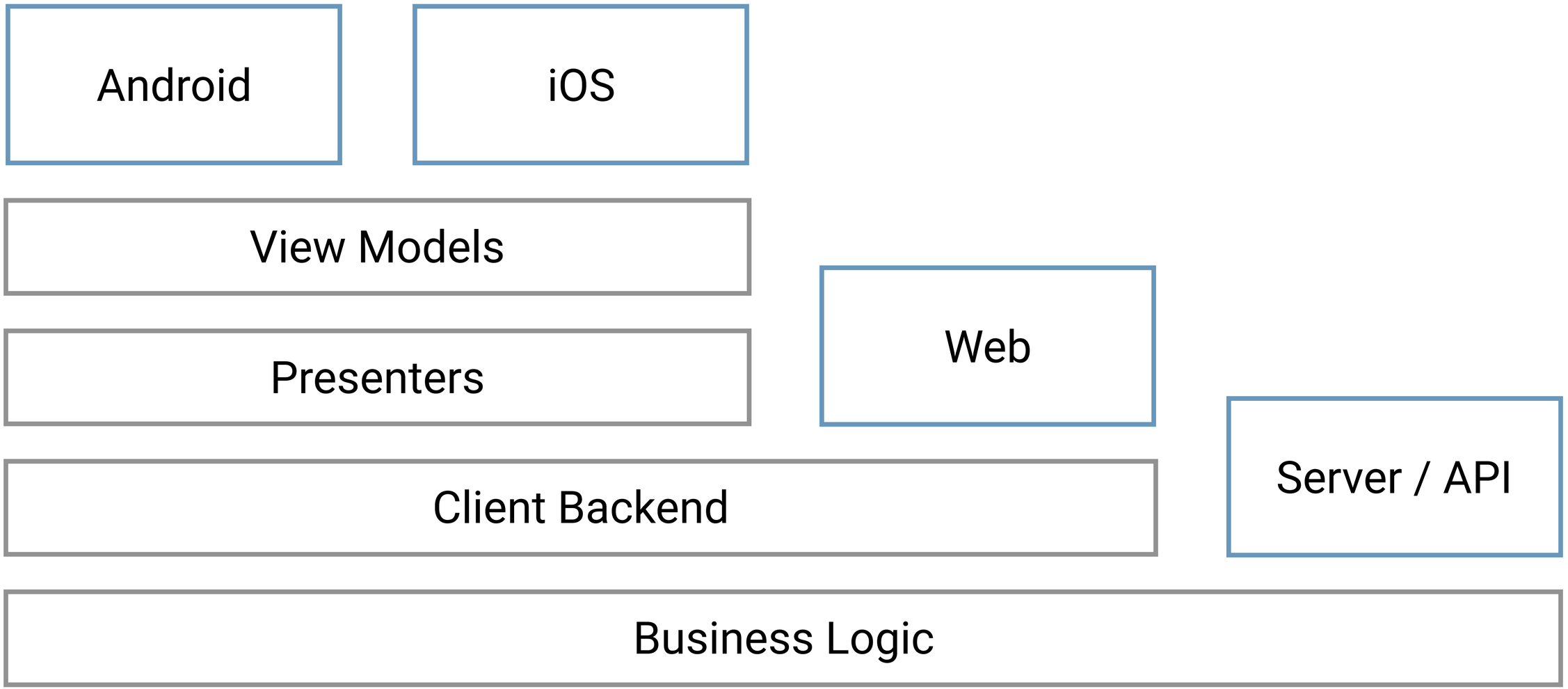

Android
iOS
View Models
Web
Server / API

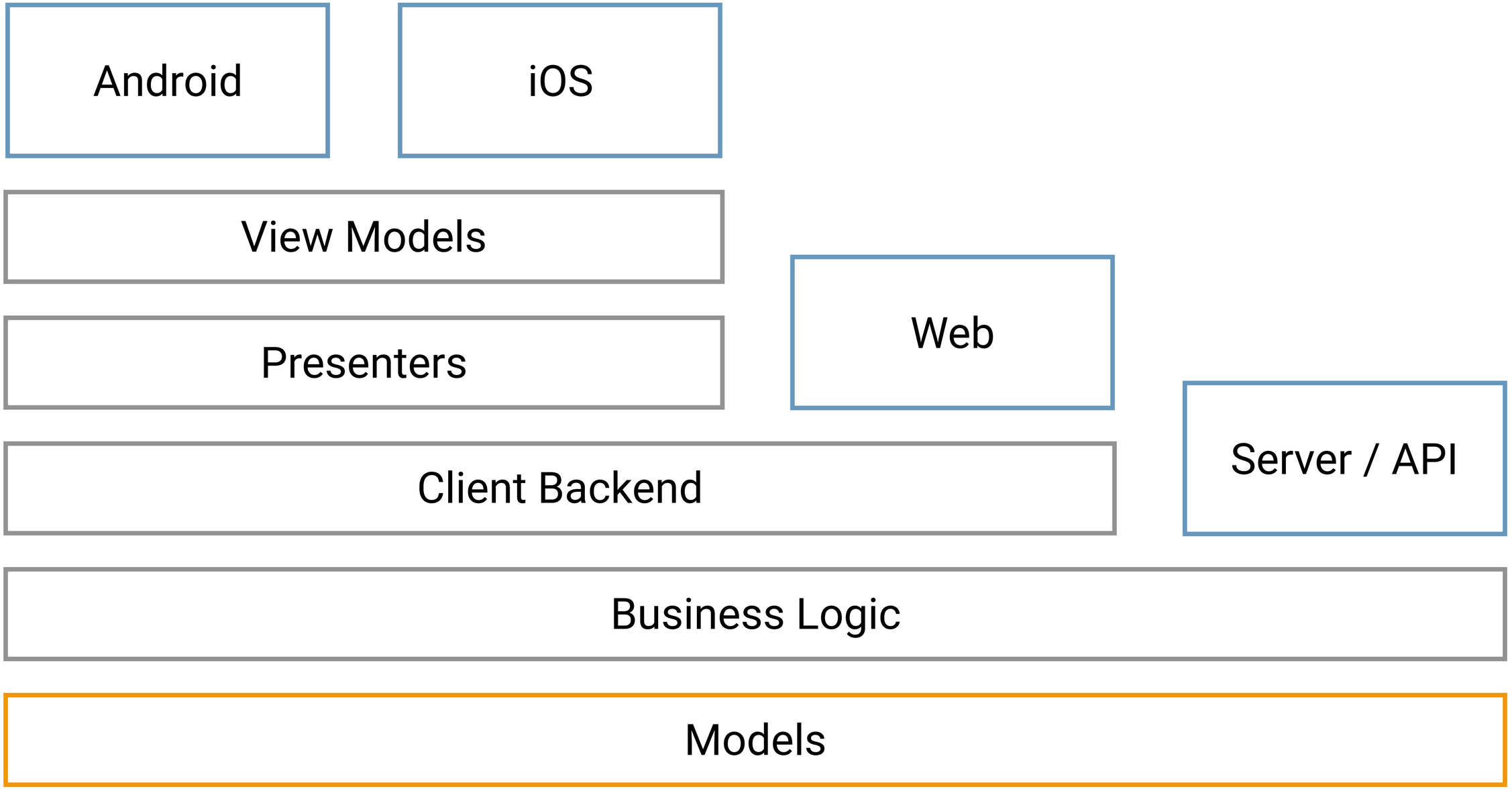

Android
iOS
View Models
Presenters
Web
Server / API

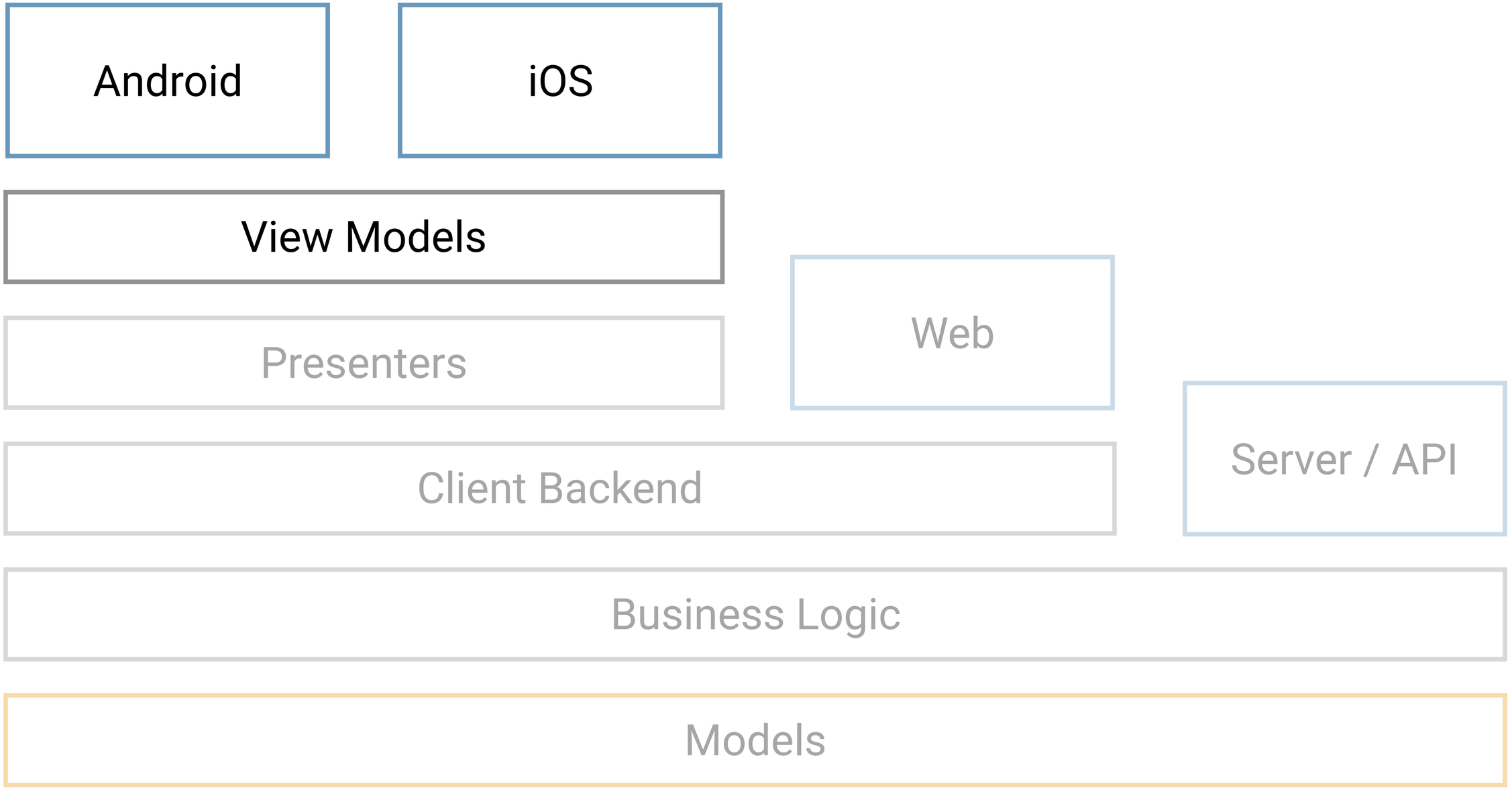

Android
iOS
View Models
Presenters
Web
Server / API


Android
iOS
View Models
Presenters
Web
Client Backend
Server / API


Android
iOS
View Models
Presenters
Web
Client Backend
Server / API


Android
iOS
View Models
Presenters
Web
Client Backend
Server / API
Business Logic


Android
iOS
View Models
Presenters
Web
Client Backend
Server / API
Business Logic
Models

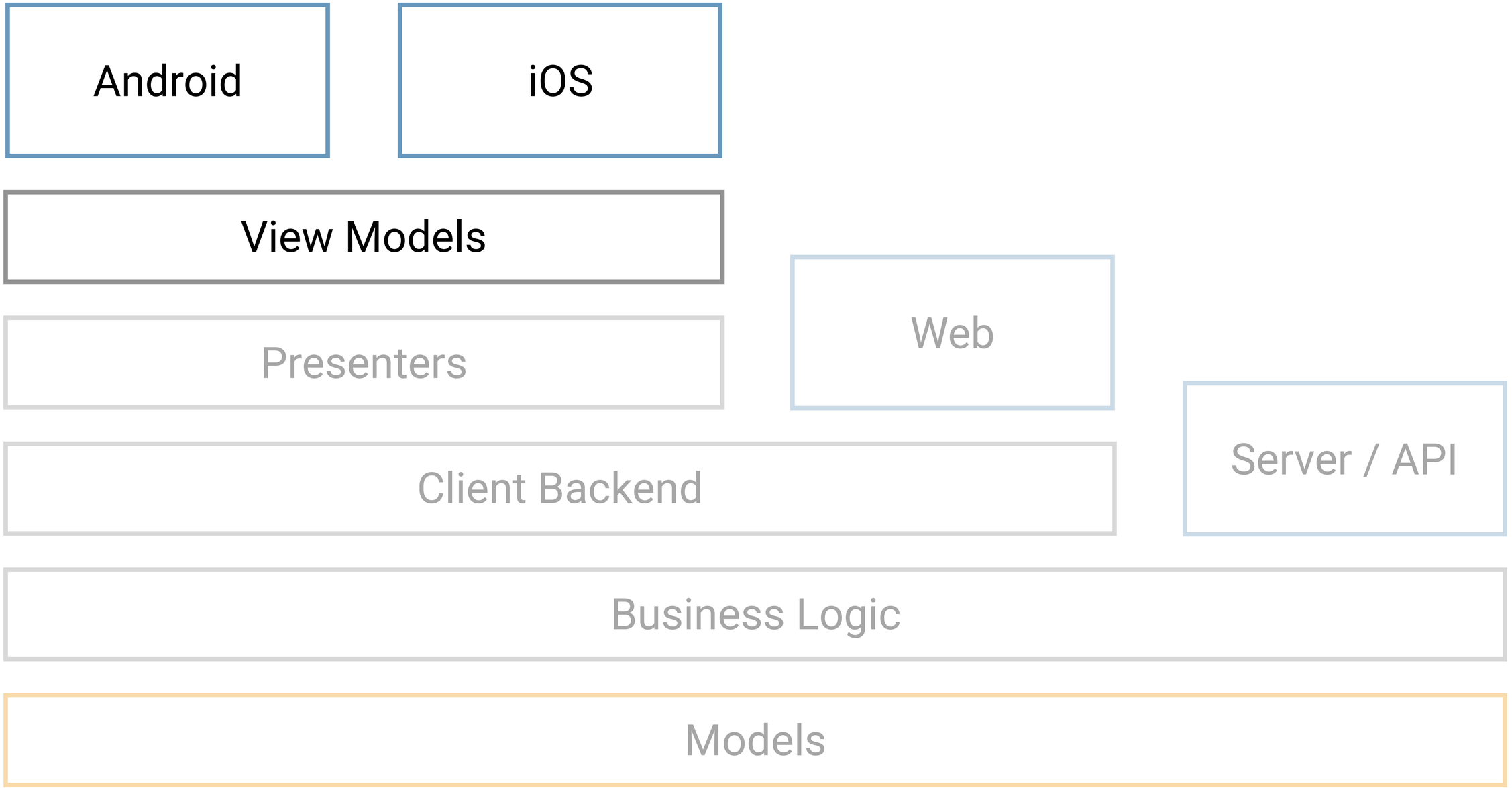

Android
iOS
View Models
Presenters
Web
Client Backend
Server / API
Business Logic
Models

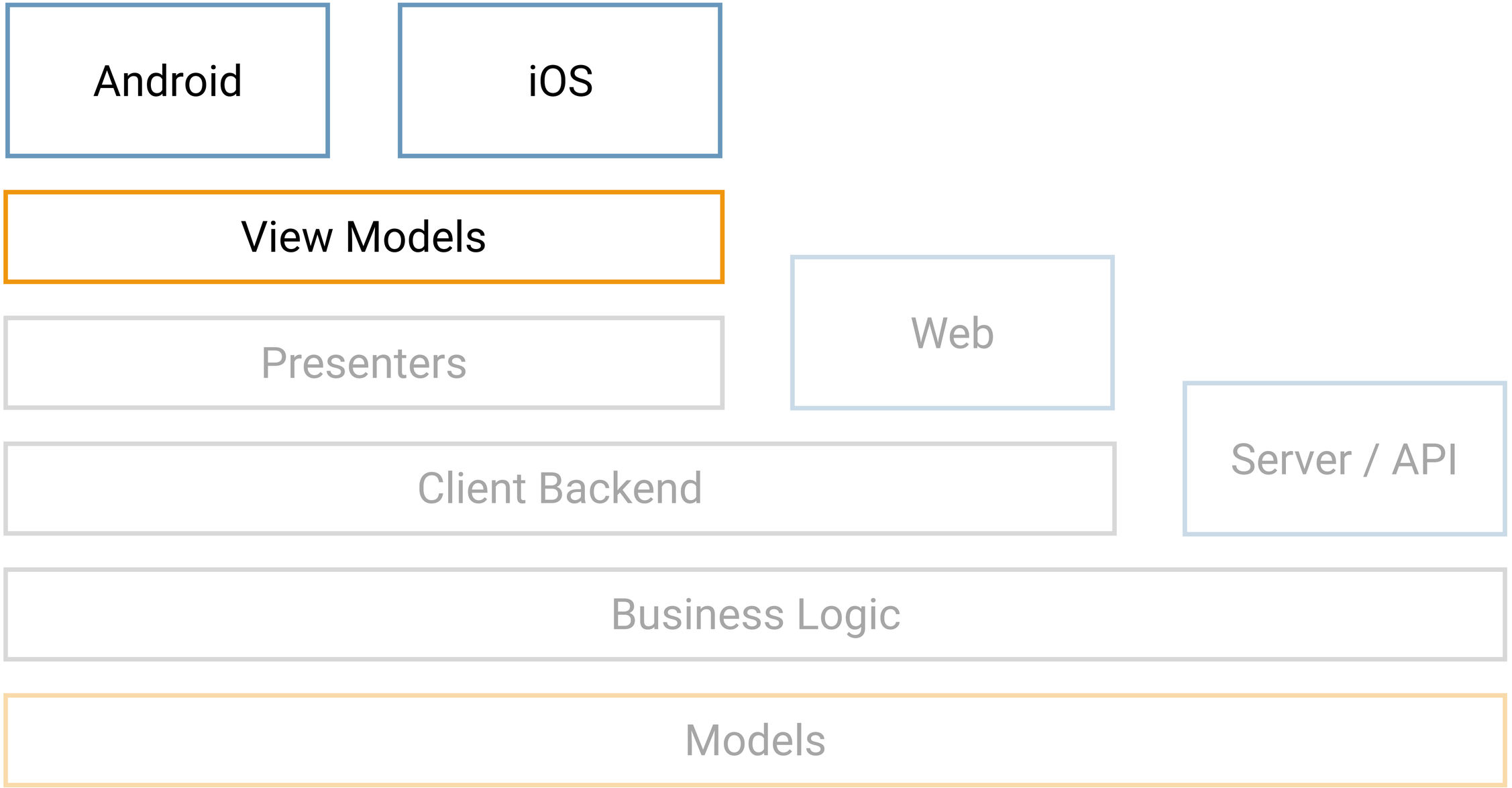
```
data class NewGameUiModel(
    val winTotal: Long,
    val lossTotal: Long
)
```

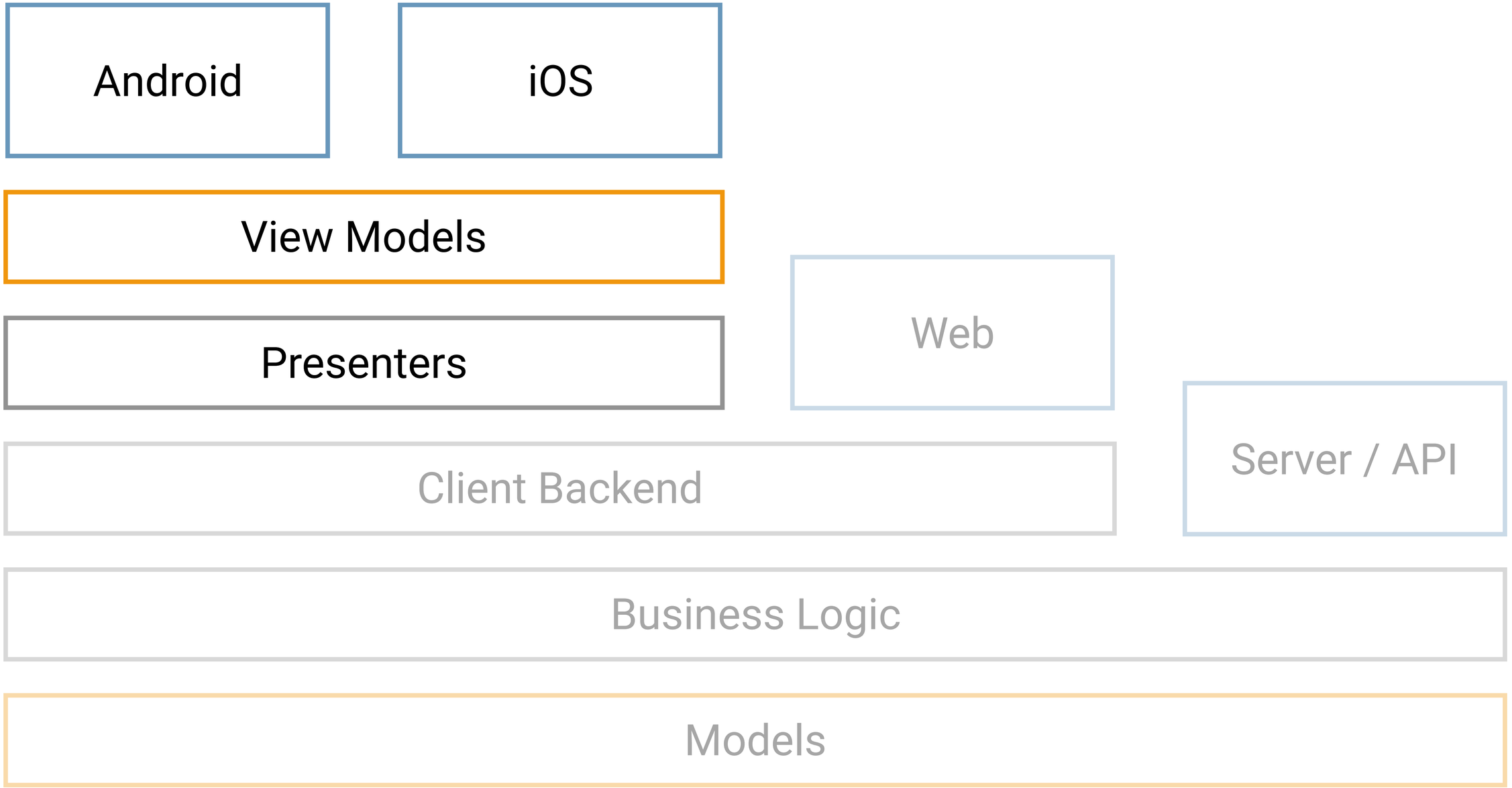
```
data class NewGameUiModel(
    val winTotal: Long,
    val lossTotal: Long
)

data class GameUiModel(
    val game: Game
)
```

```
data class NewGameUiModel(
    val winTotal: Long,
    val lossTotal: Long
)

data class GameUiModel(
    val game: Game
)
```

```
data class NewGameUiModel(
    val winTotal: Long,
    val lossTotal: Long
)

data class GameUiModel(
    val game: Game
)
```


Android
iOS
View Models
Presenters
Web
Client Backend
Server / API
Business Logic
Models


Android
iOS
View Models
Presenters
Web
Client Backend
Server / API
Business Logic
Models


Android
iOS
View Models
Presenters
Web
Server / API
Client Backend
Business Logic
Models

```
class NewGamePresenter {
  fun model(): NewGameUiModel {
  }
}
```

```
class NewGamePresenter(private val gameStore: GameStore) {
  fun model(): NewGameUiModel {
  }
}
```

```
class NewGamePresenter(private val gameStore: GameStore) {
  fun model(): NewGameUiModel {
    val totals = gameStore.totals()
    return NewGameUiModel(totals.wins, totals.losses)
  }
}
```

```
class GamePresenter {
  fun model(): GameUiModel {
  }
}
```

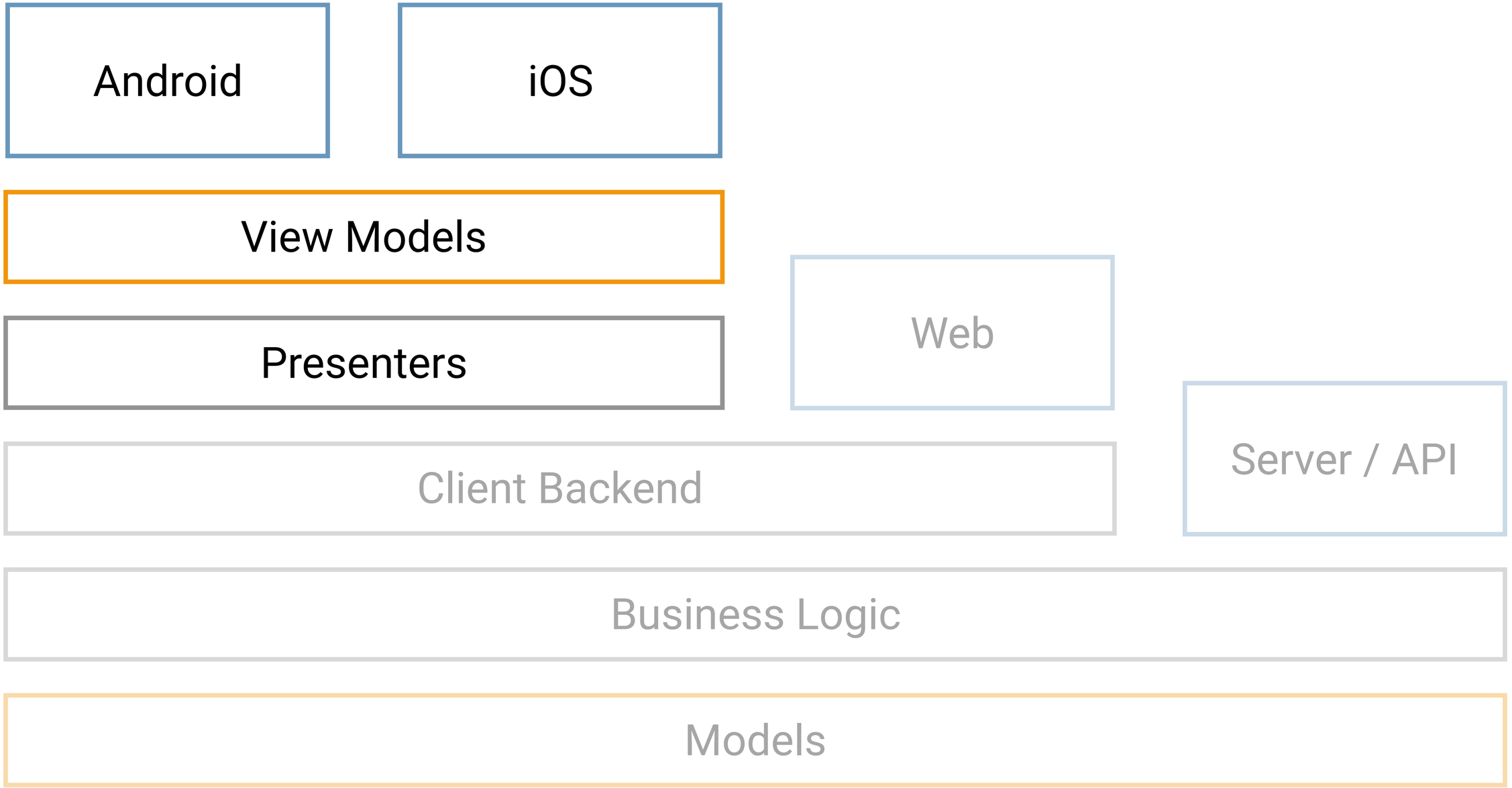
```
class GamePresenter(private val gameId: Long) {
  fun model(): GameUiModel {
  }
}
```

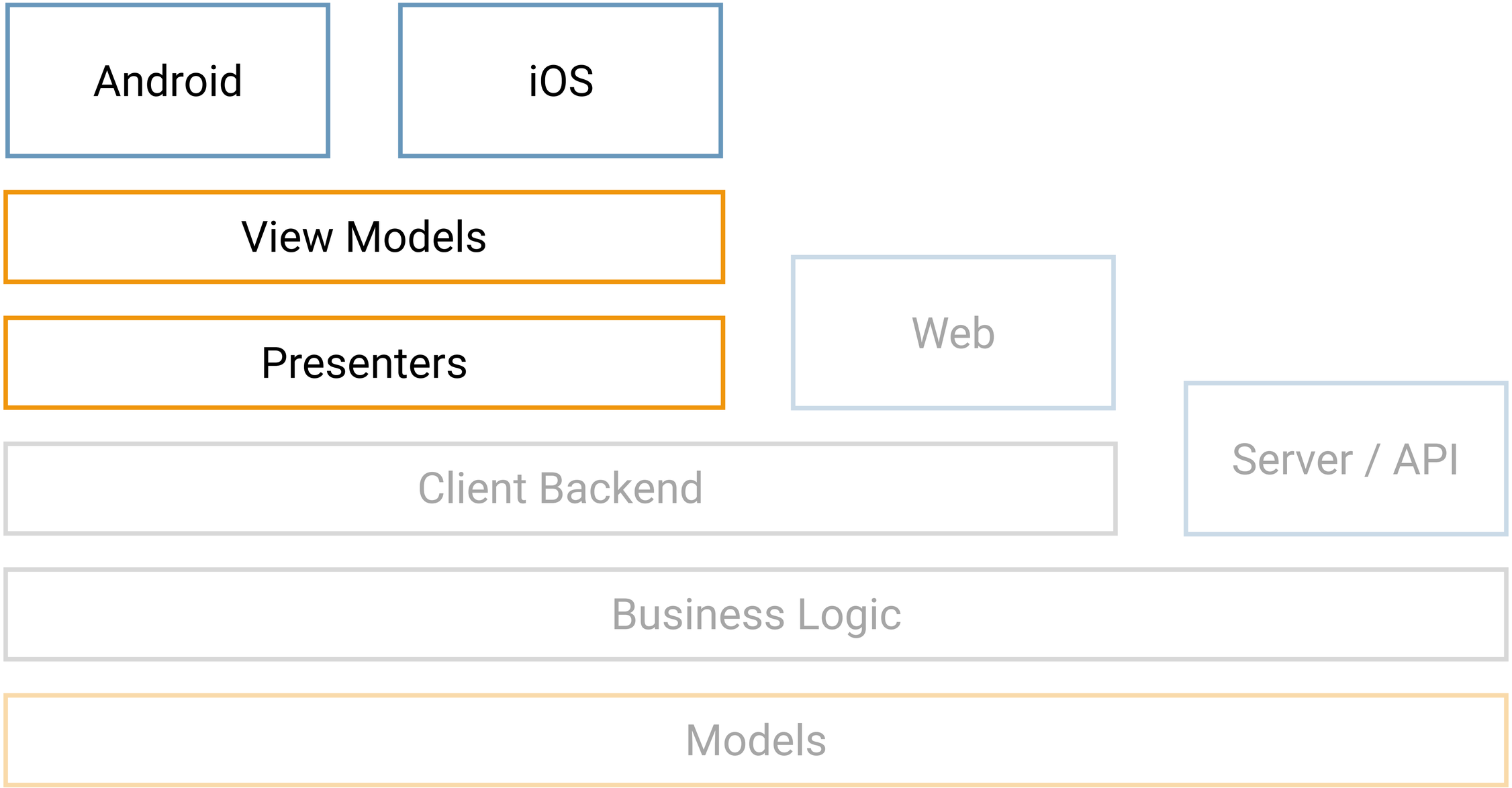
```
class GamePresenter(
    private val gameId: Long,
    private val gameStore: GameStore
) {
  fun model(): GameUiModel {
  }
}
```

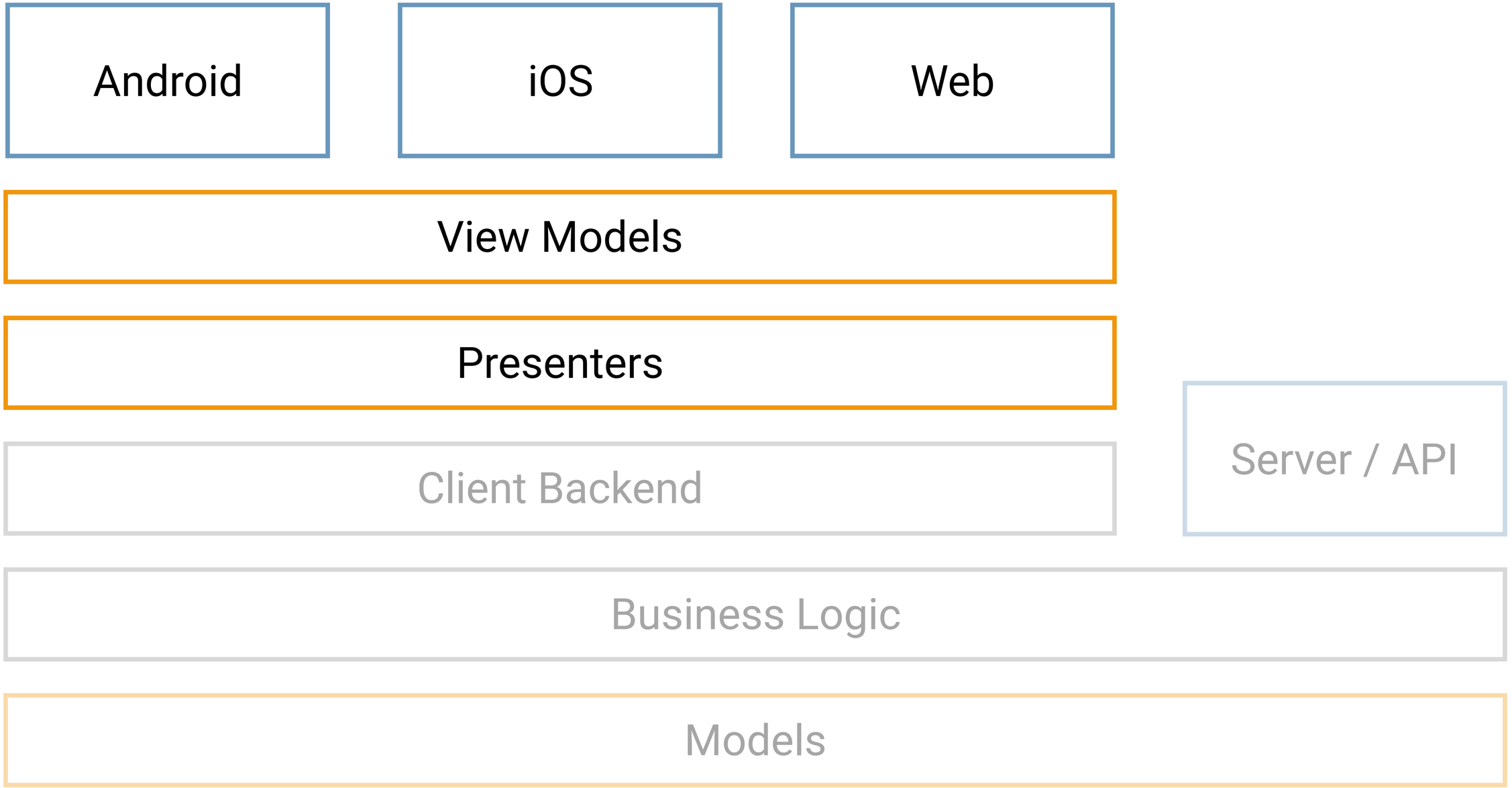
```
class GamePresenter(
    private val gameId: Long,
    private val gameStore: GameStore
) {
  fun models(): Observable<GameUiModel> {
  }
}
```

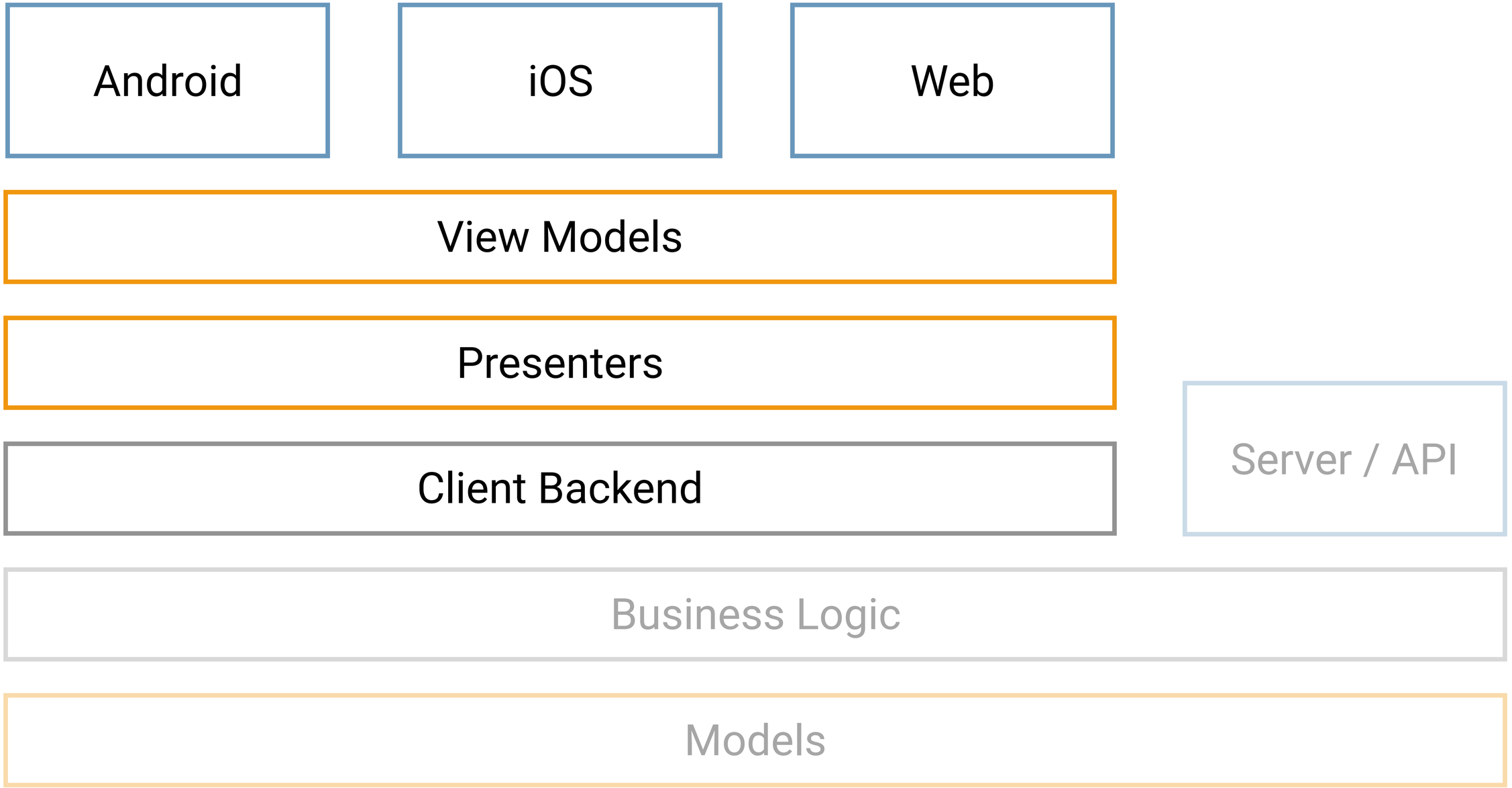
```
class GamePresenter(
    private val gameId: Long,
    private val gameStore: GameStore
) {
  fun move(row: Int, col: Int) {
  }

  fun models(): Observable<GameUiModel> {
  }
}
```

```
class GamePresenter(
    private val gameId: Long,
    private val gameStore: GameStore
) {
  fun models(events: Observable<UiEvent>): Observable<GameUiModel> {
  }

  sealed class UiEvent {
    data class Move(val row: Int, val col: Int): UiEvent()
    // ...
  }
}
```


Android
iOS
View Models
Presenters
Web
Server / API
Client Backend
Business Logic
Models


Android
iOS
View Models
Presenters
Web
Server / API
Client Backend
Business Logic
Models


Android
iOS
Web
View Models
Presenters
Client Backend
Server / API
Business Logic
Models

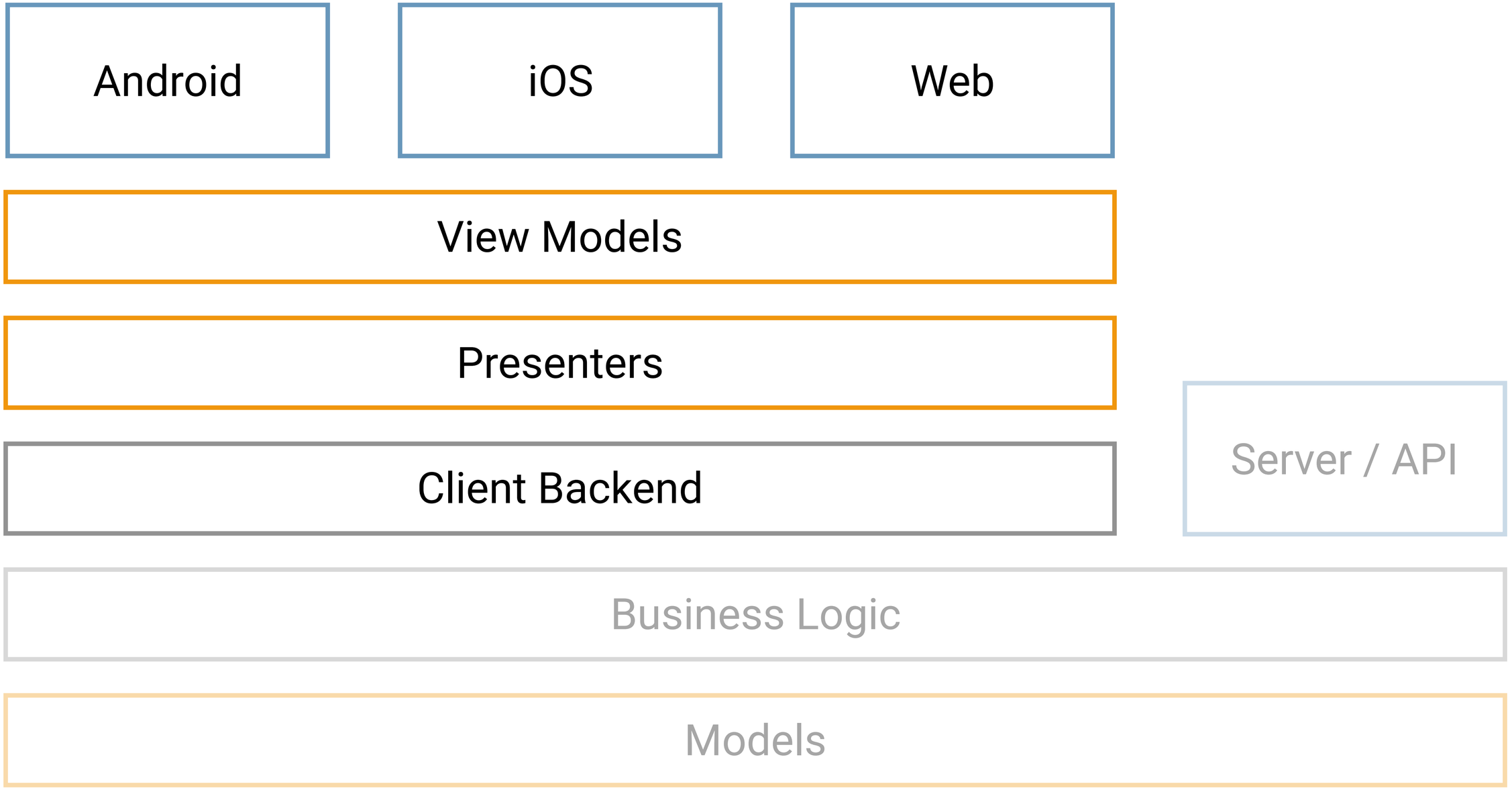

Android
iOS
Web
View Models
Presenters
Client Backend
Server / API
Business Logic
Models

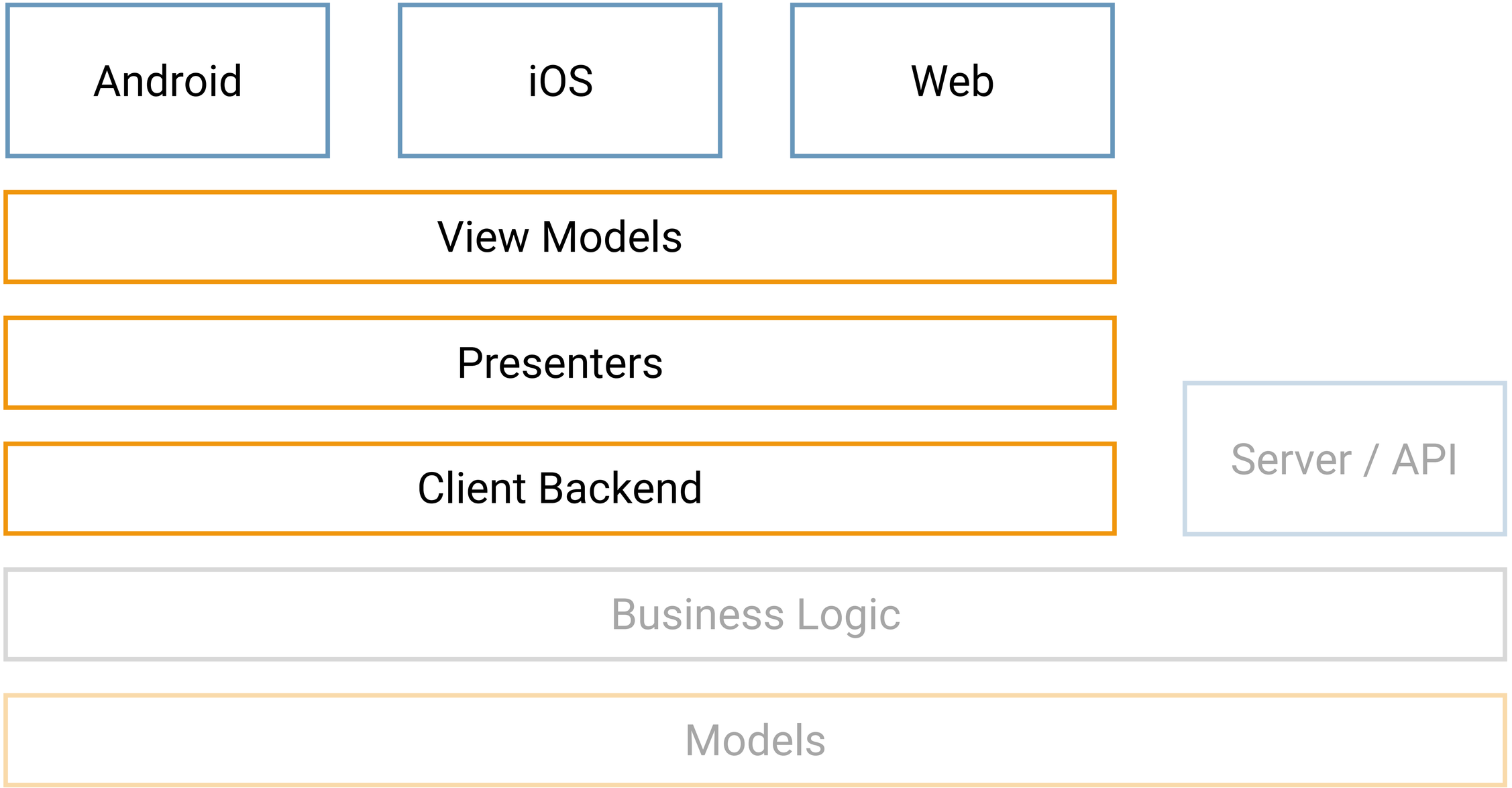
```
interface GameStore {
}
```

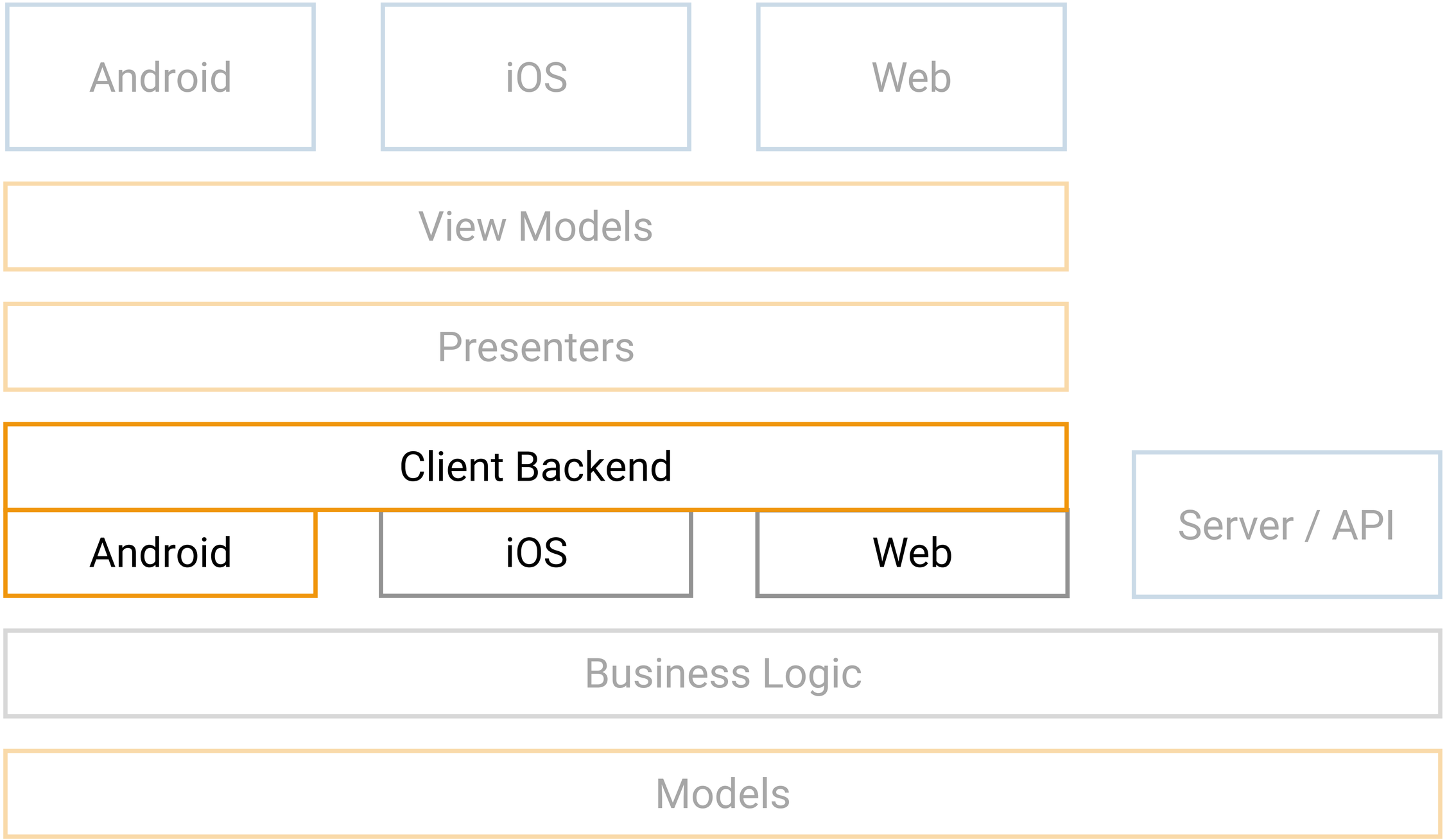
```
interface GameStore {
  fun totals(): Single<Totals>

  data class Totals(val wins: Long, val losses: Long)
}
```

```
interface GameStore {
  fun totals(): Single<Totals>
  fun game(id: Long): Observable<Game>

  data class Totals(val wins: Long, val losses: Long)
}
```

```
interface GameStore {
  fun totals(): Single<Totals>
  fun game(id: Long): Observable<Game>
  fun move(id: Long, row: Int, col: Int): Completable

  data class Totals(val wins: Long, val losses: Long)
}
```


Android
iOS
Web
View Models
Presenters
Client Backend
Server / API
Business Logic
Models


Android
iOS
Web
View Models
Presenters
Client Backend
Server / API
Business Logic
Models


Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models

```
class SqliteGameStore(private val db: SQLiteDatabase) : GameStore {
  override fun totals() = TODO()
  override fun game(id: Long) = TODO()
  override fun move(id: Long, row: Int, col: Int) = TODO()
}
```

```
class IosGameStore(private val db: CoreDataGameStore) : GameStore {
  override fun totals() = TODO()
  override fun game(id: Long) = TODO()
  override fun move(id: Long, row: Int, col: Int) = TODO()
}
```

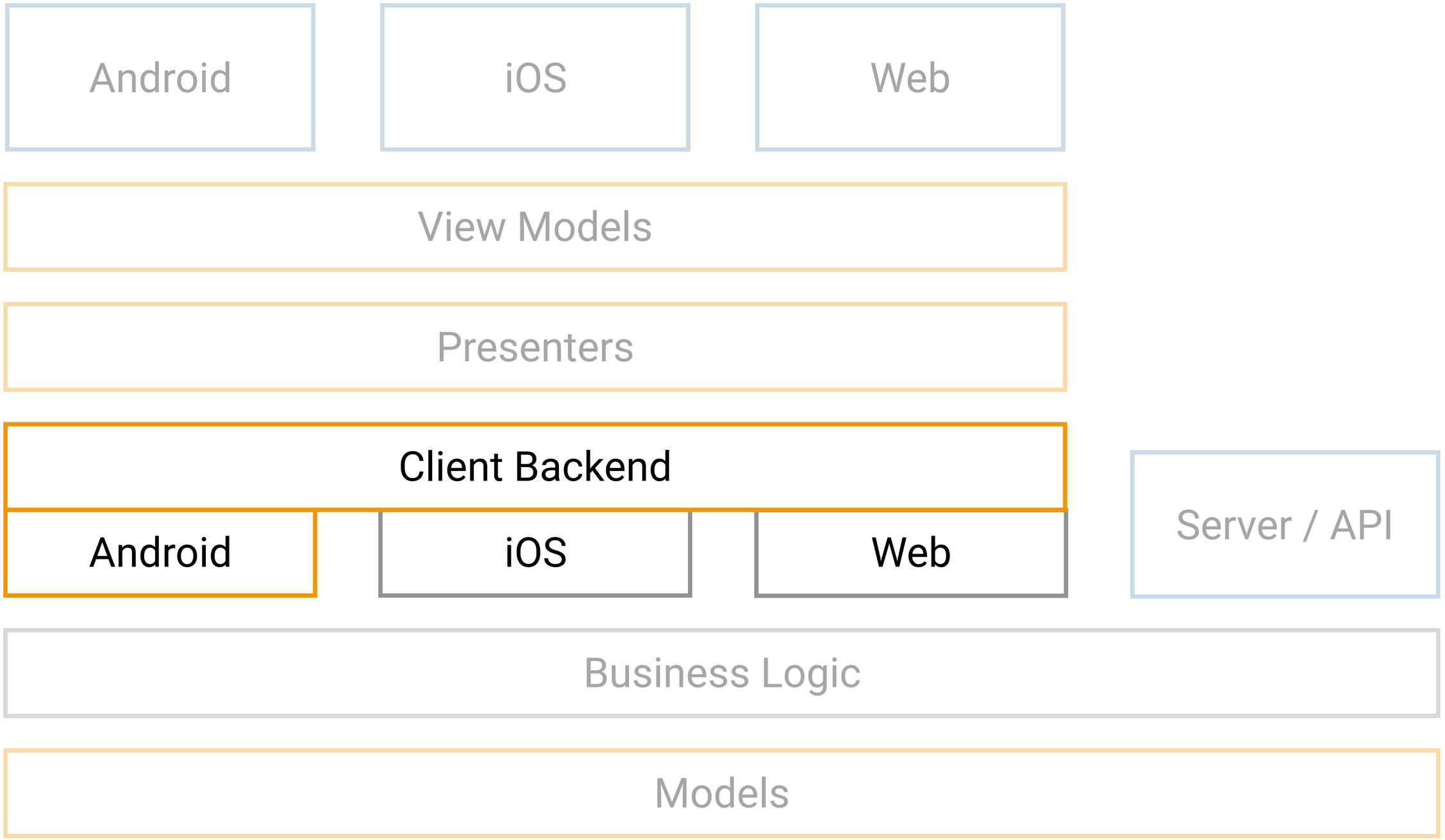
```
class IosGameStore(private val db: CoreDataGameStore) : GameStore {
  override fun totals() = TODO()
  override fun game(id: Long) = TODO()
  override fun move(id: Long, row: Int, col: Int) = TODO()
}
```

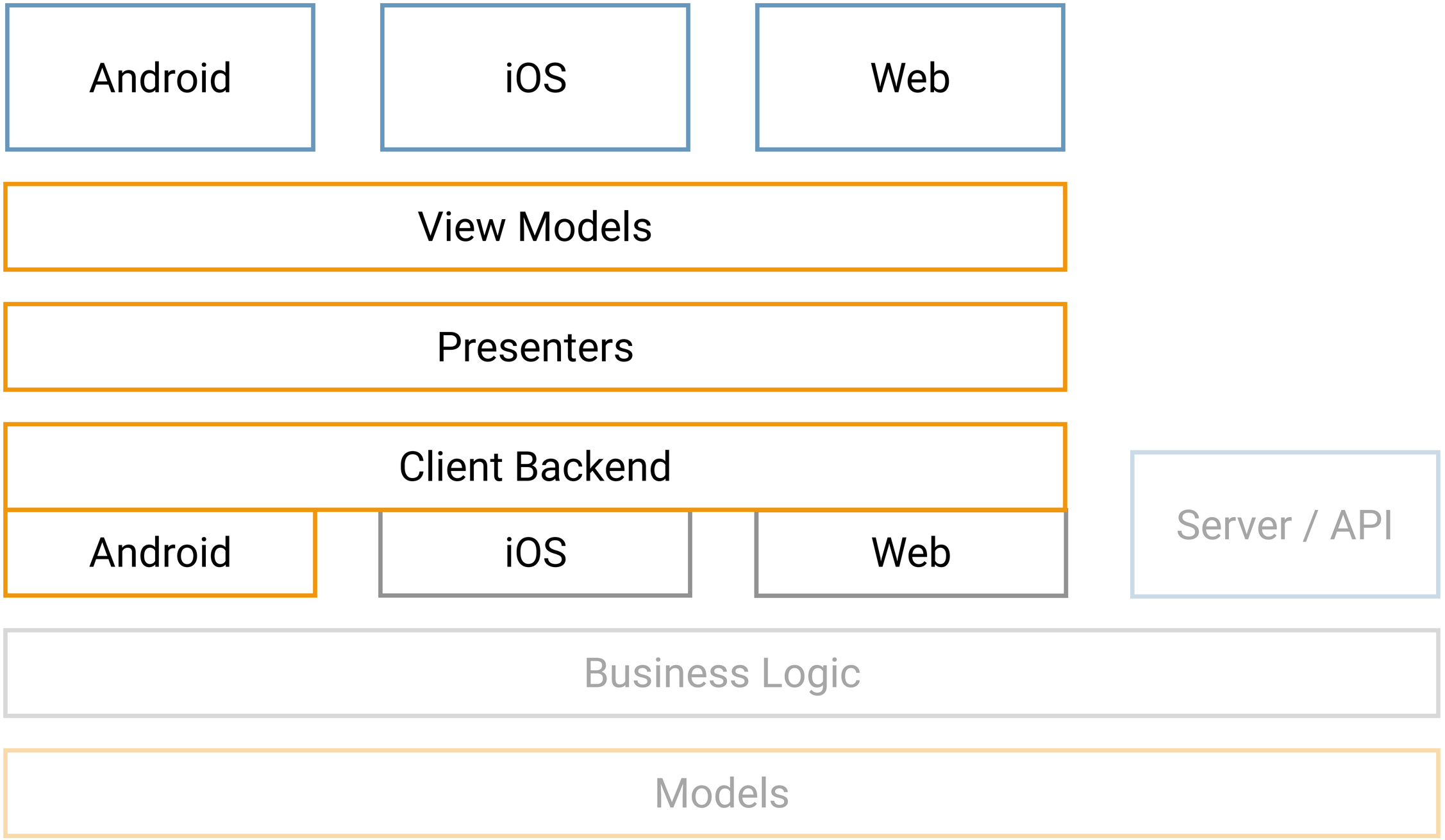
```
class IosGameStore(private val db: CoreDataGameStore) : GameStore {
  override fun totals() = TODO()
  override fun game(id: Long) = TODO()
  override fun move(id: Long, row: Int, col: Int) = TODO()
}
```

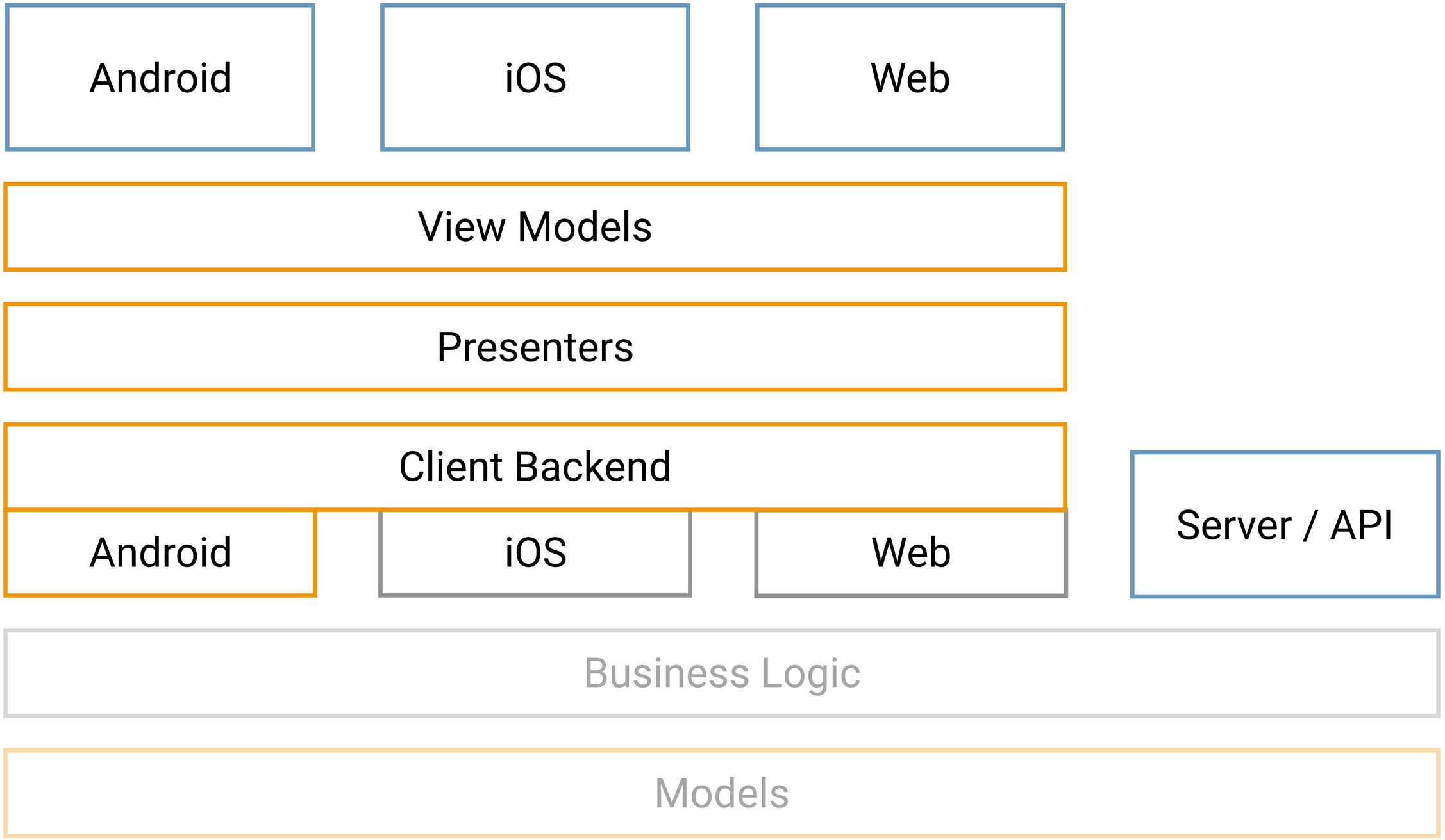
```
// tictactoe.def
headers = game_store.h
```

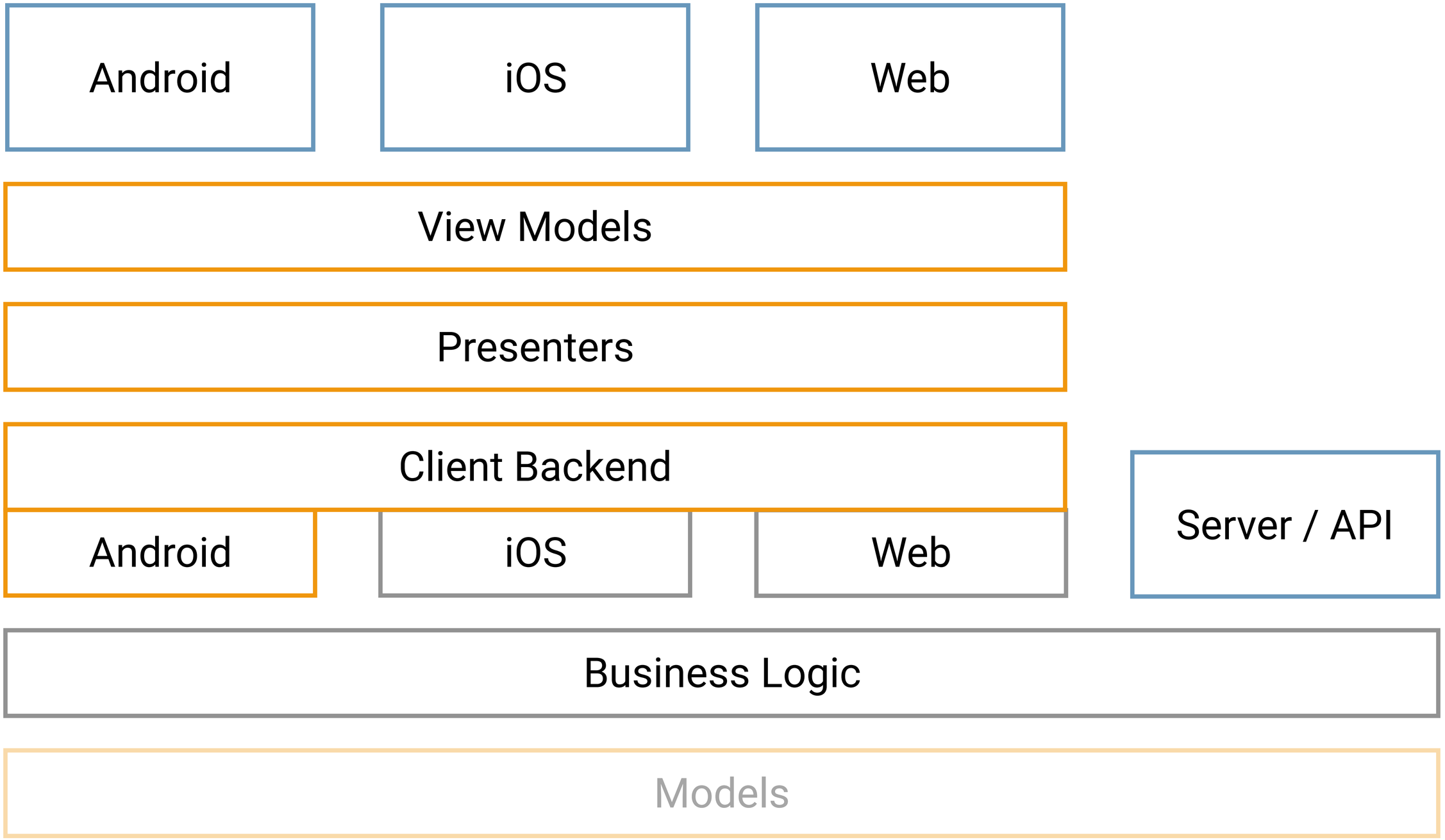
```
class StorageGameStore(private val store: Storage) : GameStore {
  override fun totals() = TODO()
  override fun game(id: Long) = TODO()
  override fun move(id: Long, row: Int, col: Int) = TODO()
}
```

```
import org.w3c.dom.Storage

class StorageGameStore(private val store: Storage) : GameStore {
  override fun totals() = TODO()
  override fun game(id: Long) = TODO()
  override fun move(id: Long, row: Int, col: Int) = TODO()
}
```

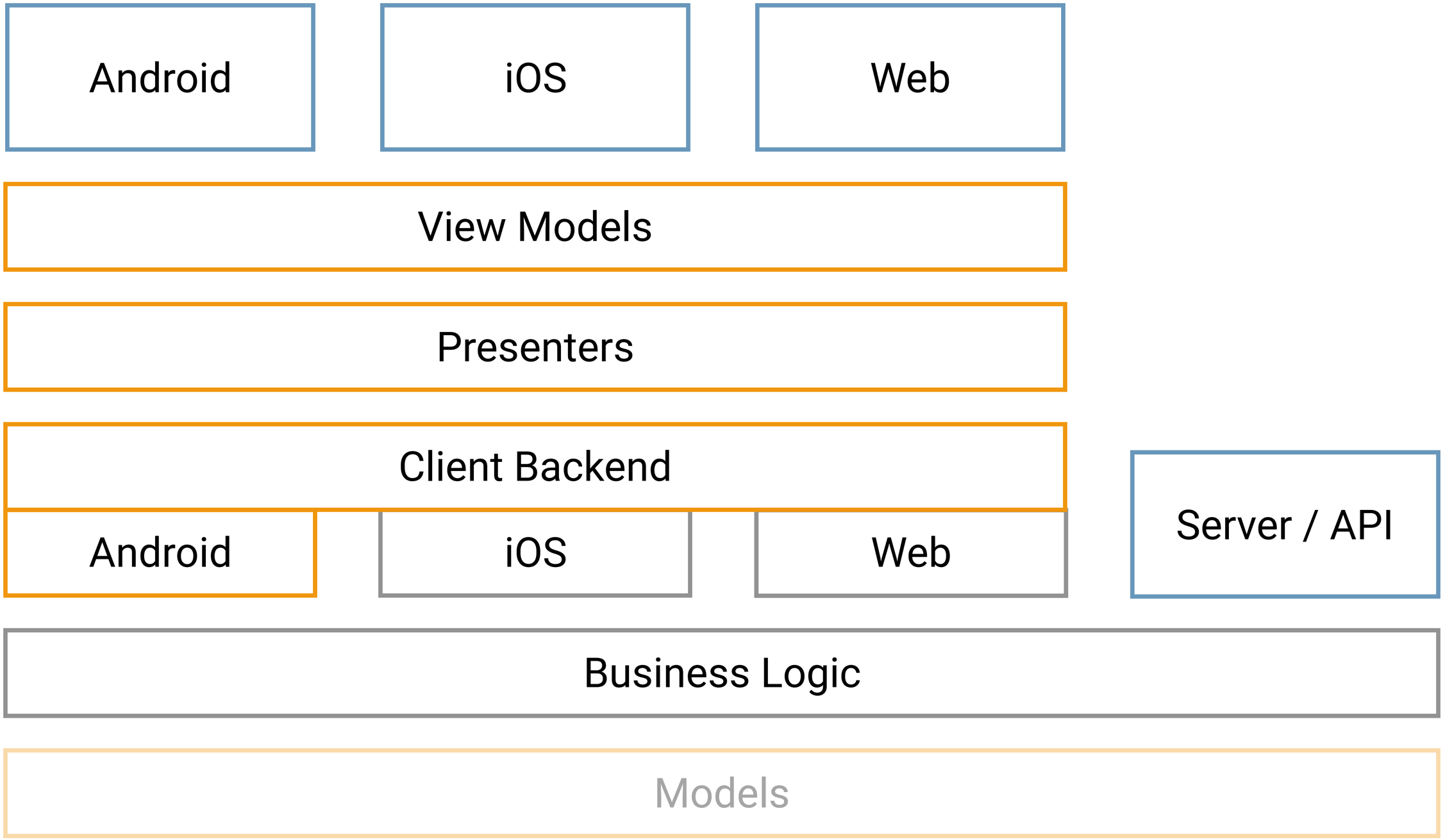

Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models

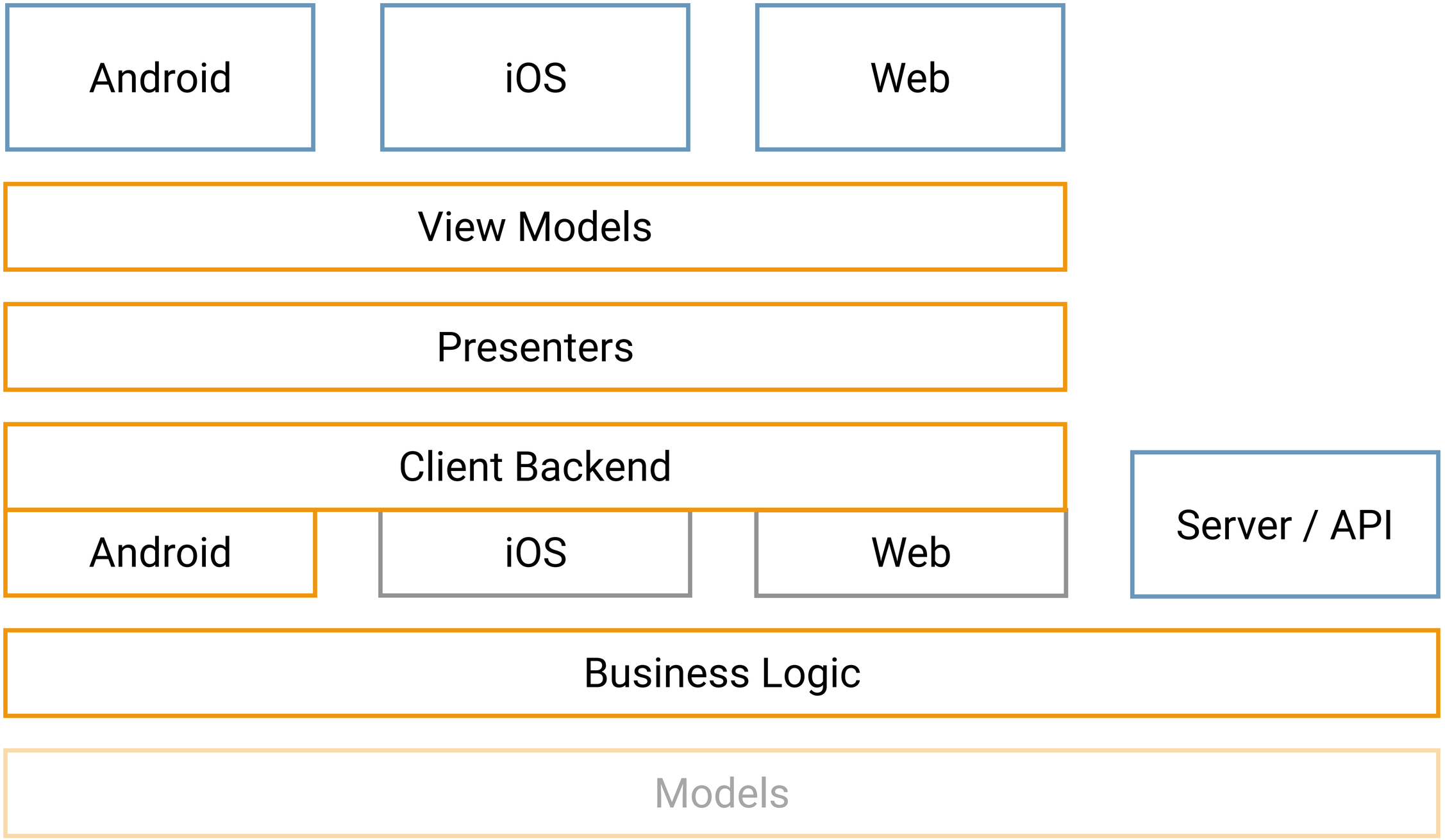

Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models

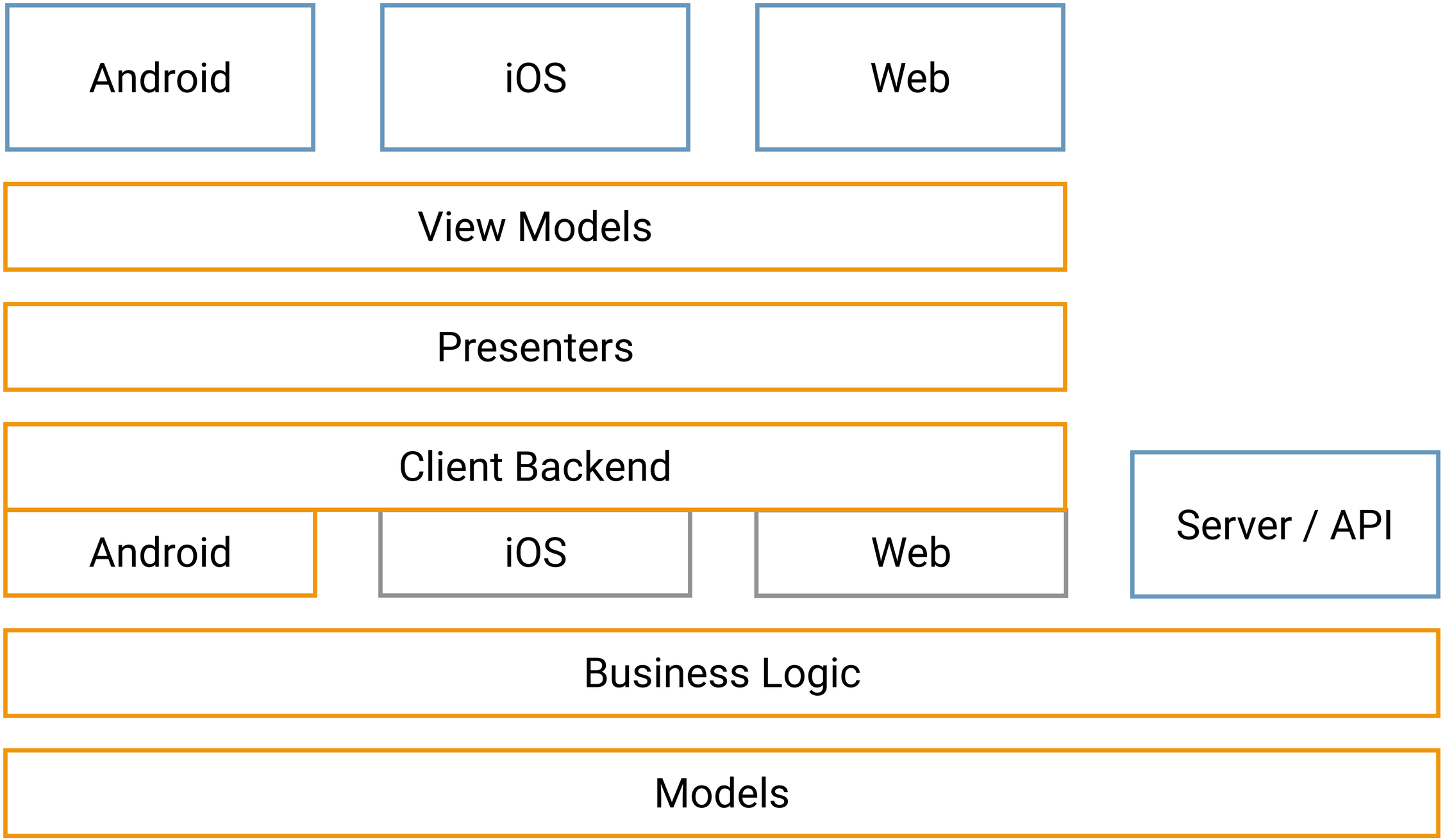

Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models

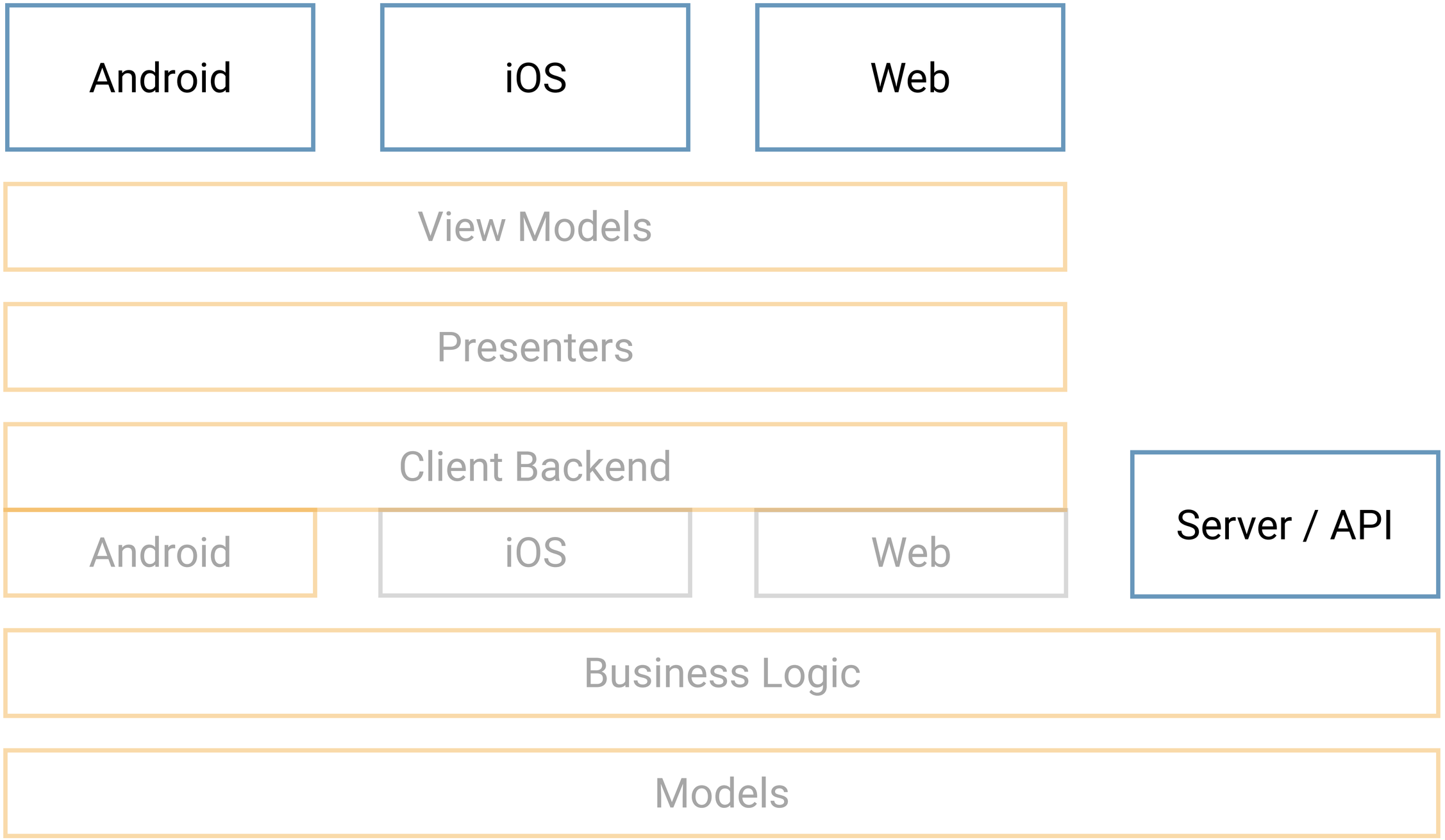

Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models

```
object TicTacToeLogic {
  fun validateMove(
      game: Game, player: Player, row: Int, col: Int): Boolean {
    when (game.state) {
      State.PLAYER_1_MOVE -> require(game.player1 == player)
      State.PLAYER_2_MOVE -> require(game.player2 == player)
      else -> error("Game is over")
    }
    return game.board[row][col] == null
  }

  fun nextState(game: Game): State {
    findWinner(game.board)?.let {
      return if (game.player1.mark == it) State.PLAYER_1_WIN
             else State.PLAYER_2_WIN
    }
    if (game.board.isComplete()) {
      return State.DRAW
    }
    return if (game.state == State.PLAYER_1_MOVE) State.PLAYER_2_MOVE
           else State.PLAYER_1_MOVE
  }

  fun findWinner(board: Board): Mark? = TODO()
  fun Board.isComplete(): Boolean = TODO()
}
```


Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models


Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models


Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models

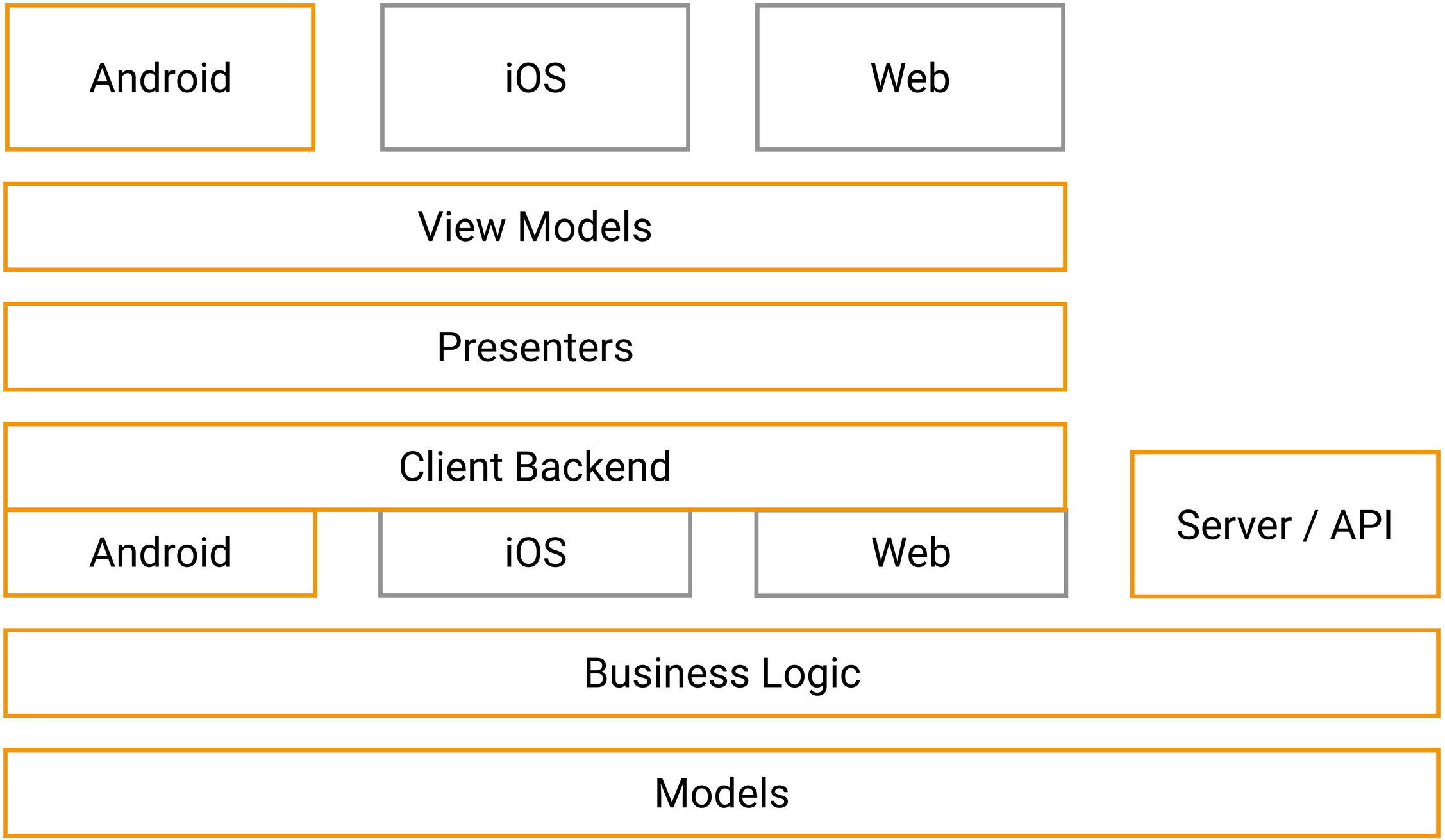

Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models

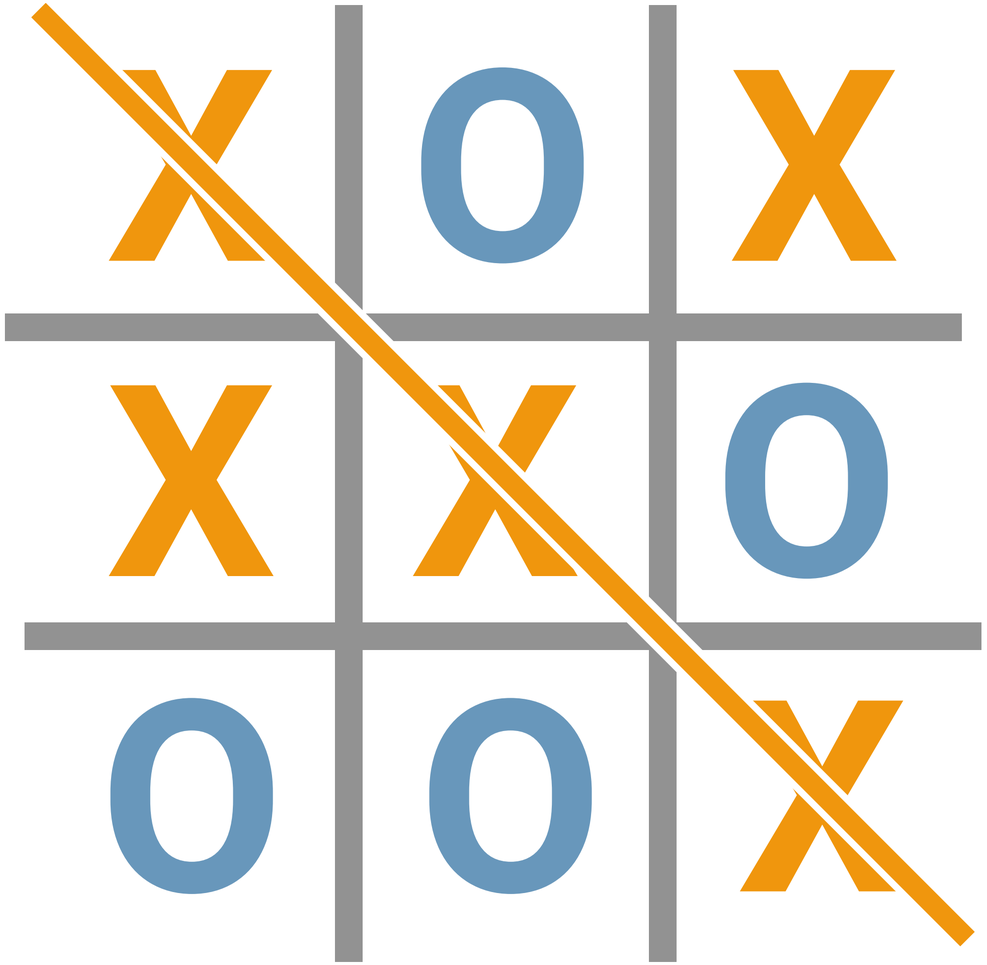
```
class GameView(context: Context, attrs: AttributeSet)
    : Consumer<GamePresenter.UiModel> {
  fun accept(model: GamePresenter.UiModel) {
    // TODO bind to view...
  }
}
```

iOS???

```
function update(model) {
  // TODO bind to DOM/template/JSX/whatever...
}
```

```
@POST @Path("/api/move")
fun Game move(
    @QueryParam("id") id: Long,
    @QueryParam("row") row: Int,
    @QueryParam("col") col: Int) {
  // TODO check business logic, persist, return udpated game ...
}
```


Android
iOS
Web
View Models
Presenters
Client Backend
Android
iOS
Web
Server / API
Business Logic
Models


X O X
X X O
O O X

@JAKEWHARTON

IT'S A KOTLIN, KOTLIN, KOTLIN WORLD!